Министерство образования и науки РФ

Нижегородский государственный университет им. Н.И. Лобачевского
Национальный исследовательский университет

Ф О Р У М
СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ
В ОБРАЗОВАНИИ, НАУКЕ И ПРОМЫШЛЕННОСТИ

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ

МАТЕРИАЛЫ XII ВСЕРОССИЙСКОЙ КОНФЕРЕНЦИИ

Нижний Новгород, 26–28 ноября 2012 г.

Нижний Новгород
Издательство Нижегородского госуниверситета
2012

УДК 681.3.012:51
ББК 32.973.26−018.2:22
В93

В93 **Высокопроизводительные параллельные вычисления на кластерных системах.** Материалы XII Всероссийской конференции (Н. Новгород, 26–28 ноября 2012 г.) / Под ред. проф. В.П. Гергеля. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2012. – 496 с.

Отв. за выпуск К.А. Баркалов

ISBN 978-5-91326-223-3

Сборник материалов Двенадцатой Всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах», состоявшейся 26−28 ноября 2012 г. в рамках форума «Суперкомпьютерные технологии в образовании, науке и промышленности» на базе Нижегородского государственного университета им. Н.И. Лобачевского, содержит тезисы докладов, посвященных основным направлениям развития проблематики высокопроизводительных вычислений в науке и образовании: принципам построения кластерных систем и методам управления параллельными вычислениями, параллельным методам и программным системам решения сложных вычислительных задач, технологиям распределенных вычислений и др.

ISBN 978-5-91326-223-3 ББК 32.973.26−018.2:22

Поддержка конференции

Российский фонд фундаментальных исследований

Компания Intel

Компания Hewlett-Packard

Компания NVIDIA

Компания «Ниагара Компьютерс»

Информационно-аналитический центр Parallel.Ru

Газета научного сообщества «Поиск»

© Нижегородский госуниверситет
им. Н.И. Лобачевского, 2012

ПРЕДИСЛОВИЕ

26–28 ноября 2012 г. Суперкомпьютерным консорциумом университетов России на базе Нижегородского государственного университета им. Н.И. Лобачевского при поддержке Российского фонда фундаментальных исследований, корпорации Intel, компаний HP, NVIDIA, «Ниагара Компьютерс» проведена XII Всероссийская конференция «Высокопроизводительные параллельные вычисления на кластерных системах».

Тематика конференции охватывала все основные направления развития высокопроизводительных вычислений в науке и образовании:

- принципы построения кластерных систем и методы управления параллельными вычислениями;
- параллельные методы и программные системы решения сложных вычислительных задач;
- программные среды, средства и инструменты для разработки параллельных программ;
- технологии распределенных вычислений и GRID-технологии;
- научная визуализация и машинная графика для высокопроизводительных вычислений;
- проблемы подготовки специалистов в области параллельных вычислений.

Основными задачами проведения конференции являлись обсуждение различных аспектов организации высокопроизводительных вычислений в кластерных компьютерных системах, научно-практической деятельности исследователей в этой перспективной области развития современных средств вычислительной техники, расширение контактов между специалистами для решения ресурсоемких прикладных задач, обмен опытом учебно-образовательной деятельности при подготовке специалистов в области параллельных вычислений.

Все перечисленные вопросы нашли отражение в настоящем сборнике, куда включены тезисы докладов, представленные на конференцию (публикуются частично в авторской редакции).

КЛАСТЕРИЗАЦИЯ ПОКАЗАТЕЛЕЙ ЭНЕРГЕТИЧЕСКОЙ АКТИВНОСТИ ЗВЕЗД НА БАЗЕ АЛГОРИТМОВ ГУСТАФСОНА-КЕССЕЛЯ И DBSCAN С ПОМОЩЬЮ СУПЕРКОМПЬЮТЕРНОГО КЛАСТЕРА

С.В. Аксёнов, Д.Н. Лайком

Национальный исследовательский Томский политехнический университет
E-mail: axoenowsw@tpu.ru, wedun@tpu.ru

Статья посвящена решению задачи кластеризации звезд на основе их активности и содержит описание используемых методов кластеризации, описание реализации и вывод по проделанному эксперименту. Показано, что решение задачи кластеризации требует высоких вычислительных затрат, и предложено решение.

Введение

Анализ колоссальных объёмов астрофизической информации о физических параметрах и энергии объектов во внеземном пространстве предполагает нахождение групп объектов со схожими характеристиками. Для анализируемой предметной области сложность решения задач кластеризации возрастает многократно как из-за большой размерности анализируемых факторов, так и из-за огромного пространства поиска. Количество объектов в фрагменте базы данных OGLE Европейского космического агентства (ESA), предоставленном Новым университетом г. Лиссабона, [1] превышает 100000 строк. Решение проблемы разделения на кластеры позволяет выявлять скрытые зависимости между анализируемыми особенностями (различными физическими характеристиками), недоступные для восприятия из-за большой размерности задачи. В работе представлен алгоритм распределения вычислений для этой задачи, позволяющий значительно ускорить обработку астрофизических данных.

Кластеризация

Одной из особенностей задачи кластеризации является требование, накладываемое на вид кластера: группы могут принимать любую форму. Подобное ограничение исключает возможность использовать без дополнительных техник алгоритмы как четкого (K-средних [2], самоорганизующиеся карты Кохонена [3]), так и нечеткого (Fuzzy K-means, алгоритм Густафсона-Кесселя [4]) разбиения. Таким образом, возможное решение проблемы – формирование кластера путем предварительного анализа объектов методом Густафсона-Кесселя и включения близких из них в кластер методом DBSCAN (*Density Based Spatial Clustering of Application with Noise*) [5].

Используемые методы

В основе работы метода Густафсона-Кесселя лежит понятие близости объектов к центру предполагаемого кластера и предварительной сортировки объектов относительно этого центра. Формирование разрастающихся кластеров происходит на основе понятия плотности объектов. Для проведения сортировки объектов необходимо задать начальные условия: центры кластеров – и сформировать матрицу нечеткого разбиения, а также точность и экспоненциальный вес. Предварительная сортировка объектов происходит в нескольких потоках. Отсортированные точки обрабатываются на узлах су-

перкомпьютерного кластера. Формирование конечных кластеров происходит методом DBSCAN. Согласно этому методу для формирования кластера необходимо превысить порог плотности minPts. Предварительная сортировка позволяет выделить группы объектов, которые впоследствии образуют кластеры. Необходимость использования именно этого метода, основанного на некоторой заданной плотности объектов, обусловлена неравномерным распределением объектов и необходимостью поиска кластеров сложной формы. На рис. 1 представлен вариант формирования двух кластеров методом DBSCAN. При объединении оба этих кластера будут суммированы в один.

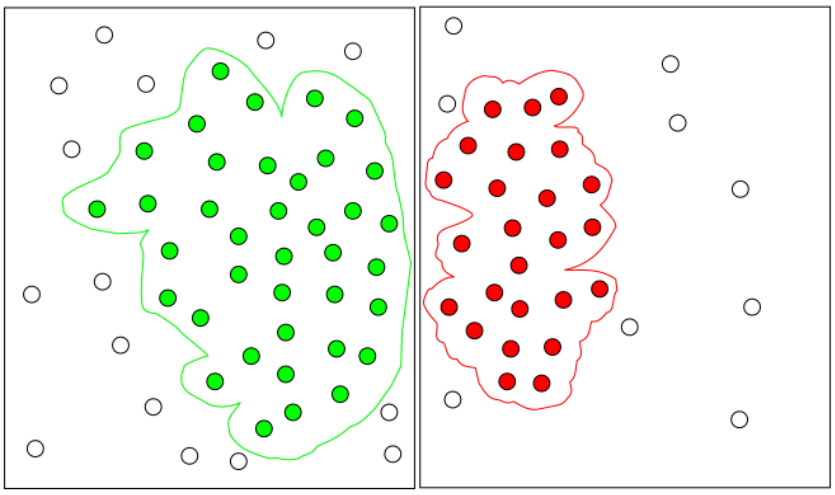


Рис. 1. Формирование кластеров методом DBSCAN на нескольких итерациях процесса кластеризации

Сам метод основан на следующих шагах. Алгоритм начинает работу с произвольной точки, которая еще не была посещена. Если ε-окрестность выбранной точки содержит достаточное количество объектов, то создается кластер. В противном случае точка обозначается как шум для ближайшего возможного кластера. Если точка оказывается частью кластера, то ее ε-окрестность также оказывается частью кластера. Таким образом все точки, найденные в ε-окрестности, добавляются вместе со своими ε-окрестностями.

Реализация

Использование технологий распределенных вычислений на суперкомпьютерном кластере предполагает распределение данных между вычислительными узлами кластера. Для решения поставленной задачи были использованы технологии MPI.NET и Microsoft.Threading.Tasks.

Процесс кластеризации можно описать следующими шагами:

- сформировать матрицу нечеткого разбиения;
- рассчитать центры кластеров;
- рассчитать матрицу ковариации;
- рассчитать расстояние между объектами и центрами кластеров;
- пересчитать элементы матрицы нечеткого разбиения;
- передать сформированные группы объектов на узлы кластера;
- проверить все объекты на возможность формирования кластеров методом DBSCAN.

Необходимость предварительной сортировки

Необходимость предварительной сортировки объясняется повышением производительности алгоритма и отсутствием необходимости поиска неучтенных кластеров.

Алгоритм нечеткой кластеризации DBSCAN показывает лучшие результаты на отсортированном наборе входных данных. Объекты, попавшие на обработку в узлы кластера, имеют большую вероятность формирования кластера с меньшим количеством шума.

Предварительно из огромного массива характеристик выбирается случайным образом по N*10000 строк, где N – количество узлов, используемых для поиска кластеров классическим методом DBSCAN, и каждый фрагмент из 10000 строк рассылается по узлам. Основная цель последующей обработки заключается в том, чтобы получить грубый набросок картины кластеров методом Густафсона-Кесселя. Отметим, что каждый узел запускает кластеризацию на своем наборе независимо от других узлов. Внутри каждого из узлов используется распараллеливание на уровне ядер, т.к. элементы матрицы нечеткого разбиения, матрицы ковариации характеристик рассчитываются независимо друг от друга и код обработки лишен рекуррентных процедур.

Получение набора независимых кластеризаций, каждая из которых выражена центрами кластеров и матрицей ковариации, ставит вопрос о нахождении наиболее подходящей для всего набора. Решение этой задачи заключается в широковещательной рассылке тестового набора 1000 векторов, выбранных случайным образом из первоначальной выборки, и определении их принадлежности к найденным кластерам внутри каждого из узлов. Для этого мы соотносим проценты объектов, попавших в каждый кластер на разных узлах. Понятно, что если большинство объектов (в наших экспериментах мы брали 80%), принадлежащих первому кластеру в первом узле, принадлежат также третьему кластеру во втором узле, это означает, что данные кластеры одинаковые. Проверяя таким образом принадлежности объектов к кластерам в разных узлах, происходит выделение похожих кластеризаций. Из набора найденных похожих кластеризаций случайным образом (т.к. их центры и матрицы ковариации приблизительно одинаковы) выбирается одна, которая признается за базовую, она рассылается на все узлы.

Теперь все узлы получают приблизительно равные фрагменты начальной выборки и кластеризуют её. Далее все точки, которые попали в первый кластер, отправляются на обработку на первый узел, которые попали во второй кластер – идут на обработку на второй узел и т.д. Таким образом, предварительная обработка позволяет разбросать объекты между узлами так, чтобы в узле располагались только близкие объекты.

Тестирование и результаты

Для проведения тестирования были использованы астрофизические данные, предоставленные Новым университетом г. Лиссабона [1]. Характеристики тестовых данных представлены в табл. 1.

Таблица 1. Характеристики наборов тестовых данных Dataset1 и Dataset2

Набор данных	Точность	Число объектов	eps	minPts	Число узлов кластера
Dataset1	0,001	10965	0,2	20	10
Dataset2	0,001	100787	0,2	20	10

Результаты тестирования подтверждают эффективность используемого метода. Предварительная сортировка входных данных позволила увеличить производительность алгоритма DBSCAN.

Выводы

В результате проделанной работы было предложено решение задачи кластеризации данных методом DBSCAN с предварительной сортировкой объектов методом Гу-

стафсона-Кесселя. Кластеризация объектов является сложной и ресурсоемкой задачей. Предложенное решение позволяет повысить производительность метода и решить проблему некорректного формирования кластеров и может быть рекомендовано к использованию при решении задачь кластеризации данных.

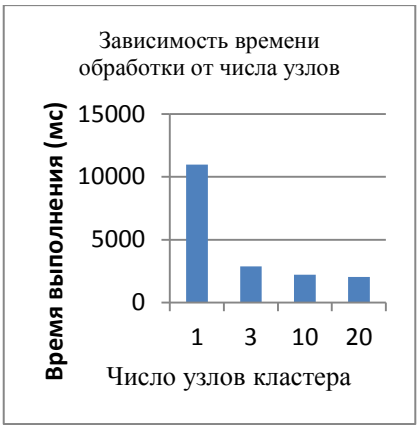


Рис. 2. Время обработки Dataset1

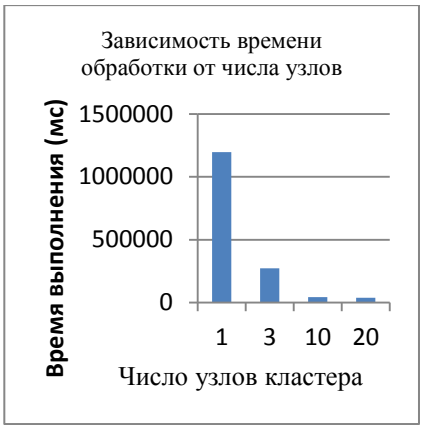


Рис. 3. Время обработки Dataset2

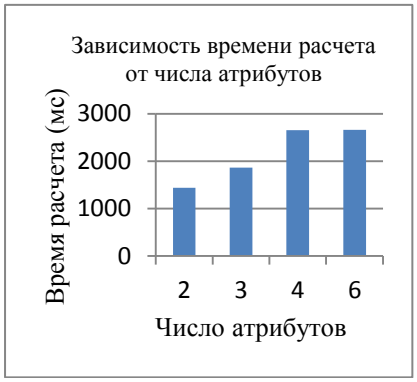


Рис. 4. Время обработки Dataset1

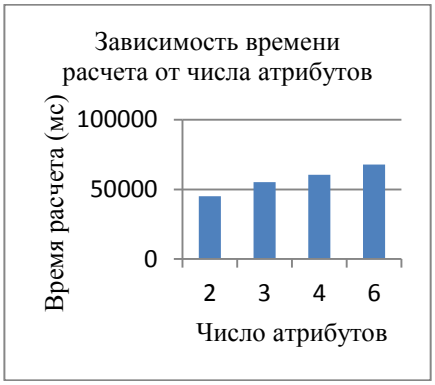


Рис. 5. Время обработки Dataset2

Литература

1. База данных характеристик звезд. URL – [http://sirius.astrouw.edu.pl/ogle]
2. MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations // Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 1967. P. 281–297.
3. Haykin S. Self-organizing maps. Neural networks – A comprehensive foundation (2nd ed.). Prentice-Hall, 1999.
4. Hoeppner F., Klawonn F., Kruse R., Runkler T. Fuzzy Cluster Analysis, Methods for classification, data analysis and image recognition. Wiley, 2000.
5. Sander J., Ester M., Kriegel H.-P., Xu X. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications // Data Mining and Knowledge Discovery. Berlin: Springer-Verlag, 1998. №2 (2). P.169–194.

ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ ОПРЕДЕЛЕНИЯ ОПТИМАЛЬНЫХ РЕЖИМОВ ГАЗОТРАНСПОРТНОЙ СИСТЕМЫ НА СТАДИИ ЕЕ ПРОЕКТИРОВАНИЯ ИЛИ МОДЕРНИЗАЦИИ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИЙ МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Л.Г. Афраймович[1], К.И. Дикарев[1], А.А. Макаров[2], Г.И. Наместников[2], Д.В. Парфёнов[2], С.Ф. Перетрухин[3], М.Х. Прилуцкий[1], Н.В. Старостин[1], А.В. Филимонов[1]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]ОАО «Гипрогазцентр», Нижний Новгород
[3]Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Рассматривается задача расчета оптимизационных режимов для участков газотранспортной системы, которая заключается в определении необходимой модернизации системы и расчете таких ее параметров, которые доставляют минимум функции суммы эксплуатационных и капитальных затрат на разработку и реализацию дополнительных элементов газотранспортной системы. Рассматриваются алгоритмы решения указанной задачи, а также средства повышения эффективности решения путем привлечения параллельной вычислительной среды.

Введение

В настоящее время в газовой промышленности России первостепенное значение приобретает проблема определения оптимальных объемов добычи газа, обеспечивающих потребности внутри страны и на экспортных рынках, в зависимости от различных прогнозных сценариев спроса на газ, возможностей ресурсной базы и обеспеченности инвестициями. При этом актуальной становится задача обеспечения оптимальных и эффективных режимов функционирования крупных участков газотранспортных систем в рамках линейно-производственных управлений магистральных газопроводов.

Наряду с задачами эффективного диспетчерского управления при помощи диалоговых программных средств, предназначенных для нахождения параметров оптимальных режимов, возникает задача оптимального проектирования или модернизации существующих участков газотранспортных систем. Такая задача в ряде случаев подразумевает расчетную оценку оптимальных режимов системы, в том числе и на стадии проектной разработки.

Учитывая сложность рассматриваемых газотранспортных систем, для ее решения необходимо построение полноценных компьютерных моделей данных участков, а также привлечение современных численных методов оптимизации и подходов теории многоуровневых иерархических систем. В рамках данной работы рассматривается задача расчетного определения оптимальных режимов функционирования газотранспортной системы по критериям минимизации суммарных экономических затрат.

1. Постановка задачи

Газотранспортная система представляет собой совокупность элементов, содержащих входы и выходы системы, коммуникационные элементы и «обобщенные» элемен-

ты. Каждый обобщенный элемент является участком газотранспортной системы и представляет собой совокупность ряда существующих и/или проектируемых объектов. Обобщенный элемент может включать в себя компрессорные станции, линейную часть магистрального газопровода, регуляторы давления и т.д. Обобщенные элементы описываются как «черные ящики», задавая для которых коммерческий расход газа на выходе, температуру газа, его относительную плотность по воздуху и теплотворную способность на входе, давление на входе и выходе можно получить коммерческий расход, температуру газа, его относительную плотность по воздуху, теплотворную способность на входе и затраты (эксплуатационные и капитальные), связанные с реализацией полученных параметров.

Пусть $G=(V,A)$ – взвешенный одностороннесвязный ориентированный граф без петель и контуров, $A \subseteq V^2$. Множество вершин графа V соответствует элементам системы, множество дуг A – связям между ними. Через $V_{вх}$, $V_{вых}$ обозначим множества входов и выходов системы, соответственно, $V_{вх}, V_{вых} \subseteq V$. Для каждого из элементов v , $v \in V \setminus (V_{вх} \cup V_{вых})$, заданы следующие функции: $q_v^{out}(q^{in}, p^{in}, p^{out})$ – определяют расход на выходе элемента v при заданных расходе на входе, давлениях на входе и выходе этого элемента; $c_v^1(q^{in}, q^{out}, p^{in}, p^{out})$ и $c_v^2(q^{in}, q^{out}, p^{in}, p^{out})$ – определяют эксплуатационные и капитальные затраты элемента v при заданных расходах на входе и выходе, давлениях на входе и выходе этого элемента. Для каждого из входов системы заданы следующие параметры: $p_v^{вх}$ – давление на входе, $q_v^{вх}$ – максимально возможный расход на входе, $v \in V_{вх}$. Для каждого из выходов системы заданы: $p_v^{вых}$ – требуемое давление на выход, $q_v^{вых}$ – требуемый объем расхода на выходе, $v \in V_{вых}$. Обозначим через x_{ij} неизвестную величину, определяющую расход вдоль дуги (i,j) , $(i,j) \in A$; через z_i, y_i – неизвестные величины давлений на входе и выходе вершины i, соответственно, $i \in V$. Для удобства введем следующие обозначения: $In(i) = \{ j \mid (j,i) \in A \}$, $Out(i) = \{ j \mid (i,j) \in A \}$, $i \in V$.

Исходная задача ставится как следующая задача математического программирования:

$$\sum_{(i,j)\in A} x_{ij} \le q_i^{вх},\ i \in V_{вх}, \tag{1}$$

$$\sum_{(j,i)\in A} x_{ji} = q_i^{вых},\ i \in V_{вых}, \tag{2}$$

$$z_i = p_i^{вх},\ i \in V_{вх}, \tag{3}$$

$$y_i = p_i^{вых},\ i \in V_{вых}, \tag{4}$$

$$z_i = y_j,\ j \in Out(i),\ i \in V, \tag{5}$$

$$\sum_{(i,j)\in A} x_{ij} - q_i^{out}\Big(\sum_{(j,i)\in A} x_{ji}, y_i, z_i\Big) = 0,\ i \in V \setminus (V_{вых} \cup V_{вх}), \tag{6}$$

$$x_{ij}, y_{ij} \ge 0,\ (i,j) \in A, \tag{7}$$

$$\sum_{i \in V \setminus (V_{вых} \cup V_{вх})} \Big(c_i^1\Big(\sum_{(j,i)\in A} x_{ji}, \sum_{(i,j)\in A} x_{ij}, y_i, z_i\Big) + c_i^2\Big(\sum_{(j,i)\in A} x_{ji}, \sum_{(i,j)\in A} x_{ij}, y_i, z_i\Big)\Big) \to \min . \tag{8}$$

Приведенные ограничения означают: (1) – ограничения на максимальные объемы расходов на входах; (2) – ограничения на требуемые объемы расходов на выходах; (3), (4) – ограничения на заданные давления на входах и выходах, соответственно; (5) – ограничения равенства входных и выходных давлений смежных вершин; (6) – ограничения баланса расходов в вершинах с учетом их возможных потерь; (7) – естественные

ограничения на переменные. Критерий (8) определяет минимизацию суммарных обобщенных затрат.

Содержательно приведенную задачу можно описать следующим образом. Требуется, учитывая значения параметров для входных и выходных элементов, определить для каждого обобщенного элемента значения коммерческих расходов газа, давления, температуры, относительной плотности газа по воздуху и его теплотворной способности таким образом, чтобы суммарные приведенные затраты всех обобщенных элементов в системе были минимальны.

2. Алгоритм решения

Следует отметить, что представленная задача относится к классу задач распределения ресурсов в иерархических системах с активными элементами [1]. Для ее решения был разработан подход, основанный на методах распределения ресурсов в иерархических системах [2].

Для решения задачи математического программирования (1)-(8) разработан и внедрен двухуровневый итерационный алгоритм, в результате работы которого получаются локально-оптимальные решения задачи минимизации стоимости для разветвленной газотранспортной системы, на основании которых инженеры по технологическому проектированию могут выработать приемлемые стратегии работы обобщенных элементов, то есть эффективные варианты модернизации с точки зрения минимума экономических затрат. После получения конечного ряда таких стратегий могут быть выбраны наиболее подходящие из них, с точки зрения политических или интуитивных решений.

Реализация алгоритма оптимизации режимов работы участков газотранспортной сети предполагает выбор оптимальных режимов функционирования отдельных элементов этой сети, представимых в виде «черных ящиков». Поскольку «черные ящики» описываются функциями, некоторые элементы поведения которых неизвестны, поиск оптимальных параметров на втором уровне алгоритма осуществляется с использованием хорошо распараллеливаемых схем перебора. Созданная диалоговая программная система поиска оптимальных режимов работы для участков газотранспортной системы была реализована с использованием компонента TPL (Task Parallel Library), входящего в состав платформы .NET 4.0 [3].

Схема работы программы на основе описанного алгоритма включает в себя формирование «подобластей интереса», содержащих локальные экстремумы в области параметров «черных ящиков». Данное предварительное исследование функций «черных ящиков» также целесообразно вести с применением многопоточных распараллеливаемых схем.

Для решения указанной задачи расчета оптимальных режимов газотранспортной системы было разработано диалоговое программное обеспечение, которое позволяет осуществлять интерактивное задание параметров и топологии газотранспортных систем и решение вышеописанной оптимизационной задачи.

Параллельная реализация алгоритма позволила осуществить решение поставленной задачи для реальной газотранспортной системы. Созданный программный продукт находится на стадии внедрения в ОАО «Гипрогазцентр».

Заключение

Предложена эффективная многопоточная схема для решения задач расчета стационарных режимов функционирования газотранспортных систем, имеющих структуру многоуровневой иерархической системы, по критерию минимизации экономических затрат. Предложенная схема реализуется в рамках специализированного программного обеспечения, предназначенного для внедрения в практику групп технологического проектирования магистральных газопроводов.

Работа выполнена при частичной финансовой поддержке Минобрнауки России (гос. соглашение о предоставлении гранта №14.B37.21.0878).

Литература

1. Прилуцкий М.Х., Дикарев К.И. Оптимизационные задачи согласования параметров для участков газотранспортной системы // Системы управления и информационные технологии, 2011. Т. 45. № 3.1. С. 185-189.
2. Афраймович Л.Г., Прилуцкий М.Х. Многоиндексные задачи оптимального планирования производства // Автоматика и телемеханика, 2010. № 10. С. 148-155.
3. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2011. – 1392 с.

МЕТОД РАЗБИЕНИЯ МНОГОМАШИННОЙ СИСТЕМЫ НА КОМПЛЕКСЫ ПРИ ОРГАНИЗАЦИИ СБОЕ- И ОТКАЗОУСТОЙЧИВЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

И.В. Ашарина[1], А.В. Лобанов[2]

[1]Московский государственный институт электронной техники
[2]ОАО «НИИ «Субмикрон», Москва

Представлен метод выделения в многомашинной вычислительной системе или сети ЦВМ произвольной структуры заданного количества комплексов требуемой отказоустойчивости с целью организации сбое- и отказоустойчивого параллельного исполнения ряда задач, взаимообменивающихся информацией. Определены достаточные условия существования таких комплексов. Предложен новый алгебраический подход к реализации представленного метода, обеспечивающий наглядность и достоверность получаемых результатов.

Введение

Суть концепции облачных вычислений заключается в предоставлении конечным пользователям удаленного динамического доступа к услугам, вычислительным ресурсам и приложениям через Интернет. Компьютеры, осуществляющие облачные вычисления, называются вычислительным облаком. Особенностями облачных вычислений является то, что при использовании любого проприетарного ПО или чужого сервера пользователь становится беззащитным перед разработчиком этого ПО.

Другая модель облачных вычислений – т.н. частное облако – обладает многими преимуществами по сравнению с компьютерной средой на базе публичных облаков. Различие между этими моделями состоит в том, что в частном облаке управление данными и процессами осуществляется внутри организации. В этом случае отсутствуют такие проблемы, как ограничение пропускной способности сети, угрозы безопасности и необходимость нормативного соответствия, которые могли бы возникнуть при использовании публичных облаков посредством открытых сетей общего пользования.

Однако ни одно вычислительное облако не обеспечивает вычислений с требуемой степенью достоверности. Предлагаемый подход обеспечивает решение этой задачи.

1. Отказоустойчивость параллельных вычислений в сетях ЦВМ

При организации многозадачных параллельных отказоустойчивых вычислений в многомашинных вычислительных системах (МВС) и сетях ЦВМ одной из важнейших является задача достижения согласованности информации в различных ЦВМ системы в условиях возникновения допустимых неисправностей, формулируемая как проблема ***взаимного информационного согласования (ВИС)*** [1]. В данной работе рассматривается МВС произвольной структуры, отображаемой ориентированным графом (орграфом) G, ЦВМ (вершины) которой соединены двухточечными симплексными каналами связи (дугами). ***Модель неисправности ЦВМ*** – враждебная, при которой поведение неисправной ЦВМ может быть произвольным, неодинаковым по отношению к взаимодействующим с ней другим ЦВМ МВС, в том числе подобным «злонамеренному».

Комплексом k_i (i=1, 2,…, l) из заданного множества $K=\{k_1, k_2,\ldots, k_l\}$ называется подсистема (орподграф), удовлетворяющая достаточным условиям, состоящим из трех частей [2, 3]: Ч1) наличие в k_i орподграфа H_i, гомеоморфного полному орграфу M с количеством вершин более $3m_i$, где m_i принадлежит интервалу $m_i^{\max} - m_i^{\min}$ и задает допустимое количество неисправных ЦВМ в k_i. Вершины H_i, соответствующие вершинам M, называются основными и составляют множество X_i. Остальные вершины k_i называются неосновными и составляют множество N_i; Ч2) наличие исходящих, не пересекающихся далее путей из каждой неосновной вершины k_i к не менее чем $2m_i+1$ конечным основным вершинам, причем в каждом пути имеется только одна основная вершина; Ч3) наличие входящих путей в каждую неосновную вершину k_i от не менее чем $2m_i+1$ основных вершин, причем эти пути пересекаются только в этой неосновной вершине и в каждом пути имеется только одна основная вершина, являющаяся начальной.

При сбое- и отказоустойчивом исполнении в системе l взаимообменивающихся параллельных задач необходимо наличие в системе $K=\{k_1, k_2,\ldots, k_l\}$ комплексов, причем i-ю задачу решает k_i. В таких системах процесс достижения **системного ВИС (СВИС)** можно рассматривать в виде двух последовательных этапов: 1) **внутрикомплексное ВИС** для каждого k_i, при котором после внутрикомплексного обмена во всех исправных ЦВМ k_i вычисляется одинаковый вектор согласованных значений этого k_i, 2) **межкомплексное ВИС**, включающее межкомплексный обмен согласуемой информацией и вычисление **вектора СВИС**, одинакового в каждой исправной ЦВМ системы и содержащего вектор согласованных значений всех ЦВМ системы.

Алгоритм межкомплексного ВИС в многокомплексной МВС можно построить, если в G выделены комплексы требуемой структуры и между каждой возможной парой комплексов определена среда межкомплексных пересылок. Присутствие допустимых неисправностей в среде не должно препятствовать межкомплексному ВИС в этой паре. Ниже рассматривается только задача выделения заданного множества комплексов.

Предлагаемый метод выделения комплексов состоит в следующем. Пусть в некотором орграфе G системы необходимо последовательно выделить $K=\{k_1, k_2,\ldots, k_l\}$. Вначале все вершины орграфа G считаются незадействованными. Выделение k_1 начинается с задания текущего значения $m_1=m_1^{\max}$. Из состава незадействованных вершин формируются всевозможные множества O_1^1, O_1^2,..., O_1^r вершин-кандидатов в основные вершины k_1 мощностью $n_1=3m_1+1$. Затем выбирается O_1^1 и среди множества всевозможных пронумерованных вариантов структур-кандидатов, определяемых O_1^1, последовательно находится вариант орподграфа H_1, удовлетворяющий достаточным условиям Ч1-Ч3 существования k_1. При положительном исходе процесс выделения k_1 прерывается и считается, что k_1 с подорграфом H_1 выделен. Из множества незадействованных вершин G исключаются вершины из k_1, и осуществляется переход к аналогичному выделению k_2 в орподграфе, порожденном в графе G оставшимися незадействованными вершинами. Иначе делается вывод о невозможности выделения k_1 для заданного O_1^1, выбирается O_1^2 и повторяется весь анализ. Если невозможно выделение k_1 также и для O_1^r, то делается вывод о невозможности выделения k_1 при заданном m_1. Тогда, если $m_1 > m_1^{\min}$, повторяются все действия для $m_1:=m_1-1$. Если же невозможно выделение k_1 при любом допустимом значении m_1, то делается вывод о невозможности выделения заданного $K=\{k_1, k_2,\ldots, k_l\}$. Если невозможно выделение k_i (i>1), делается вывод о неудачном прерывании процесса выделения k_{i-1} и осуществляется возврат к продолжению прерванного процесса выделения k_{i-1}. Для сокращения объема вычислений перед переходом к выделению очередного k_i (i>1) выполняется проверка достаточности количества незадей-

ствованных вершин для выделения оставшихся комплексов. При недостаточности выполняется переход к продолжению прерванного выделения предыдущего комплекса. В результате применения предложенного метода осуществляется полный перебор возможных вариантов выделения $K=\{k_1, k_2,\ldots, k_l\}$ и, если имеется успешный вариант, он будет найден.

Проверку $O_i^t=\{\alpha, \gamma, \chi,..., \varepsilon\}$ на удовлетворение Ч1-Ч3 можно осуществлять следующим образом. Представим орподграф из вершины α, соединенной в орграфе G исходящими дугами с вершинами β, γ,..., ζ, в виде **ДНФ исходящей смежности для α в орграфе *G* (ДНФИС (α, *G*))**: $\alpha\beta\vee\alpha\gamma\vee...\vee\alpha\zeta$. В этой ДНФ не допускается использование коммутативного закона. Построим подобные ДНФИС для всех вершин орграфа G. Выражение ДНФИС (α, G) также является выражением **ДНФ путей единичной длины из α в *G* (ДНФП (1, α, *G*)).** Для формирования выражения **связки α→λ в орграфе *G* (связки (α→λ, *G*))**, отображающей всевозможные простые пути из α в λ, заменим в каждом терме из ДНФП (1, α, G) последний его символ на соответствующую ДНФИС и при помощи преобразования вида $(A\vee B)\cdot(C\vee D)=AC\vee AD\vee BC\vee BD$ построим **ДНФ путей двойной длины из α в *G* (ДНФП (2, α, *G*))** без применения идемпотентного и коммутативного законов, сохраняя имевшуюся последовательность символов в термах. В ДНФП (2, α, G) продолжим итеративно подобные подстановки и преобразования. После каждой j-й (j=1, 2,…) итерации находим в получаемой ДНФП (j, α, G) каждый терм, заканчивающийся символом λ, и переносим его (с исключением из ДНФП (j, α, G)) в виде терма в формируемое выражение **ДНФ связки α→λ внутри *G* (ДНФС (α→λ, *G*)**. Из полученной ДНФП (j, α, G) исключаем каждый терм, заканчивающийся обозначением вершины стока орграфа G или символом, уже имеющимся в этом терме (исключение контура), а также каждый терм, в котором появляется обозначение вершины из O_i^t, отличной от α и λ. В результате таких преобразований для любого конечного орграфа G за конечное число $q>0$ итераций получаемая ДНФП (q, α, G) из-за переносов и исключений становится пустой, а в ДНФС (α→λ, G) появляются термы, каждый из которых описывает один из путей связки, термы не повторяются, и все термы отображают всевозможные простые пути в связке (α→λ, G).

Логическое выражение, отображающее всевозможные **исходящие пучки (α, $O_i^t\backslash\alpha$, *G*)** путей, проходящих внутри G из вершины $\alpha\in O_i^t$ во все другие вершины β, χ,..., ε, из O_i^t, строится путем конъюнкции всех ДНФС (α→β, G), (α→γ, G),…, (α→ε, G), ее преобразования к виду ДНФ без применения идемпотентного и коммутативного законов с сохранением имеющейся последовательности символов в термах и исключением каждого терма с повторяющимися символами, отличными от символа α. В результате получается выражение **ДНФ исходящих пучков из α в вершины из $O_i^t\backslash\alpha$ внутри *G* (ДНФИП (α, $O_i^t\backslash\alpha$, *G*)**, каждый терм которого отображает совокупность путей одного исходящего пучка (α, $O_i^t\backslash\alpha$, G), а все термы – всевозможные такие пучки.

Метод выделения в G для $O_i^t=\{\alpha, \beta, \gamma,..., \varepsilon\}$ очередного k_i, такого, что для каждой вершины из O_i^t, например $\delta\in O_i^t$, существует хотя бы один исходящий пучок (δ, $O_i^t\backslash\delta$, G), состоит в следующем. Из всех ДНФИП (α, $O_i^t\backslash\alpha$, G), (β, $O_i^t\backslash\beta$, G), (γ, $O_i^t\backslash\gamma$, G),..., (ε, $O_i^t\backslash\varepsilon$, G) строится конъюнкция и преобразуется к виду ДНФ без применения идемпотентного и коммутативного законов с сохранением имеющейся последовательности символов в термах из ДНФИП. В результате получается **ДНФ всевозможных орподграфов в орграфе *G*, гомеоморфных полному орграфу с множеством основных**

вершин O_i^t**={α, β, γ,..., ε} (ДНФОГП (**O_i^t**, *G*))**, каждый терм которой отображает один такой возможный орподграф $p_i^{t,s}$ (s=1,...), а все термы – всевозможные такие орподграфы из множества $P_i^t = \{p_i^{t,1}, p_i^{t,2},...\}$. Каждый $p_i^{t,s}$ удовлетворяет Ч1. Для каждого $p_i^{t,s}$ все вершины из $p_i^{t,s}$, не принадлежащие O_i^t, составляют множество $N_i^{t,s}$ вершин – кандидатов в неосновные вершины k_i. Для каждой неосновной вершины должны выполняться условия Ч2 и Ч3. Обеспечение Ч2, например, для $\eta \in N_i^{t,s}$ можно достичь путем успешного построения ДНФИП (η, $O_i^{t,v}$, *G*) (здесь $O_i^{t,v}$ является очередным анализируемым v-м подмножеством мощностью $2m_i$+1 из всех таких пронумерованных возможных подмножеств, построенных из O_i^t), если все предыдущие $O_i^{t,1}$,…, $O_i^{t,v-1}$ оказались безуспешными. При этом возможно, что в найденных путях будут иметься внутренние вершины, не принадлежащие $N_i^{t,s}$, которые необходимо, в свою очередь, ввести в k_i (в $N_i^{t,s}$) и для каждой из них аналогично обеспечить Ч2 и Ч3.

Анализ и обеспечение Ч3 для каждой вершины, принадлежащей $N_i^{t,s}$, например η, осуществляется путем построения для очередного выбранного подмножества $O_i^{t,v}$={β, γ,..., ε} **ДНФ пучков, сходящихся в η из** $O_i^{t,v}$**, в G (ДНФПС (η,** $O_i^{t,v}$**, *G*))** из конъюнкции связок (β→η, G), (γ→η, G),…, (ε→η, G), каждый терм которой отображает один из искомых пучков, а все термы – всевозможные такие пучки.

Предложенный метод построения ДНФП (q, α, G) (q=1, 2,...) является, по сути, новой алгебраической формой алгоритма фронта волны [4]. При этом получаемое алгебраическое выражение можно эффективно применять для других задач анализа графов и орграфов, связанных с определением путей и их взаимоотношений.

2. Метод определения достаточной среды межкомплексной посылки при организации сбое- и отказоустойчивых параллельных вычислений в сетях ЦВМ

Второй этап достижения СВИС включает межкомплексный обмен согласуемой информацией и вычисление вектора СВИС в каждой ЦВМ системы. Рассмотрим i-й комплекс-источник, j-й комплекс-получатель и их среду межкомплексной посылки. Выделим в множество $S_{i\to j}$ каждую вершину α i-го комплекса, имеющую непустую ДНФИС (α, α∪$R_{i\to j}$∪K_j). Для каждой вершины из $R_{i\to j}$∪K_j, например β, построим ДНФИС (β, β∪$R_{i\to j}$∪K_j). По результатам внутрикомплексного ВИС в i-м комплексе-источнике все исправные вершины множества $S_{i\to j}$ должны иметь правильный вектор согласованных значений i-го комплекса. В то же время не более m_i неисправных вершин из множества $S_{i\to j}$ могут иметь произвольные значения вектора, отличные от правильного. Отсюда следует правильность следующего утверждения.

Утверждение 1. Для достижения межкомплексного ВИС необходимо $|S_{i\to j}| \geq 2m_i+1$.

Рассмотрим достаточные условия вычисления в вершине α_j определенного значения вектора W_i, совпадающего с правильным значением вектора ***согласованных значений*** V_i i-го комплекса-источника. Назовем ***вершиной 0-го ранга*** такую вершину α_j, в которой согласованное значение W_i, совпадающее с правильным V_i, может быть вычислено только по результатам посылки копий V_i согласованного значения вершин комплекса-источника в эту вершину α_j без предварительного вычисления этого согласованного значения хотя бы в одной вершине из K_j. Если в вершине α_j для вычисления

правильного согласованного значения W_i необходимо использовать копию этого согласованного значения, предварительно вычисленного, например, в вершине γ_j ранга t, то эта вершина α_j приобретает ***ранг не менее t+1***.

Рассмотрим путь из граничной вершины λ_i, принадлежащий некоторому сходящемуся пучку ($U^{t,s}_{i\to j}$, α_j). Возможны следующие четыре типа такого пути: ***a*** – путь состоит только из одной дуги, образующей связь между начальной и конечной вершинами; ***b*** – все промежуточные вершины пути принадлежат орподграфу $R_{i\to j}$; ***c*** – все промежуточные вершины пути принадлежат j-му комплексу; ***d*** – часть промежуточных вершин пути принадлежит орподграфу $R_{i\to j}$, а другая часть – j-му комплексу.

Назовем орподграфом вида A (B, C, D) сходящегося пучка ($U^{t,s}_{i\to j}$, α_j) орподграф, содержащий все пути типа a (b, c, d соответственно) этого пучка и только их. Аналогично, орподграфом вида AB (AC, AD,…, ABC,…, $ABCD$) сходящегося пучка ($U^{t,s}_{i\to j}$, α_j) будет орподграф, содержащий все пути типа a и b (a и c, a и d,…, a и b и c, …, a и b и c и d соответственно) этого пучка и только их. Из простого анализа можно вывести следующие достаточные условия вычисления согласованного значения W_i=V_i в вершине α_j ранга 0. **У1**) Если в орподграфе вида A сходящегося пучка ($U^{t,s}_{i\to j}$, α_j) имеются $2m_i$+1 путей, то однократной посылки копий согласованного значения i-го комплекса-источника по этим путям достаточно для вычисления в вершине α_j при помощи функции мажорирования правильного согласованного значения i-го комплекса-источника. Аналогично, достаточными условиями являются **У2**) наличие $2(m_i+m_{i\to j})$+1 путей в орподграфе вида B либо вида AB, **У3**) наличие $2(m_i+m_j)$+1 путей в орподграфе вида C либо вида AC, **У4**) наличие $2(m_i+m_j+m_{i\to j})$+1 путей в орграфе любого другого из оставшихся видов.

В дополнение к вышеприведенным типам путей, ведущим в вершину α_j из вершины $\lambda_i \in S_{i\to j}$, назовем ***путем типа e*** простой путь, ведущий из некоторой вершины γ_j в вершину α_j, внутренними вершинами которого являются только вершины, принадлежащие множеству K_j. Пусть посылка копии согласованного значения W_i, вычисленного в вершине $\gamma_j \in Z^k_{i\to j}$, в вершину α_j должна осуществляться по пути типа e. В дополнение к вышеприведенным видам орподграфов сходящегося пучка ($U^{t,s}_{i\to\alpha_j}$, α_j) назовем орподграф сходящегося пучка ($Z^k_{i\to j}$, α_j), содержащий все пути типа e этого пучка и только их, орподграфом вида E. Аналогично, орподграфом вида AE (BE,…, $ABCDE$) сходящегося пучка ($U^{t,s}_{i\to\alpha_j} \cup Z^k_{i\to j}$, α_j) будет орподграф, содержащий все пути типа a и e (b и e,..., a и b и c и d и e соответственно) этого пучка и только их.

Достаточными условиями вычисления согласованного значения W_i=V_i в вершине α_j ранга k>0 будут**: У5**) если в орподграфе вида E сходящегося пучка ($Z^{k,l}_{i\to j}$, α_j) имеются $2m_j$+1 путей, то однократной посылки копий согласованного значения i-го комплекса-источника по этим путям достаточно для вычисления в вершине α_j при помощи функции мажорирования правильного согласованного значения i-го комплекса-источника. Аналогично, достаточными условиями являются: **У6**) наличие $2(m_i+m_j)$+1 путей в орподграфе вида AE либо вида CE, либо вида ACE; **У7**) наличие $2(m_i+m_j+m_{i\to j})$+1 путей в орподграфе любого другого из оставшихся видов.

Орподграф, удовлетворяющий хотя бы одному из достаточных условий У1 – У7 и найденный первым в процессе поиска таких орподграфов, является орподграфом посылки α_j, и он приписывается вершине α_j в процессе поиска достаточных условий межкомплексного ВИС.

Литература

1. Генинсон Б.А., Панкова Л.А., Трантенгерц Э.А. Отказоустойчивые методы обеспечения взаимной информационной согласованности в распределенных вычислительных системах // Автоматика и телемеханика. 1989. N5. С. 3–18.
2. Ашарина И.В., Лобанов А.В., Мищенко И.Г. Взаимное информационное согласование в неполносвязных многомашинных вычислительных системах // Автоматика и телемеханика. 2003. № 5. С. 190-198.
3. Ашарина И.В., Лобанов А.В. Взаимное информационное согласование в неполносвязных гетерогенных многомашинных вычислительных системах // Автоматика и телемеханика. 2010. № 5. С. 133-146.
4. Нефедов В.Н., Осипова В.А. Курс дискретной математики: Учеб. пособие. М.: Изд-во МАИ, 1992. 264 с.

АНАЛИЗ ПРИМЕНИМОСТИ OPENCL-РЕАЛИЗАЦИЙ МЕТОДОВ ГАУССА, ХОЛЕЦКОГО И ЗЕЙДЕЛЯ ДЛЯ РЕШЕНИЯ ЛИНЕЙНОЙ ЗАДАЧИ НАИМЕНЬШИХ КВАДРАТОВ

Д.А. Баранов

Оренбургский госуниверситет

Рассматриваются вопросы реализации методов Гаусса, Холецкого и Зейделя с применением технологии OpenCL для задействования графического процессора. Полученные реализации анализируются на применимость к решению линейной задачи наименьших квадратов. Приводятся результаты экспериментов в виде временных затрат и погрешности вычислений.

Введение

Поиск решения переопределённой системы линейных алгебраических уравнений (СЛАУ)

$$A_{mxn}x = b\,(m>n) \tag{1}$$

называют линейной задачей наименьших квадратов, где A_{mxn} – матрица коэффициентов СЛАУ размерности *m* строк и *n* столбцов, *b* – вектор-столбец свободных членов СЛАУ. Её суть заключается в нахождении псевдорешения $\tilde{x}$ системы (1), минимизирующего невязку $\|Ax-b\|$ [1]:

$$\tilde{x} : \|A\tilde{x}-b\|_2 = min_x \|Ax-b\|_2 \,. \tag{2}$$

Существует несколько подходов к решению линейной задачи наименьших квадратов:

1. Приведение к нормальной системе уравнений.
2. QR-разложение.
3. SVD-разложение.

В данной работе исследуются OpenCL-реализации методов, основанных на первом подходе.

Приведение СЛАУ (1) к системе нормальных уравнений осуществляется умножением слева обеих частей матричного уравнения на матрицу A^T:

$$A^T Ax = A^T b \,. \tag{3}$$

Матрица A^TA является квадратной, симметричной и неотрицательно определённой, следовательно, для решения системы (3) применимы такие методы, как схема Холецкого и итерационный метод Гаусса-Зейделя.

Прямые методы обычно не применяют для поиска решений систем размерностью более 100 [2]. Основными преимуществами итерационных методов на системах большой размерности является меньшая трудоёмкость (по сравнению с прямыми методами) и возможность контролировать точность результата (в то время как в случае применения прямых методов неизбежно появляется погрешность из-за ограниченности разрядной сетки). Тем не менее прямые методы, как правило, обладают большим потенциалом параллелизма. Однако проблема точности остаётся, но в задачах, где точность не так важна, как скорость вычислений (такие задачи возникают, в частности, в графосе-

мантическом моделировании), прямые методы решения СЛАУ, реализованные с применением технологии OpenCL, могут оказаться более приемлемыми.

Сначала отметим одну важную особенность стандарта OpenCL: стандарт допускает погрешность округления при выполнении операции деления до 3ulp (Unit at the Last Place), что означает, что при большом количестве таких операций результат может значительно отличаться от аналогичного, полученного на вычислительном устройстве, реализующем стандарт IEEE 754 для чисел с плавающей запятой одинарной точности [3].

Кроме того, при реализации алгоритма на OpenCL целевой платформой, как правило, подразумевается графический ускоритель. Однако память на современных графических ускорителях – крайне дефицитный ресурс, который, к тому же, нельзя наращивать по своему усмотрению, как оперативную память. Так, на большинстве современных графических ускорителей можно обнаружить маркировку «1Gb VRAM», но программисту доступна лишь часть этой памяти (по стандарту OpenCL – минимум ¼ от общего объёма). Данное ограничение не позволяет, например, одновременную работу с двумя матрицами размерности 5792x5792. Поэтому при выборе алгоритма для реализации с использованием OpenCL очень важным фактором является потребность в дополнительной памяти.

Метод Гаусса

Реализация алгоритма метода Гаусса с использованием OpenCL разделена на четыре функции («ядра» в терминологии OpenCL), по два на прямой и обратный ход. На каждом шаге прямого хода вызывается сначала первое, затем второе ядро. Пусть текущий шаг имеет номер k, размерность СЛАУ – n. В первом ядре прямого хода производится вычисление делителей для каждой строки на текущем шаге:

$$A_{ik}^{k+1} = \frac{A_{ik}^{k}}{A_{kk}^{k}}, i = \overline{k+1, n}. \tag{4}$$

Во втором ядре прямого хода вычисляются новые значения матрицы A и вектора b:

$$\begin{gathered} A_{ij}^{k+1} = A_{ij}^{k} - A_{kj}^{k} A_{ik}^{k}, i = \overline{k+1, n}, j = \overline{k+1, n}, \\ b_{i}^{k+1} = b_{i}^{k} A_{ik}^{k}, i = \overline{k+1, n}. \end{gathered} \tag{5}$$

Заметим, что формула (5) не обнуляет элементы k-го столбца, это необходимо, т.к. все элементы A_{ij}^{k+1} вычисляются параллельно и используют значения k-го столбца. Обнуление этого столбца должно производиться после вычисления всех элементов A_{ij}^{k+1}, однако это не обязательно – алгоритм в дальнейшем просто игнорирует эти элементы (т. е. все элементы ниже главной диагонали), считая их нулевыми.

Рассмотрим обратный ход метода Гаусса. На каждом шаге обратного хода так же вызываются два ядра. В первом ядре обратного хода вычисляется x_k :

$$x_k = \frac{b_k^k}{A_{kk}^k}. \tag{6}$$

Очевидно, для первого шага эта формула работает, а для последующих это становится возможно благодаря второму ядру:

$$b_i^{k+1} = b_i^k - x_k A_{ik}^k, i = \overline{1, k-1}. \tag{7}$$

Как видно из (6), вычисленное значение x_k сразу используется для модификации следующих уравнений. Фактически, происходит перенос компонент $x_k A_{ik}$ в правую

часть уравнений, но в левой части они не обнуляются, а просто игнорируются в дальнейшем. Ядро (5), очевидно, неоптимально, поскольку использует всего один процессор, но его использование более оправданно, чем пересылка из памяти графического процессора в оперативную память и обратно.

Метод Холецкого

Данный метод использует симметричность матрицы *A*, что позволяет сократить необходимое число операций вдвое по сравнению с методом Гаусса. Следует отметить, что в данном методе присутствует *n* операций вычисления корня, которые по стандарту OpenCL [3] имеют ошибку округления 3ulp, т. е. равную ошибке операции деления.

Метод Холецкого можно разделить на 3 этапа:

- вычисление матрицы *U*;
- вычисление вектора *y*;
- вычисление вектора *x*.

Каждый из этапов реализуется с помощью двух ядер OpenCL (как и в методе Гаусса, здесь не используется редукция). Все ядра каждого этапа вызываются последовательно для всех строк матрицы. Для хранения промежуточных значений и результата не требуется дополнительная матрица, однако необходим отдельный вектор *D* размерности *N* для хранения диагональной матрицы знаков [4]. Первое ядро первого этапа вычисляет значение вектора d_i и значение матрицы u_{ii}:

$$\begin{gathered} d_i = sign(a_{ii}), \\ u_{ii} = \sqrt{a_{ii}}\,. \end{gathered} \tag{8}$$

Здесь и далее *i* означает номер итерации. Очевидно, что первое ядро выполняется для одного элемента, но выгода использования аналогична таковой в методе Гаусса. Так же как и в методе Гаусса, первое ядро возвращает правильный результат для a_{11}, но для его корректной работы на последующих элементах необходимо второе ядро:

$$\begin{gathered} a_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} a_{ki} d_k a_{kj}}{a_{ii} d_i}, j = \overline{i+1, n}, \\ a_{jj} = a_{jj} - \left(\frac{a_{ij} - \sum_{k=1}^{i-1} a_{ki} d_k a_{kj}}{a_{ii} d_i} \right)^2 d_i j = \overline{i+1, n}. \end{gathered} \tag{9}$$

Ядра второго и третьего этапов аналогичны обратному ходу методу Гаусса. Как видно из формулы (9), во втором ядре вычисляется сумма (очевидно, в обеих формулах сумма одна, поэтому она может быть вычислена один раз), вычисление которой реализуется через цикл. Фактически, такой подход не является самым оптимальным, но полностью параллельная реализация вычисления данной суммы требует хранения промежуточных результатов и этапа редукции (фактически, метод Map-Reduce). Оптимальная редукция может быть реализована на OpenCL, но для этого необходим анализ оптимальных размеров рабочих групп (и, возможно, их сечение).

Метод Зейделя

В качестве итерационного метода будет рассмотрен метод Зейделя, поскольку система (3) удовлетворяет условию сходимости этого метода, а малое количество опера-

ций деления (и отсутствие извлечений корня) позволяет свести погрешность, вносимую OpenCL, до минимума.

Поскольку в методе Зейделя (как и в любом итерационном методе) на каждой итерации в вычислениях используется исходная матрица и правая часть (в данном случае $A^T A$ и $A^T b$), для хранения промежуточного результата необходима дополнительная память. Представленная ниже реализация использует дополнительный вектор T размерности N+1 (на 1 больше, чем в методе Холецкого) для хранения значений

$$t_i = \sum_{j=1}^{i-1} a_{ij} x_j^k + \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} . \qquad (10)$$

Метод Зейделя реализуется через 2 ядра OpenCL, вызываемых последовательно на каждой итерации. На k-ой итерации первое ядро сохраняет предыдущее значение $x_i^{(k-1)}$ в последний элемент вспомогательного вектора T и вычисляет значение x_i^k :

$$t_n = x_i^{(k-1)} , \quad x_i^k = \frac{t_i}{a_{ii}} . \qquad (11)$$

Во втором ядре производится обновление значений элементов вектора T:

$$t_j = t_j + t_n a_{ji} - x_i a_{ji} , j = \overline{1,n}, j \neq i . \qquad (12)$$

Таким образом, использование t_n для хранения значения $x_i^{(k-1)}$ позволяет поддерживать вектор T в актуальном состоянии.

Следует отметить, что матрица $A^T A$ обусловлена хуже A [1]:

$$cond A_2 \left(A^T A\right) = cond A_2^2 \left(A\right). \qquad (13)$$

Как следствие, при подстановке решения, полученного методом Зейделя, в систему (1) ошибка будет больше заданной точности.

На рис. 1 приведён график зависимости времени вычислений описанной реализации метода Зейделя от размерности матрицы $A^T A$, а также результаты для однопоточной CPU-версии.

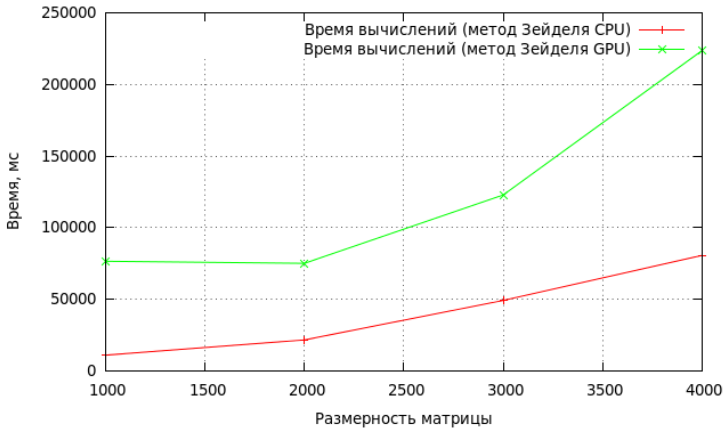


Рис. 1. Зависимость времени вычислений от размерности СЛАУ для метода Зейделя (CPU и GPU)

Как видно из рис. 1, последовательная версия работает быстрее параллельной. Это объясняется иерархической организацией памяти. Иерархия включает «частную» память, локальную и глобальную. Эта архитектура унаследована от графических процессоров, поэтому следует учитывать их особенности. В частности, доступ к глобальной

памяти является медленной операцией, поэтому реализация алгоритмов с большой долей операций с глобальной памятью на OpenCL может дать отрицательный результат, выражающийся в падении производительности по сравнению с последовательной версией.

Сопоставление

На рис. 2 приведены графики зависимости времени вычислений рассмотренных выше реализаций методов решения СЛАУ.

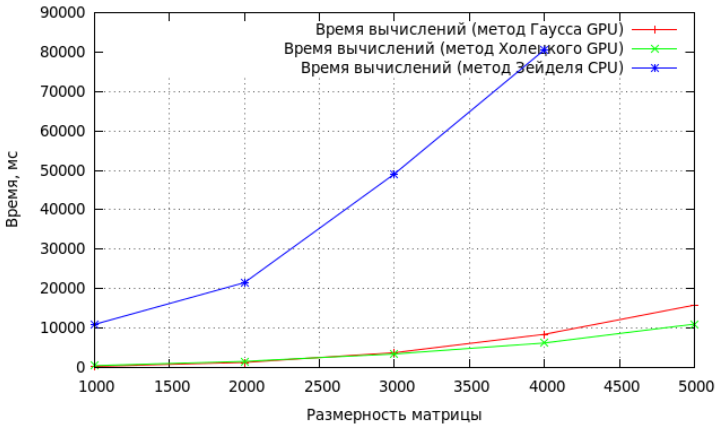


Рис. 2. Зависимость времени вычислений от размерности СЛАУ для методов Гаусса, Холецкого и Зейделя

В табл. 1 приведены значения невязки (2) для каждой из рассмотренных реализаций на каждой из тестовых матриц.

Таблица 1. Невязки (2) для методов Гаусса, Холецкого и Зейделя

N	Метод		
	Гаусса	Холецкого	Зейделя (eps=0.001)
1000	0,000740	0,00070	0,06330
2000	0,001543	0,00170	0,06570
3000	0,020710	0,01990	0,06970
4000	0,012975	0,06520	0,09250
5000	0,037140	0,13590	0,11340

В результате анализа рис. 2 и табл. 1 можно сделать вывод, что реализация метода Гаусса на OpenCL является наиболее оптимальной в задачах, где скорость важнее точности, т. к. она незначительно уступает реализации метода Холецкого на матрицах размерности >3000 и при этом имеет меньшую погрешность. В то же время методу Зейделя требуется на порядок больше времени для достижения сравнимой точности.

Исследование выполнялось при финансовой поддержке Российского гуманитарного научного фонда (проект № 12-04-12034в) и Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (контракт N 14.B37.21.0176).

Литература

1. Белов С.А., Золотых Н.Ю. Лабораторный практикум по численным методам линейной алгебры. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2005. – 235 с.
2. Самарский А.А., Гулин А.В. Численные методы. – М.: Наука, 1989. – 432 с.
3. Khronos OpenCL Working Group The OpenCL Specification 1.1., 2011. – [http://www.khronos.org/registry/cl].
4. Богачёв К.Ю. Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений. – М.: Изд-во МГУ, 1998. – 137 с.

ИСПОЛЬЗОВАНИЕ ЗНАЧЕНИЙ ПРОИЗВОДНЫХ В ПАРАЛЛЕЛЬНЫХ АЛГОРИТМАХ УСЛОВНОЙ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

К.А. Баркалов

Нижегородский госуниверситет им. Н.И. Лобачевского

Предложен алгоритм решения многоэкстремальных задач с невыпуклыми ограничениями, использующий значения первых производных функций задачи. Предполагается, что производные удовлетворяют условию Липшица. Для редукции условной задачи к безусловной используется индексная схема, основанная на раздельном учете каждого ограничения задачи; идеи метода штрафных функций не используются. Проведены численные эксперименты, подтверждающие сходимость метода и его эффективность по сравнению с алгоритмами без учета значений производных.

Введение

Оригинальный подход к минимизации многоэкстремальных функций при невыпуклых ограничениях – индексный метод – основан на раздельном учете каждого ограничения задачи и не связан с использованием штрафных функций [1, 2]. В соответствии с правилами индексного метода каждая итерация включает последовательную проверку выполнимости ограничений задачи в этой точке. При этом обнаружение первого нарушенного ограничения прерывает испытание и инициирует переход к точке следующей итерации.

Алгоритмы из работ [1, 2] основаны на предположении о липшицевости всех функционалов задачи, что типично и для многих других подходов (см., например, [3, 4]). Но если для функций, входящих в постановку задачи оптимизации, известны значения производных, их использование может существенно ускорить сходимость. Так, в [5-7] предложены эффективные алгоритмы, использующие значения производной целевой функции в задачах глобальной оптимизации без ограничений. В то же время остается открытым вопрос использования значений производных в алгоритмах условной глобальной оптимизации.

1. Постановка задачи

Рассматривается одномерная задача глобальной оптимизации

$$\varphi(x^*) = \min\{\varphi(x) : x \in [a,b],\ g_j(x) \le 0,\ 1 \le j \le m\}, \tag{1}$$

когда целевая функция $\varphi(x)$ (в дальнейшем обозначаемая также g_{m+1}) и левые части ограничений $g_j(x)$, $1 \le j \le m$, определены и вычислимы лишь в соответствующих подобластях $Q_j \subset [a,b]$, где

$$Q_1 = [a,b],\ Q_{j+1} = \{x \in Q_j : g_j(x) \le 0\},\ 1 \le j \le m. \tag{2}$$

С учетом условий (2), исходная задача (1) может быть представлена в виде

$$\varphi(x^*) = \min\{g_{m+1}(x) : x \in Q_{m+1}\}.$$

Вводится классификация точек x из области поиска $[a,b]$ с помощью *индекса* $\nu = \nu(x)$. Указанный индекс ν определяется условиями

$$g_j(x) \le 0,\ 1 \le j \le \nu - 1,\ g_\nu(x) > 0,$$

где последнее неравенство несущественно, если $\nu = m+1$, и удовлетворяет неравенствам

$$1 \le \nu = \nu(x) \le m+1.$$

Данная классификация порождает «индексную» функцию

$$f(x) = g_\nu(x),\ \nu = \nu(x),$$

определенную и вычислимую всюду на $[a,b]$. Ее значение в точке x есть либо значение левой части ограничения, нарушенного в этой точке (случай, когда $\nu \le m$), либо значение минимизируемой функции (случай, когда $\nu = m+1$).

Основная идея индексного подхода состоит в редукции условной задачи (1) к безусловной задаче

$$\psi(x^*) = \min\{\psi(x):\ x \in [a,b]\},$$

где

$$\psi(x) = \begin{cases} g_\nu(x), & \nu < m, \\ g_m - g_m^*, & \nu = m. \end{cases}$$

Как результат, функция $\psi(x)$ будет дифференцируемой на каждом множестве Q_ν, $1 \le \nu \le m$. Однако эта новая функция будет иметь разрывы первого рода на граничных точках множеств Q_ν из (2), поэтому применять для ее минимизации известные алгоритмы [6-8] непосредственно нельзя. Требуются дополнить алгоритм новыми правилами учета разрывов функции $\psi(x)$.

2. Оценка минимума миноранты для «индексной» функции

Основой рассматриваемого в работе алгоритма является предположение о липшицевости первых производных функций задачи, т.е. выполнение условия

$$\left|g_j'(x_1) - g_j'(x_2)\right| \le L_j \left|x_1 - x_2\right|,\ 1 \le j \le m+1,\ [x_1, x_2] \subset [a,b].$$

В работе [7] приведен вывод нижней оценки значения минимизируемой функции на отрезке $[x_1, x_2]$ в задачах безусловной оптимизации (т.е. при m=0, j=1)

$$g_j(x) \ge g_j(x_1) + g_j'(x_1)(\hat{x} - x_1) - 0.5L_j(\hat{x} - x_1)^2,$$

где

$$\hat{x} = \frac{-(g_j(x_2) - g_j(x_1)) + (g_j'(x_2)x_2 - g_j'(x_1)x_1) + 0.5L_j(x_2^2 - x_1^2)}{L_j(x_2 - x_1) + (g_j'(x_2) - g_j'(x_1))}.$$

Указанная оценка может быть использована и для задач с ограничениями, если $j = \nu(x_1) = \nu(x_2)$, т.е. когда на границах отрезка нарушено одно и то же ограничение, или же точки x_1, x_2 являются допустимыми. В случае если $\nu(x_1) \ne \nu(x_2)$, то в качестве нижней оценки можно использовать либо неравенство

$$g_j(x) \ge g_j(x_1) + g_j'(x_1)(x_2 - x_1) - 0.5L_j(x_2 - x_1)^2,\ j = \nu(x_1) > \nu(x_2)$$

(т.е. когда у функции вычислено значение и производная на левой границе отрезка), либо неравенство

$$g_j(x) \ge g_j(x_2) - g_j'(x_2)(x_2 - x_1) - 0.5L_j(x_2 - x_1)^2,\ j = \nu(x_2) > \nu(x_1)$$

(т.е. когда у функции вычислены значение и производная на правой границе отрезка).

Полученные нижние оценки могут быть использованы в алгоритме в качестве характеристик интервалов поиска: очередное испытание нужно проводить в интервале, которому соответствует наименьшее значение миноранты, т.е. в интервале с минимальной характеристикой.

3. Последовательный и параллельный алгоритмы

Первое испытание осуществляется в произвольной внутренней точке $x^1 \in (a,b)$. Выбор точки x^{k+1}, $k \geq 1$, любого последующего испытания определяется следующими правилами.

Правило 1. Перенумеровать точки $x^1, \ldots, x^k$ предшествующих испытаний нижними индексами в порядке увеличения значений координаты, т.е.

$$a = x_0 < x_1 < \ldots < x_i < \ldots < x_k < x_{k+1} = b, \tag{3}$$

и сопоставить им значения $z_i = g_\nu(x_i)$ и $z'_i = g'_\nu(x_i)$, $\nu = \nu(x_i)$, $1 \leq i \leq k$, вычисленные в этих точках; точки $x_0 = a$ и $x_{k+1} = b$ введены дополнительно (значения z_0, z_{k+1}, z'_0, z'_{k+1} не определены) для удобства последующих обозначений.

Правило 2. Провести классификацию номеров $i, 1 \leq i \leq k$, точек из ряда (3) по числу ограничений задачи, выполняющихся в этих точках, путем построения множеств

$$I_\nu = \{ i:\ 1 \leq i \leq k,\ \nu = \nu(x_i) \},\ 1 \leq \nu \leq m+1,$$

содержащих номера всех точек $x_i, 1 \leq i \leq k$, имеющих индексы, равные одному и тому же значению ν. Граничные точки $x_0 = a$ и $x_{k+1} = b$ интерпретируются как имеющие нулевые индексы, и им сопоставляется дополнительное множество $I_0 = \{0, k+1\}$.

Определить максимальное значение индекса

$$M = \max\{ \nu = \nu(x_i), 1 \leq i \leq k \}.$$

Правило 3. Вычислить текущие нижние границы для неизвестных констант Липшица L_ν первой производной функций $g_\nu, 1 \leq \nu \leq m+1$

$$\mu_\nu = \max\left\{ \max\left\{ \begin{array}{c} \left| z'_i - z'_j \right| / (x_i - x_j), \\ 2\left[-(z_i - z_j) + z'_j (x_i - x_j) \right] / (x_i - x_j)^2, \\ 2\left[(z_i - z_j) - z'_i (x_i - x_j) \right] / (x_i - x_j)^2, \end{array} \right\} \quad i, j \in I_\nu,\ i > j \right\} \tag{4}$$

Если множество I_ν содержит менее двух элементов или если μ_ν из (4) оказывается равным нулю, то принять $\mu_\nu = 1$. Из (4) следует, что оценки μ_ν являются неубывающими, начиная с момента, когда (4) порождает первое положительное значение μ_ν.

Правило 4. Для всех непустых множеств I_ν, $1 \leq \nu \leq m+1$, вычислить оценки

$$z^*_\nu = \begin{cases} 0, & \nu < M, \\ \min\{ g_\nu(x_i) : i \in I_\nu \}, & \nu = M. \end{cases}$$

Правило 5. Для каждого интервала (x_{i-1}, x_i), $1 \leq i \leq k+1$, вычислить *характеристику* $R(i)$, при этом если $\nu = \nu(x_{i-1}) = \nu(x_i)$, то

$$R(i) = (z_{i-1} - z^*_\nu) + z'_{i-1}(\hat{x}_i - x_{i-1}) - 0.5 r_\nu \mu_\nu (\hat{x}_i - x_{i-1})^2,$$

где

$$\hat{x}_i = \frac{-(z_i - z_{i-1}) + (z'_i x_i - z'_{i-1} x_{i-1}) + 0.5 r_\nu \mu_\nu (x_i^2 - x_{i-1}^2)}{r_\nu \mu_\nu (x_i - x_{i-1}) + (z'_i - z'_{i-1})},$$

если $\nu = \nu(x_i) > \nu(x_{i-1})$, то

$$R(i) = (z_i - z^*_\nu) - z'_i (x_i - x_{i-1}) - 0.5 r_\nu \mu_\nu (x_i - x_{i-1})^2,$$

если $\nu = \nu(x_{i-1}) > \nu(x_i)$, то

$$R(i) = (z_{i-1} - z^*_\nu) + z'_{i-1}(x_i - x_{i-1}) - 0.5 r_\nu \mu_\nu (x_i - x_{i-1})^2.$$

Величины $r_\nu > 1$, $1 \le \nu \le m+1$, являются параметрами алгоритма. Подходящий выбор значений r_ν позволяет использовать произведение $r_\nu \mu_\nu$ как оценку констант Липшица L_ν, $1 \le \nu \le m+1$, для первых производных.

Правило 6. Определить интервал (x_{t-1}, x_t), которому соответствует минимальная характеристика

$$R(t) = \min\{R(i):\ 1 \le i \le k+1\}. \tag{5}$$

Правило 7. Провести очередное испытание в серединной точке интервала (x_{t-1}, x_t), если индексы его концевых точек не совпадают, т.е.

$$x^{k+1} = \frac{x_t + x_{t-1}}{2},\ \ \nu(x_{t-1}) \ne \nu(x_t).$$

В противном случае провести испытание в точке

$$x^{k+1} = \hat{x}_t = \frac{-(z_t - z_{t-1}) + (z'_t x_t - z'_{t-1} x_{t-1}) + 0.5 r_\nu \mu_\nu (x_t^2 - x_{t-1}^2)}{r_\nu \mu_\nu (x_t - x_{t-1}) + (z'_t - z'_{t-1})},$$

$$\nu = \nu(x_{t-1}) = \nu(x_t).$$

Описанные правила можно дополнить условием остановки, прекращающим испытания, если

$$x_t - x_{t-1} \le \varepsilon,$$

где t из (5) и $\varepsilon > 0$ есть заданная точность.

Предложенный алгоритм относится к классу характеристических алгоритмов, поэтому его распараллеливание возможно в рамках синхронной и асинхронной схем, представленных в [8]. При этом алгоритм распараллеливается на единой области оптимизации, что позволяет учитывать результаты испытаний, полученные всеми процессорами.

Проведены численные эксперименты, подтверждающие сходимость метода и его эффективность по сравнению с алгоритмами без учета значений производных. Примененная процедура оценки основана на численном решении сравниваемыми методами всех задач из некоторой случайно генерируемой выборки большого объема и использует аппарат *операционных характеристик, описанный, например, в* [9].

В заключение отметим, что в работе предложен способ использования производных в задачах условной глобальной оптимизации, решение которых осуществляется с помощью индексной схемы. Предложенный алгоритм существенно ускоряет процесс решения задач условной оптимизации. Данное утверждение установлено путем численного решения набора тестовых задач из различных классов.

Работа выполнена при поддержке совета по грантам Президента Российской Федерации (грант № НШ-1960.2012.9), РФФИ (грант № 11-01-00682-а), а также ФЦП «Научные и научно-педагогические кадры инновационной России» (соглашение № 14.B37.21.0878).

Литература

1. Стронгин Р.Г., Баркалов К.А. О сходимости индексного алгоритма в задачах условной оптимизации с ε-резервированными решениями // Математические вопросы кибернетики. М.: Наука, 1999. С. 273–288.
2. Баркалов К.А., Стронгин Р.Г. Метод глобальной оптимизации с адаптивным порядком проверки ограничений // Ж. вычисл. матем. и матем. физ. 2002. Т.42, №9. С. 1338–1350.
3. Евтушенко Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации. М.: Наука, 1982.

4. Pinter J. Global optimization in action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications). Kluwer Acad. Publ., 1996.
5. Гергель В.П. Об одном способе учета значений производных при минимизации многоэкстремальных функций // Ж. вычисл. матем. и матем. физ. 1996. Т.36, №6. С. 51–67.
6. Sergeyev Ya.D. A global optimization algorithm using derivatives and local tuning. ISI-CNR, Report 1, 1994.
7. Gergel V.P., Sergeyev Ya.D. Sequential and parallel algorithms for global minimizing functions with lipschitzian derivatives// Computers & Mathematics with Applications. 1999. V. 37 № 4-5. P. 163–179.
8. Grishagin V.A., Sergeyev Y.D., Strongin R.G. Parallel characteristical algorithms for solving problems of global optimization // Journal of Global Optimization. 1997. V.10. № 2. P. 185–206.
9. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации – М.: ФИЗМАТЛИТ, 2008.

ОПТИМИЗАЦИЯ ПАРАМЕТРОВ SVM-РЕГРЕССИИ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА ГЛОБАЛЬНОГО ПОИСКА

К.А. Баркалов, И.Б. Мееров, А.Н. Половинкин, С.В. Сидоров, Н.Ю. Золотых

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача выбора оптимальных параметров для метода построения регрессии с использованием алгоритма опорных векторов. Предлагается подход, основанный на оптимизации функции ошибки перекрестного контроля с использованием алгоритма глобального поиска для решения задачи глобальной оптимизации. Приводятся результаты вычислительного эксперимента.

Введение

Одним из распространенных методов для решения задачи восстановления регресии является алгоритм SVM-регрессии [1]. Среди преимуществ данного алгоритма можно выделить возможность эффективного моделирования нелинейных зависимостей, а также независимость его обобщающей способности от размерности пространства признаков. Однако в отдельных случаях практическое применение алгоритма ограничено вследствие того, что точность метода сильно зависит от выбора его параметров [3]. Наиболее часто применяемые на практике подходы для выбора оптимальных параметров в итоге сводятся к решению задачи глобальной оптимизации [2, 3]. В данной работе предлагается новый метод, основанный на оптимизации функции ошибки перекрестного контроля с использованием основанного на информационно-статистическом подходе алгоритма глобального поиска.

1. Оптимизация параметров алгоритма SVM-регресии

В работе рассматривается задача восстановления регрессии в следующей постановке. Пусть задана обучающая выборка $\{(x_i, y_i), i = 1,..., N\}$, где $x_i \in R^d$ – вектор признаков, $y_i \in R$ – ответ. Требуется найти функцию $f(x)$, принадлежащую некоторому определенному классу K, минимизирующую значение эмпирического риска (ошибка предсказания на обучающей выборке). Для алгоритма построения SVM-регрессии функцию $f(x)$ в общем виде можно записать как

$$f(x) = w^T \phi(x) + b,$$

где $\phi(x)$ – нелинейное (в общем случае) отображение $R^d \to R^m$, $w \in R^m$ – вектор коэффициентов линейной функции в новом пространстве признаков R^m. В качестве функции потерь используется кусочно-линейная функция ε-чувствительности:

$$L_\varepsilon(y, f(x)) = \max(0, |y - f(x)| - \varepsilon),$$

где ε – заранее заданный порог (в случае если предсказанное значение отличается от истинного на величину, меньшую данного порога, ошибка считается равной нулю). Функция эмпирического риска записывается в виде:

$$R_{emp}(w) = \frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(y_i, f(x_i)).$$

Рассматриваемая задача минимизации эмпирического риска сводится к задаче квадратичной оптимизации

$$\min_{w} \frac{\|w\|^2}{2} + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right), \qquad (1)$$
$$f(x_i) - y_i \le \varepsilon + \xi_i,\ y_i - f(x_i) \le \varepsilon + \xi_i^*,\ \xi_i, \xi_i^* \ge 0,$$

где C – параметр регуляризации, который соответствует отношению сложности модели и эмпирической ошибки в минимизируемой функции. Используя метод множителей Лагранжа, задачу (1) можно свести к двойственной форме:

$$\max \sum_{i=1}^{N} y_i\left(\alpha_i^* - \alpha_i\right) - \varepsilon\sum_{i=1}^{N}\left(\alpha_i^* + \alpha_i\right) - \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right)K(x_i, x_j),$$
$$\sum_{i=1}^{N}\left(\alpha_i^* - \alpha_i\right) = 0,\ 0 \le \alpha_i, \alpha_i^* \le C, \qquad (2)$$

где α_i, α_i^* – множители Лагранжа, $K(x_i, x_j)$ – функция ядра, которая соответствует скалярному произведению в новом пространстве признаков R^m. Одними из наиболее часто применяемых на практике ядер являются радиальные базисные функции:

$$K\left(x_i, x_j\right) = \exp\left(-\left\|x_i - x_j\right\|^2 \Big/ 2\sigma^2\right).$$

Как показано в [3], обобщающая способность алгоритма SVM-регрессии существенным образом зависит от выбора параметров C, ε и σ. В работе [2] предлагается метод, основанный на минимизации ошибки перекрестного контроля. Идея метода заключается в разделении тренировочной выборки случайным образом на S частей $\{G_s, s = 1,...,S\}$, обучении модели на $(S-1)$ части и использовании оставшейся части для вычисления тестовой ошибки. Усредненная ошибка по всем тестовым множествам используется в качестве оценки обобщающей способности алгоритма

$$MSE_{CV} = \frac{1}{N}\sum_{s=1}^{S}\sum_{i \in G_s}\left(y_i - f(x_i \mid \theta_s\right)^2,$$

где θ_s – решение задачи (2), полученное при использовании в качестве тренировочной выборки множества $D \setminus \theta_s$. В случае когда число объектов в обучающей выборке невелико, можно использовать LOO (leave-one-out) ошибку

$$MSE_{LOO} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i \mid \theta_i\right)^2,$$

где θ_i – решение задачи (2), полученное при использовании в качестве тренировочной выборки множества $D \setminus \{(x_i, y_i)\}$. В силу того, что решение задачи квадратичного программирования (2) для каждого набора параметров (C, ε, σ) существует и единственно, для заданной тренировочной выборки можно рассмотреть leave-one-out-ошибку MSE_{LOO} как фунцию, зависящую от C, ε и σ:

$$MSE_{LOO} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i \mid \theta_i(C, \varepsilon, \sigma))\right)^2 = F(C, \varepsilon, \sigma). \qquad (3)$$

Таким образом, задача выбора оптимальных параметров алгоритма построения SVM-регрессии свелась к задаче минимизации функции $F(C, \varepsilon, \sigma)$. В общем случае функция является многоэкстремальной, что требует применения алгоритмов глобальной оптимизации для нахождения её оптимума.

2. Параллельный алгоритм глобального поиска

Рассматриваемый алгоритм основан на подходе к решению задач условной глобальной оптимизации, использующем информационно-статистическую теорию глобального поиска [4, 5, 7].

Рассмотрим задачу безусловной глобальной оптимизации вида

$$\varphi = \varphi(y) = \min\{\varphi(y) : y \in D\},$$
$$D = \{y \in R^N : a_i \le y_i \le b_i,\ 1 \le i \le N\}, \tag{4}$$

где целевая функция $\varphi(y)$ удовлетворяет условию Липшица с соответствующей константой L (которая в общем случае может быть не задана). В обсуждаемом подходе (см. [4, 6]) решение многомерных задач сводится к решению эквивалентных им одномерных задач (редукция размерности). Идея метода заключается в том, чтобы, используя кривые типа развертки Пеано $y(x)$, однозначно отображающие отрезок $[0,1]$ на N-мерный гиперкуб D,

$$D = \{y \in R^N : -2^{-1} \le y_i \le 2^{-1},\ 1 \le i \le N\} = \{y(x):\ 0 \le x \le 1\},$$

свести исходную задачу к следующей одномерной задаче:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\}. \tag{5}$$

Таким образом, рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией и липшицевыми ограничениями одномерную задачу, в которой соответствующие функции удовлетворяют равномерному условию Гельдера.

Недостатком подобного подхода является потеря части информации о близости точек в многомерном пространстве. Это объясняется тем, что точка $x \in [0,1]$ имеет лишь левых и правых соседей, а соответствующая ей точка $y(x) \in R^N$ имеет соседей по 2^N направлениям. Сохранить часть информации о близости точек позволяет использование множества отображений

$$Y_L(x) = \{y^1(x), ..., y^L(x)\} \tag{6}$$

вместо применения единственной кривой Пеано $y(x)$. Каждая кривая Пеано $y^i(x)$ из $Y_L(x)$ может быть получена в результате некоторого сдвига вдоль главной диагонали гиперинтервала D [5] или поворота [8].

Использование множественных отображений (6) позволяет решать задачу (4) путем параллельного решения L задач вида (5) на наборе отрезков $[0,1]$. Каждая одномерная задача решается на отдельном процессоре с использованием развертки $y^s, 1 \le s \le L$. Результаты испытания в точке x^k, полученные конкретным процессором для решаемой им задачи, интерпретируются как результаты испытаний во всех остальных задачах (в соответствующих точках $x^{k1}, ..., x^{kL}$) и рассылаются другим процессорам. При таком подходе испытание в точке $x^k \in [0,1]$, осуществляемое в s-й задаче, состоит в последовательности действий:

1. Определить образ $y^k = y^s(x^k)$ при соответствии $y^s(x)$;
2. Проинформировать остальные процессоры о начале проведения испытания в точке y^k (*блокирование* точки y^k);
3. Вычислить величину $\varphi(y)$. Пара $\{y^s(x^k), z^k = \varphi(y^s(x^k))\}$ является результатом испытания в точке x^k;

4. Определить прообразы $x^{kl} \in [0,1]$, $1 \le l \le L$, точки y^k и интерпретировать испытание, проведенное в точке $y^k \in D$, как проведение испытаний в L точках $x^{k1},...,x^{kL}$ с одинаковыми результатами $\varphi(y^1(x^{k1}) = ... = \varphi(y^L(x^{kL})) = z^k$.
5. Проинформировать остальные процессоры о результатах испытания в точке y^k, разослав им пары (y^k, z^k).

Каждый процессор имеет свою копию программных средств, реализующих вычисление функций задачи, и решающее правило алгоритма. Для организации взаимодействия на каждом процессоре создается очередь, в которую процессоры помещают информацию о выполненных итерациях в виде пар: точка очередной итерации и значение из (4).

Различные варианты алгоритма глобального поиска для решения одномерных и многомерных задач и соответствующая теория сходимости представлены в работах [4-8].

3. Результаты вычислительного эксперимента

Рассмотрим функцию $y(x) = 0.5 + 0.4\sin(2\pi x)\cos(6\pi x)$. Пусть $x_i = 0.05 \cdot (i-1)$, $i = 1,...,21$, $y_i = y(x_i) + N(0,0.05)$, где $N(0,0.05)$ – гауссово распределение с математическим ожиданием, равным 0, и стандартным отклонением, равным 0.05. На рис. 1 приведен график зависимости ошибки перекрестного контроля MSE_{LOO} от ε и σ при фиксированном значении параметра C =1. Как видно из графика, функция, описывающая данную зависимость, содержит несколько локальных минимумов в области поиска.

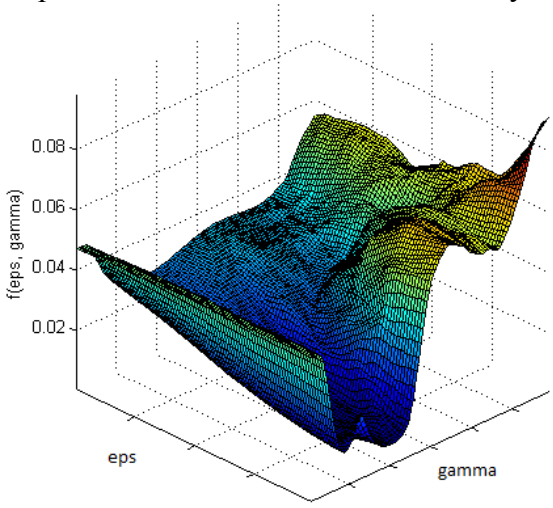


Рис. 1. График зависимости ошибки перекрестного контроля от параметров

Для каждой задачи использовалось правило остановки $\|y - y^*\| \le \rho$, т.е. достижение известного глобального оптимума с заданной точностью ρ в евклидовой метрике.

При решении использовались следующие параметры алгоритма: точность поиска $\rho = 0.01$, параметр надежности $r = 2.5$, точность развертки $M = 12$, количество разверток $L = 2$. Общее число итераций, выполненных параллельной версией алгоритма до достижения критерия останова, равно 1550 (для достижения той же точности методом полного перебора потребовалось бы 10000 итераций), найденное значение оптимума равно 0.006728.

4. Заключение

В работе предложен новый метод для нахождения оптимальных параметров SVM-регрессии, основанный на оптимизации функции ошибки перекрестного контроля с ис-

пользованием основанного на информационно-статистическом подходе алгоритма глобального поиска. Проведенный вычислительный эксперимент показал преимущество подхода над методом полного перебора, обычно используемым на практике. Среди возможных продолжений работы наиболее перспективным представляется расширение метода для нахождения оптимальных параметров алгоритмов обучения с учителем, предназначенных для решения задач классификации.

Работа выполнена в рамках программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы», государственный контракт № 11.519.11.4015.

Литература

1. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. – Springer, 2008.
2. Ito K., Nakano R. Optimization Support Vector Regression Hyperparameters Based on Cross-Validation // Proceedings of the International Joint Conference on Neural Networks. 2003. Vol. 3. P. 2077–2083.
3. Ren Yu., Bai G. Determination of Optimal SVM Parameters by Using GA/PSO // Journal of Computers. 2010. Vol. 5. No 8, P. 1160-1168.
4. Стронгин Р.Г. Численные методы в многоэкстремальных задачах (Информационно-статистические алгоритмы). М.: Наука, 1978.
5. Стронгин Р.Г. Параллельная многоэкстремальная оптимизация с использовани-ем множества разверток // Ж. вычисл. матем. и матем. физ. 1991. Т.31, №8. С. 1173 – 1185.
6. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000.
7. Gergel V.P., Strongin R.G. Parallel computing for globally optimal decision making on cluster systems // Future Generation Computer Systems. 2005. Vol. 21. № 5. P. 673-678.
8. Баркалов К.А., Сидоров С.В., Рябов В.В. Параллельные вычисления в задачах многоэкстремальной оптимизации // Вестник ННГУ. 2009. №6(1). С. 171-177.

АДАПТАЦИЯ НЕКОТОРЫХ РЕШАТЕЛЕЙ СЛАУ ДЛЯ АРИФМЕТИЧЕСКИХ УСКОРИТЕЛЕЙ NVIDIA

Ю.Г. Бартенев, А.П. Карпов, И.А. Крючков, И.А. Пищулин

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Решению СЛАУ на GPGPU (General Purpose Graphical Processor Unit) уделяется большое внимание ([1-4] и др.). Известно, что эффективная реализация для GPGPU ряда актуальных алгоритмов решателей затруднительна. Это относится к решателям для разреженных матриц, особенно использующим этапы решения треугольных линейных систем (Гаусс-Зейдель, ILU [5]).

Выбор решателя СЛАУ, учет в решателе специфики свойств СЛАУ, потока СЛАУ в приложении позволяет лучше воспользоваться высокой производительностью GPGPU. Авторами разработаны, усовершенствованы программы решателей для NVIDIA GPGPU – наискорейшие (или близкие к ним) на одном ядре универсального микропроцессора в ряде 3D-приложений. Для ЭВМ с одним GPGPU и одним ядром микропроцессора реализованы:

1) предобусловливатель ADI [6] (метод переменных направлений) и решатель BiCGSTAB [5] для регулярно разреженных матриц СЛАУ задач теплопроводности;
2) предобусловливатель Якоби [5] (обратная точечная или мелкоблочная диагональ) и BiCGSTAB для нерегулярно разреженных матриц простых СЛАУ задач гидродинамики;
3) предобусловливатель ILU0/ILDU0 [5] и BiCGSTAB для мелкоблочных нерегулярно разреженных матриц СЛАУ задач аэродинамики;
4) возможность сочетания с BiCGSTAB (помимо CG [5]), оптимизирован предобусловливатель AMG библиотеки Parallel Toolbox [7] для потока СЛАУ давления задач гидродинамики с нерегулярно разреженными матрицами.

Реализация выполнена на языке CUDA [8]. Тестирование проведено на СЛАУ размером 10^6, выработанных программами [9-10]. Собственно BiCGSTAB выполняется на Nvidia Tesla C2050® ~ в 15 раз быстрее, чем на ядре Intel Westmere® с частотой 2,93 ГГц.

В методе 1 предобусловливатель представляется произведением нескольких трехдиагональных матриц, производных от исходной матрицы, путем естественного переупорядочивания, количество которых не больше размерности сетки; используются форматы DIA, ELL [5] матриц; решение СЛАУ на GPGPU ускоряется в 7-10 раз.

В методе 2 используется формат CSR [5] исходной матрицы и плотный формат для предобусловливающей обратной блочной диагонали; решение СЛАУ на GPGPU ускоряется в 10 раз.

В методе 3 используется формат CSR матриц; выполняется предварительное предвычисление этапов конвейера решения нижней и верхней треугольной системы с возможностью использования результата этой процедуры для решения треугольных СЛАУ в предобусловливателе на итерациях BiCGSTAB (и следующей СЛАУ при совпадении портретов матриц); решение СЛАУ на GPGPU ускоряется в 3-6 раз.

В методе 4 реализован более быстрый алгоритм $A_c = P^T A_f P$ вычисления грубых матриц в расчете на известное расположение ненулевых коэффициентов в них, которое определяется при решении первой в потоке СЛАУ. Сохраненная пирамида матриц про-

лонгации (P) и указанный алгоритм используются для решения следующей СЛАУ, обладающей матрицей с таким же портретом. Это резко сокращает среднее время этапа построения предобусловливателя AMG, выполняемого на ядре универсального процессора; решение СЛАУ на GPGPU ускоряется в 3-5 раз.

Проведенные эксперименты показывают:

- более быстрое решение СЛАУ пакета ЛОГОС в последовательном режиме с использованием GPGPU при включении данных разработок в LParSol [11] возможно;
- реализация параллельного режима будет менее эффективной из-за выполнения обмена информацией через универсальный микропроцессор;
- скорость решения СЛАУ с разреженными матрицами на TESLA C2050 сдерживают скорость доступа к оперативной памяти и механизма управления потоками.

Перевод решателей на OpenCL [12] позволит испытать GPGPU AMD и MIC [13] Intel.

Авторы благодарны А.В. Ерзунову, Е.Б. Щаниковой за консультации по решателям.

Литература

1. Демидов Д.Е., Егоров А.Г., Нуриев А.Н. Использование технологии NVIDIA CUDA для решения задач гидромеханики // XI Международный семинар «Супервычисления и математическое моделирование»: Тез. докл. Саров, 2009. С. 58-59.
2. Naumov M. Parallel Incomplete-LU and Cholesky Factorization in the Preconditioned Iterative Methods on the GPU: NVIDIA Technical Report NVR-2012-003. – NVIDIA, 2012.
3. Dalton S., Olson L., Bell N. Efficient Algebraic Multigrid Methods on GPUs // Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods: NVIDIA Technical Report NVR-2011-002. Illinois, 2011.
4. Березин С.Б., Каргапольцев И.С., Марковский Н.Д., Сахарных Н.А. Па-раллельная реализация метода расщепления для системы из нескольких GPU с применением в задачах аэрогидродинамики // Межд. научн. конф. «Параллельные вычислительные технологии 2012»: Сб. тр. Новосибирск, 2012. С. 54–64.
5. Saad Y. Iterative Methods for Sparse linear Systems. Boston: International Thomson Publishing Company, 1996.
6. Meurant G. Computer Solution of Large Linear Systems. Amsterdam: Elsevier Science B.V., 1999.
7. Liebmann M. Algebraic Multigrid Methods on GPU-Accelerated Hybrid Architectures. Institute for Mathematics and Scientific Computing, University of Graz, 2011.
8. NVIDIA CUDA Programming Guide. Ver 4.0. NVIDIA Corporation, 2011.
9. Бабанов А.В., Бельков С.А., Бондаренко С.В. и др. Методика МИМИЗА-НДЗД. Расчет трехмерных задач спектрального переноса излучения // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2012. Вып. 2. С. 64-72.
10. Козелков А.С., Дерюгин Ю.Н., Зеленский Д.К., Глазунов В.А., Голубев А.А., Денисова О.В., Лашкин С.В., Жучков Р.Н., Тарасова Н.В., Сизова М.А. Многофункциональный пакет программ ЛОГОС для расчета задач гидродинамики и тепломассопереноса на суперЭВМ: базовые технологии и алгоритмы // XII Международный семинар «Супервычисления и математическое моделирование»: Тез. докл. Саров, 2010. С. 65-66.
11. Бартенев Ю.Г., Бондаренко Ю.А., Ерзунов В.А., Карпов А.П., Макаров Л.С., Наумов А.О., Петров Д.А., Романова М.Д., Стаканов А.Н., Щаникова Е.Б., Харченко С.А., Сушко Г.Б., Коньшин И.Н., Прилуцкий М.Х., Старостин Н.В., Филимонов

А.В. Комплекс LParSol для решения СЛАУ // Труды XIII Международного семинара «Супервычисления и математическое моделирование» / Под ред. Р.М. Шагалиева. Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2011.
12. OpenCL. The Open Standard for Heterogeneous Parallel Programming. Khronos Group, 2009.
13. Klemm M. Programming for the Intel Many Integrated Core Architecture. Intel Corporation, 2012.

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ И МАСШТАБИРУЕМОСТИ КОДА PICADOR ДЛЯ МОДЕЛИРОВАНИЯ ПЛАЗМЫ

С.И. Бастраков[1], А.А. Гоносков[2], Р.В. Донченко[1], Е.С. Ефименко[2], А.С. Малышев[1], И.Б. Мееров[1], И.А. Сурмин[1]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]Институт прикладной физики РАН

Введение

Теоретические исследования поведения вещества в экстремальных условиях, при которых происходит ионизация вещества, т.е. переход в состояние плазмы, приобретают всё большее значение для решения многих прикладных и фундаментальных задач. Эти исследования представляют интерес для целого ряда приложений, среди которых можно выделить лазерное ускорение заряженных частиц, концепцию быстрого поджига для управляемого термоядерного синтеза и генерацию излучения с уникальными характеристиками в труднодоступных частотных диапазонах. Среди медицинских приложений стоит отметить адронную терапию, которую используют для лечения онкологических заболеваний.

Часто, в связи с высокой степенью нелинейности и геометрической сложностью задачи, исследование динамики плазменных структур основывается на моделировании плазмы методом частиц в ячейках (particle-in-cell, PIC) [12]. Основная идея метода заключается в одновременной обработке принципиально разнородных массивов данных, содержащих информацию о координатах и скоростях заряженных частиц плазмы и об электромагнитном поле, заданном в узлах дискретной решетки, представляющей часть трехмерного пространства. Для решения прикладных задач необходимо создание специализированного программного обеспечения, ориентированного на использование суперкомпьютеров – существуют задачи, требующие моделирования $\sim 10^9$ частиц и $\sim 10^8$ узлов пространственной сетки.

В настоящее время разрабатывается ряд пакетов (PIC-кодов), основанных на методе частиц в ячейках:

- VLPL [7], разрабатываемый в немецком институте Max-Planck-Institut für Quantenoptik, первая система, позволяющая проводить эксперименты по полномасштабному трёхмерному моделированию релятивистских плазменных взаимодействий. Использует MPI для работы на кластерных системах.
- OSIRIS [4], разрабатываемый португальским институтом Instituto Superior Técnico совместно с другими институтами Европы и Америки, поддерживает двумерное и трёхмерное моделирование и, подобно VLPL, предназначен для работы на традиционных кластерных системах.
- QuickPIC [6], разрабатываемый четырьмя институтами США и Португалии, использует упрощенную модель для достижения большей эффективности вычислений, а именно стократного или большего ускорения по сравнению с OSIRIS.
- VPIC [3], разрабатываемый в лаборатории Los Alamos National Laboratory (США), который, помимо обыкновенных кластерных систем на базе процессорной архитектуры x86, был портирован на кластеры на базе процессоров IBM Cell, чтобы обеспечить возможность запуска на суперкомпьютере Roadrunner.

- PIConGPU [5], разрабатываемый в немецком университете Technische Universität Dresden, который предназначен для работы на кластерах из графических процессоров и имеет версии для интерфейсов NVIDIA CUDA и OpenCL.

Пакет Picador [1, 2, 10, 11] разрабатывается коллективом сотрудников ННГУ и ИПФ РАН с 2010 года. Отличительной особенностью пакета является поддержка гетерогенных кластеров. Пакет позволяет выполнять расчеты как на обычных, так и на графических процессорах. В работе приводится анализ производительности и масштабируемости пакета Picador при решении модельных задач.

1. Постановка задачи и метод решения

С физической точки зрения задача моделирования плазмы заключается в моделировании пространственно-временной динамики заряженных частиц (электронов и ионов) и электромагнитного поля. Основная идея метода частиц в ячейках [12] заключается в том, что ансамбль заряженных частиц заменяется сравнительно небольшим количеством более массивных и обладающих большим зарядом частиц, с сохранением общей массы и заряда. Это упрощение позволяет моделировать движение каждой такой частицы в электромагнитном поле под действием силы Лоренца. Эволюция электрического и магнитного полей описывается при помощи уравнений Максвелла, в которые входят токи, создаваемые движением заряженных частиц.

Классическая схема метода частиц в ячейках состоит из 4 этапов: интегрирование уравнений поля, взвешивание полей, интегрирование уравнений движения, взвешивание токов. Для интегрирования уравнений поля используется алгоритм FDTD [9], [8]. На пространство накладывается равномерная сетка, внутри каждой ячейки компоненты электрического поля определяются на серединах рёбер, а компоненты магнитного поля – в центрах граней. В качестве источников в уравнения Максвелла входят плазменные токи, возникающие вследствие движения заряженных частиц. В отличие от полей и токов, определенных на сетке, координата частицы является непрерывной, поэтому на этапе взвешивания полей происходит вычисление электрического и магнитного полей в точке нахождения частицы по рассчитанным ранее сеточным значениям полей. Эти значения используются при интегрировании уравнений движения с использованием алгоритма Boris [12] для вычисления положения и скорости частицы в следующий момент времени. На этапе взвешивания токов определяются сеточные значения плотности тока, возникающего вследствие движения заряженных частиц.

2. Программная реализация

Пакет Picador разрабатывается на языке C++. Вычислительно трудоемкие расчетные модули реализуются на языке C. Параллельная реализация для кластерных систем основана на использовании MPI. Используется пространственная декомпозиция: расчётная область делится на подобласти (домены) с перекрытием на границах. Операции над доменами выполняются параллельно. Расчеты в рамках каждого домена могут выполняться как на CPU, так и на GPU. Распараллеливание внутри домена основано на OpenMP (реализация для CPU) или OpenCL (реализация для GPU). Операции с плавающей запятой на CPU выполняются с использованием набора инструкций SSE. Возможно одновременное использование центральных и графических процессоров для расчетов. Более подробно с параллельной реализацией можно ознакомиться в работах [10], [11], [1] и [2].

3. Анализ производительности

Производительность прикладного программного обеспечения (ПО) зависит от многих факторов, среди которых качество кода, используемый компилятор и другое системное ПО, а также вычислительная система. В данной работе вычисляется степень

утилизации вычислительной мощности используемого процессора (как CPU, так и GPU) при работе пакета Picador. Степень утилизации можно определить, разделив число осуществлённых процессором арифметических операций с плавающей запятой (FLOP) за нужный промежуток времени (фаза расчетов) на максимально возможное число таких операций для конкретного процессора.

Чтобы замерить число реально выполнявшихся операций для CPU-версии, использовались аппаратные счётчики числа SSE-инструкций с плавающей запятой, доступные на процессорах Intel (SSEX_UOPS_RETIRED для типов PACKED_DOUBLE, PACKED_SINGLE, SCALAR_DOUBLE, SCALAR_SINGLE), и программный инструмент Intel VTune Amplifier XE. На GPU, к которым имелся доступ, нет счётчиков, позволяющих установить число арифметических операций с плавающей запятой. Поэтому для GPU-версии была получена лишь приблизительная оценка путём подсчёта арифметических операций в исходном коде и экстраполяции, основанной на параметрах задачи.

Для эксперимента использовался один из узлов кластера ННГУ (процессор Intel Xeon L5630 с частотой 2,13 ГГц). Пиковая производительность одного ядра составляет 8,52 GFLOPS для двойной точности и 17,04 GFLOPS – для одинарной. Для GPU NVIDIA Tesla X2070 (14 мультипроцессоров, частота 1,15 ГГц), установленных на этом кластере, пиковая производительность для двойной точности равна 515,2 GFLOPS, для одинарной – 1030,4 GFLOPS.

Использовался тест с ленгмюровскими колебаниями плазмы, в котором задействовано 32 768 ячеек и 983 000 частиц.

Таблица 1. Времена работы и метрики производительности для CPU и GPU версий

Метрика	**Взвешивание токов**	**Интегрирование уравнений Максвелла**	**Интегрирование уравнений движения**	**Итого**
CPU				
Выполнено операций, млрд.	97,14	3,32	444,98	545,44
Время, сек	35,052	2,635	343,784	381,471
Производительность, GFLOPS	2,77	1,26	1,29	1,43
Процент от пика	32,53%	14,77%	15,19%	16,78%
GPU				
Выполнено операций, млрд.	537,37	10,64	2127,91	2675,92
Время, сек	59,02	24,848	40,648	124,516
Производительность, GFLOPS	9,10	0,43	52,35	21,49
Процент от пика	1,77%	0,08%	10,16%	4,17%

Таким образом, в итоге мы получаем ≈ 17% утилизации центрального процессора и ≈ 4% – графического, что свидетельствует о наличии потенциала для оптимизации в обеих версиях. Оценка для GPU является приближённой.

4. Анализ масштабируемости

Эксперименты для оценки масштабируемости проводились на кластере Akka, который находится в High Performance Computing Center North (2x Intel Xeon L5420 2.5 ГГц, 16 ГБ RAM, Infiniband 4x, CentOS 5.6 x86_64). В качестве теста была выбрана за-

дача о ленгмюровских колебаниях плазмы, в которой 16,77 миллиона ячеек и 503,1 миллиона частиц.

Таблица 2. Результаты масштабируемости MPI-версии

Число процессов	**512**	**1024**	**2048**	**4096**
Вычисления, сек				
Токи	344,10	171,97	76,53	27,93
Поля	5,25	2,65	1,38	0,67
Частицы	687,81	345,76	170,16	83,98
(итого)	1037,16	520,38	248,07	112,58
Передачи, сек				
Токи	52,27	17,55	26,98	70,44
Поля	28,98	28,24	33,74	154,38
Частицы	52,02	36,77	36,34	112,59
(итого)	133,27	82,56	97,06	337,41
Полное время, сек	1170,43	602,94	345,13	449,99
Эффективность относительно 512 процессов	100%	96,85%	84,78%	28,56%

Плохой результат для 4096 ядер может быть объяснен тем, что тестовая задача слишком мала для такого числа процессов. Время передач для токов и полей прямо пропорционально сумме площадей граней всех доменов, то есть зависит от их размеров и количества. Для 4096 процессов домены стали слишком малыми и накладные расходы на синхронизацию передач стали занимать существенную часть времени работы.

Эксперименты, в которых одна и та же задача решается всё большим числом процессов, не вполне точно показывают реальную масштабируемость комплекса. На практике дополнительные вычислительные ресурсы можно использовать не только для решения имеющихся проблем за меньшее время, но и для решения более сложных проблем за то же время. Поэтому была проведена серия экспериментов для оценки слабой масштабируемости – способности комплекса сохранять время работы при увеличении сложности задачи пропорционально числу используемых процессов. В следующей таблице представлены результаты одного такого эксперимента на кластере Akka.

Таблица 3. Результаты слабой масштабируемости для MPI-реализации

Процессов	**512**	**768**	**1024**	**1280**	**1792**	**2048**	**3072**
Вычисления, сек							
Токи	161,68	162,01	160,9	162,72	161,01	160,74	160,66
Поля	4,49	4,35	4,40	4,45	4,36	4,47	4,42
Частицы	446,93	453,58	452,29	456,41	456,19	448,28	450,45
(итого)	613,10	619,94	617,59	623,58	621,56	613,49	615,53
Передачи, сек							
Токи	40,00	22,08	44,82	56,79	46,25	38,86	48,32
Поля	24,50	49,38	32,21	65,35	50,53	47,15	36,92
Частицы	33,43	41,51	62,20	54,18	70,04	75,00	80,44
(итого)	97,93	112,97	139,23	176,32	166,82	161,01	165,68
Полное время	711,03	732,91	756,82	799,90	788,38	774,50	781,21
Эффективность	1	0,97	0,94	0,89	0,90	0,92	0,91

Под эффективностью в данном случае понимается масштабированная эффективность, т.е. величина $E(p) = \frac{T(p,1)}{pT(p,p)} = \frac{T(1,1)}{T(p,p)}$, где $T(n,p)$ – время работы p процессов на

задаче размера n. Как и в предыдущем случае, мы считаем относительную эффективность, принимая эффективность запуска на 512 процессах за единицу. Слабая масштабируемость показывает хорошие результаты даже на 3072 процессах.

Эксперименты для графических процессоров были проведены на кластере ННГУ (2x Intel Xeon L5630 2.13 ГГц, 24 ГБ RAM, 2x NVIDIA Tesla X2070 (по 448 CUDA-ядер), Infiniband QDR, Windows Server 2008 HPC Edition SP2 x64). В качестве тестовой задачи были выбраны ленгмюровские колебания плазмы с сеткой, содержащей 32 768 ячеек и 983 000 частиц.

Таблица 4. Результаты работы гетерогенной версии

Точность	Двойная точность				Одинарная точность			
Число процессов	1 ядро CPU	8 ядер CPU	1 GPU	2 GPU	1 ядро CPU	8 ядер CPU	1 GPU	2 GPU
Вычисления, сек								
Токи	438,72	90,43	59,02	32,18	347,58	52,52	25,65	15,38
Поля	16,03	2,53	24,85	22,11	14,81	10,36	19,40	18,10
Частицы	2160,52	325,11	40,65	23,48	1681,99	247,26	20,57	12,84
(итого)	2615,27	418,07	124,52	77,77	2044,38	310,14	65,62	46,32
Передачи, сек								
Токи	4,20	3,72	4,15	2,84	3,46	3,45	3,40	2,21
Поля	3,69	2,76	5,12	3,79	3,64	12,15	4,06	2,52
Частицы	4,20	8,98	5,50	7,61	3,19	12,90	4,74	6,79
(итого)	12,09	15,46	14,77	14,24	10,29	28,50	12,20	11,52
Полное время, сек	2627,36	433,53	139,29	92,01	2054,67	338,64	77,82	57,84

Из таблицы 4 видно, что один GPU работает быстрее, чем восемь CPU-ядер, примерно в 3,11 раза в двойной точности. К сожалению, при совмещении CPU- и GPU-процессов узла не удалось добиться более высокой производительности, чем при использовании только двух имеющихся GPU. Дело в том, что, хотя в целом GPU-процессы быстрее CPU-процессов, стадия интегрирования уравнений Максвелла на них выполняется медленнее. Таким образом, независимо от того, как расчётная область разделена между CPU- и GPU-процессами, по крайней мере одна из стадий будет выполняться разное время, и часть потоков будет вынуждена ожидать завершения вычислений в другой части.

5. Заключение

Для разработанной параллельной версии кода Picador был проведён анализ производительности, масштабируемости, слабой масштабируемости в CPU- и GPU-версиях. Анализ производительности указал на наличие потенциала для программной оптимизации кода с целью повышения утилизации вычислительных устройств. Исследование масштабируемости показало, что время работы можно сократить при помощи асинхронных передач либо подбирая число процессов исходя из размеров задачи. Рассмотрение слабой масштабируемости дало представление о возможности использования кода для решения сложных задач, а анализ гетерогенной версии выявил проблему различного времени работы для разных этапов в CPU- и GPU-версиях.

Литература

1. Bastrakov S., Donchenko R., Gonoskov A., Efimenko E., Malyshev A., Meyerov I., Surmin I. Particle-in-cell plasma simulation on heterogeneous cluster systems // Journal of Computational Science. 2012. V. 3. P. 474-479.

2. Bastrakov S., Gonoskov A., Donchenko R., Efimenko E., Malyshev A., Meyerov I. Three-dimensional Particle-In-Cell Plasma Simulation on Heterogeneous Computing Systems // Twenty-Second International Conference on Numerical Simulation of Plasmas. 2011. – [http://icnsp2011.pppl.gov/abstracts/Meyerov_I_Three-dimensional%20Particl.pdf].
3. Bergen B., Albright B.J., Bowers K.J., Daughton W., et al. Petascale Plasma Physics Simulations on Roadrunner Using VPIC. 2008. – [http://www.lanl.gov/orgs/adtsc/docs/5Bergen.pdf].
4. Fonseca R.A., Silva L.O., Tsung F.S., Decyk, V.K., Lu W., et al. OSIRIS: A Three-Dimensional, Fully Relativistic Particle in Cell Code for Modeling Plasma Based Accelerators. // Proceedings of the International Conference on Computational Science. Part III. 2002. P. 342-351.
5. Hönig W., Schmitt F., Widera R., Burau H., et al. A Generic Approach for Developing Highly Scalable Particle-Mesh Codes for GPUs. 2010. – [http://saahpc.ncsa.illinois.edu/10/papers/paper_10.pdf].
6. Huang C., Decyk V.K., Zhou M., Lu W., et al. QuickPIC: a highly efficient fully parallelized PIC code for plasma-based acceleration // Journal of Physics: Conference Series. 2006. V. 46. P. 190-199.
7. Pukhov A. High Performance 3D PIC Code VLPL: Virtual Laser Plasma Lab. 1999. – [http://www.billingpreis.mpg.de/hbp99/pu4hbp99.pdf].
8. Taflove A. Computational Electrodynamics: The Finite-Difference Time-Domain Method. – London: Artech House, 1995. – 599 p.
9. Yee K. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media // IEEE Transactions on Antennas and Propagation. – V. 14. – P. 302-307.
10. Бастраков С.И., Гоносков А.А., Донченко Р.В., Ефименко Е.С., Малышев А.С., Меров И.Б. Исследование и поиск наиболее эффективных подходов к параллельному моделированию плазмы методом частиц в ячейках на кластерных системах // Параллельные вычислительные технологии (ПаВТ’2011). Труды Международной научной конференции. – Челябинск: Издательский центр ЮУрГУ, 2011. – С. 411-417. – [http://omega.sp.susu.ac.ru/books/conference/PaVT2011/short/118.pdf].
11. Бастраков С.И., Гоносков А.А., Донченко Р.В., Ефименко Е.С., Малышев А.С., Меров И.Б. Моделирование плазмы методом частиц в ячейках на гетерогенных кластерных системах // Параллельные вычислительные технологии (ПаВТ’2012): Труды Международной научной конференции. – Челябинск: Издательский центр ЮУрГУ, 2012. – С. 367-372. – [http://pavt.susu.ru/2012/short/062.pdf].
12. Бэдсел Ч., Ленгдон А. Физика плазмы и численное моделирование: Пер. с англ. – М.: Энергоатомиздат, 1989. – 452 с.

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ПРИ МОДЕЛИРОВАНИИ РЕЛАКСАЦИИ МАССИВА КРЕМНИЕВЫХ НАНОКРИСТАЛЛОВ ПО МЕТОДУ МОНТЕ-КАРЛО

В.А. Беляков[1], К.В. Сидоренко[2], И.Б. Крылов[1]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского

[2]Научно-исследовательский физико-технический институт, Нижний Новгород

Рассматривается задача моделирования релаксации массива кремниевых нанокристаллов. Предлагается схема решения задачи с использованием метода Монте-Карло, включающая случайное размещение массива нанокристаллов в плоском слое и моделирование процесса эмиссии массива. Описаны используемые алгоритмы и схема их распараллеливания и адаптации к выполнению на графическом ускорителе. Приведены результаты вычислительных экспериментов.

Введение

Кремний является основным элементом современной микроэлектроники. Возможность высокой очистки кремния, его доступность и дешевизна, высокие качества границы раздела Si/SiO_2 обеспечили кремнию лидирующие позиции в создании различных приборов микроэлектроники и их применении. Однако в оптоэлектронике кремний пока не получил широкого распространения по причине фундаментальной особенности кремниевой зонной структуры – ее непрямозонности. Волновые функции электронов (в зоне проводимости) и дырок (в валентной зоне) являются полностью локализованными в импульсном пространстве. При этом значения импульсов для электронов и дырок очень сильно различаются, и разница эта превосходит на 3–4 порядка импульс фотона, который мог бы быть излучен при межзонном электронно-дырочном переходе. В результате, закон сохранения импульса при излучательном межзонном переходе не выполняется, что приводит к полному запрету таких переходов в системе.

Преодоление этого препятствия видится в наноструктурировании кремния, т.е. создании различных структур на основе квантовых точек – нанокристаллов – кремния, динамика электронов в которых описывается чисто квантовыми законами. В нанокристалле, помещенном в широкозонную диэлектрическую матрицу, например в SiO_2, электронное состояние оказывается локализованным внутри нанокристалла, что ведет к «размытию» волновых функций электронов и дырок в импульсном пространстве. Это, в свою очередь, делает электронно-дырочный излучательный переход разрешенным.

Теоретическое изучение оптических свойств нанокристаллов кремния чаще всего выполняется для одного нанокристалла, в то время как в экспериментах всегда приходится иметь дело с их ансамблями, содержащими миллионы или даже миллиарды кристаллитов. Эти ансамбли характеризуются разбросом по размерам кристаллитов, что само по себе может приводить к некоторым особенностям в спектрах излучения, а кроме того, в плотных массивах нанокристаллов становится возможным энергообмен между нанокристаллами, что самым существенным образом может сказаться на люминесценции массива.

1. Задача численного моделирования

Адекватное описание световой эмиссии массива нанокристаллов должно учитывать возможность энергообмена между элементами массива, а также различные безызлуча-

тельные релаксационные процессы, идущие в самих нанокристаллах и составляющие конкуренцию излучательным межзонным переходам. Энергообмен между нанокристаллами может проявляться в первую очередь посредством туннельной миграции возбужденных носителей (электронов и дырок), а также посредством экситонного переноса, идущего по механизму Ферстера-Декстера. Среди безызлучательных процессов, идущих «внутри» нанокристалла независимо от наличия «соседей», следует выделить захват электронов на оборванные связи на поверхности нанокристалла и Оже-рекомбинацию. Все эти безызлучательные процессы, как и излучательные межзонные переходы, носят вероятностный характер и описываются скоростями (вероятностями за единицу времени), значения которых известны.

Количество нанокристаллов в массиве должно быть таким, чтобы позволить набрать достаточную статистику излучательных переходов при релаксации массива. Число нанокристаллов не может быть величиной порядка сотни или тысячи, поскольку безызлучательные процессы обладают на несколько порядков большими скоростями, чем излучательные переходы, вследствие чего из сотни или даже тысячи нанокристаллов подавляющее большинство срелаксирует безызлучательно, и только единицы сумеют испустить фотон. Мы используем количества нанокристаллов в массиве на уровне 10^8.

Задача моделирования люминесценции в плотных ансамблях нанокристаллов кремния разбивается на два логических этапа:

- случайное размещение нанокристаллов в плоском слое;
- непосредственное моделирование процесса эмиссии массива.

2. Размещение нанокристаллов в плоском слое

Нанокристаллы представляют собой окружности со случайными радиусами, разброс значений которых будем описывать логарифмическим нормальным распределением (как показывают эксперименты, именно логарифмическое нормальное распределение наиболее точно описывает статистику размеров нанокристаллов в реальных структурах).

В рамках первого этапа равномерно размещаются окружности (нанокристаллы) внутри заданного прямоугольника на плоскости так, чтобы они не пересекались (пуассоновское размещение). Для этого используется иерархическое итерационное разбиение области и бросание центров окружностей в рамках полученных подобластей – так называемое параллельное пуассоновское размещение окружностей, включающее выполнение следующей последовательности действий:

1) на заданную прямоугольную область вбрасывается точка (центр окружности), при этом проверяется, нет ли пересечений данной окружности с уже имеющимися;
2) область разбивается на четыре равные непересекающиеся подобласти (делением пополам по вертикали и горизонтали), вброшенная точка приписывается той новой подобласти, куда она попала после разбиения;
3) в каждую из получившихся подобластей, не содержащую точки, снова бросается точка, как на шаге 1.

Обход подобластей на шаге 3 осуществляется в случайном порядке. В случае когда при размещении окружности в подобласти имеет место пересечение с уже размещенными окружностями, делается еще несколько попыток случайного размещения, количество этих попыток является параметром алгоритма. Алгоритм завершается либо по достижении нужной плотности, либо когда размер подобластей будет недостаточен для дальнейшего размещения окружностей.

Следующие дополнения обеспечивают распараллеливание алгоритма:

- после разбиения области из всех ее подобластей выделяются фазовые группы: подобласти принадлежат одной фазовой группе, если размещение окружностей в них возможно независимо и параллельно; в частности, к одной фазовой группе могут относиться подобласти, которые расположены на определенном расстоянии друг от друга;
- выполняется последовательная обработка фазовых групп, при этом области из одной фазовой группы обрабатываются параллельно; выбор фазовой группы для обработки на каждом шаге происходит случайным образом.

Были реализованы последовательная и параллельная (для систем с общей памятью с использованием технологии OpenMP) версии описанного алгоритма для CPU.

3. Моделирование люминесценции массива нанокристаллов

Для каждого нанокристалла рассматриваются четыре конкурирующих между собой процесса – межзонная излучательная рекомбинация (скорость τ_R^{-1}), безызлучательная рекомбинация (захват экситона на оборванную связь плюс Оже-рекомбинация), идущая со скоростью $\tau_{NR}^{-1} = \tau_c^{-1} + \tau_A^{-1}$, туннелирование электрона или дырки (скорость τ_t^{-1}) и экситонный переход (скорость τ_x^{-1}). Также может реализоваться еще одно, так называемое «нулевое», событие, в результате которого система остается в исходном состоянии, т.е. ни одно из вышеперечисленных четырёх событий не происходит. Шаг по времени δt выбирается так, чтобы нулевое событие обладало наибольшей вероятностью, много большей вероятности любого из других событий, – самое быстрое из всех событий, исключая нулевое, среди всех нанокристаллов массива имеет вероятность, не превышающую 0.01.

При инициализации алгоритма выполняются следующие предварительные вычисления:

- для каждого нанокристалла рассчитываются энергии основного уровня по формулам: $E_e = 0.7/R^{3/2}$ эВ – для электронов; $E_h = 0.8/R^{3/2}$ эВ – для дырок; $E_{ph} = E_e + E_h + 1.12$ эВ – для фотонов (радиус R берется в нанометрах);
- для каждого нанокристалла в соответствии с его размером задаются скорости излучательного перехода, Оже-процесса и захвата носителя на оборванную связь;
- задается начальное число электронов N_e, дырок N_h и фотонов N_{ph} в каждом нанокристалле (в экспериментах использовались значения $N_e = 1$, $N_h = 1$, $N_{ph} = 0$);
- случайным образом определяется наличие оборванной связи в нанокристалле (вероятность 0.6);
- для каждого нанокристалла определяется набор соседей, с которыми он способен обмениваться электронами, дырками и экситонами, посредством проверки выполнения условий $L < 2$ нм и $R_1 < R_2$, где L – расстояние между краями нанокристаллов, R_1 – радиус нанокристалла, у которого ищется окружение, R_2 – радиус соседнего нанокристалла; для всех соседей рассчитываются скорости туннелирования электронов и дырок и скорости экситонных переходов.

На каждом шаге алгоритма выполняются следующие действия:

- рассчитывается оптимальный шаг по времени по формуле $\delta t = 10^{-[\lg(P_m)+2]}$, где P_m – максимальная среди всех кристаллитов вероятность наступления ненулевого события;
- для каждого нанокристалла строится линейка вероятностей наступления возможных событий и выбирается одно из них путем генерации случайного значения.

Были реализованы последовательная версия данного алгоритма для CPU, параллельная для CPU для систем с общей памятью (OpenMP), параллельная версия для графического ускорителя GPU (CUDA), а также версия для работы на системе с распределенной памятью, использующей GPU, выполняющая расчет для многослойных систем – каждый отдельный слой моделируется на своем узле кластера, используя при этом вычислительные возможности GPU (использовалась технология MPI).

Были проведены эксперименты моделирования экситонного переноса для массива из 1000 нанокристаллов, размещенных с разной плотностью. Рисунок 1 показывает зависимость общего количества экситонов от времени при четырех вариантах плотности размещения нанокристаллов.

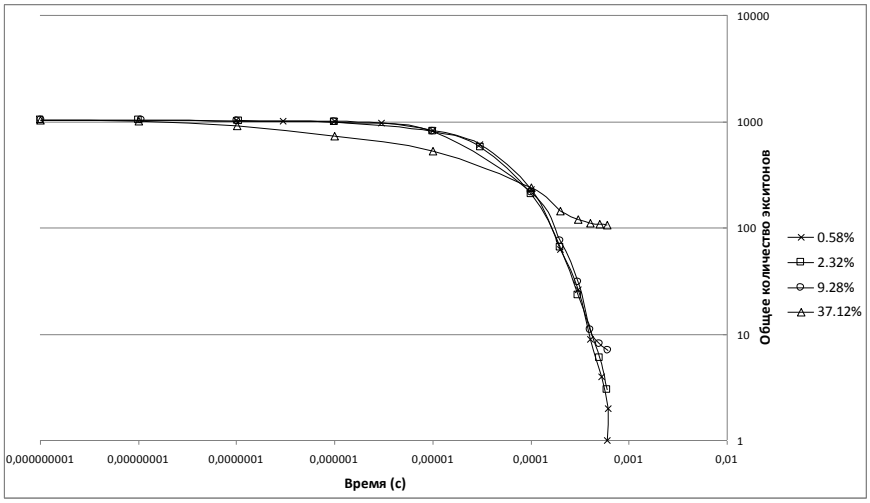


Рис. 1. Зависимость общего количества экситонов в системе от времени для различных плотностей заполнения области нанокристаллами

4. Производительность решения

Вычислительные эксперименты проводились на следующей платформе:

– язык программирования: C;

– используемые технологии: OpenMP, CUDA, MPI;

– операционная система: Microsoft Windows Server 2008 HPC Edition SP2 x64;

– среда разработки: Microsoft Visual Studio 2010;

– компилятор: Intel C++ Compiler XE 12.1;

– 16 узлов: 2 CPU Intel Xeon L5630 2.13 ГГц, 4 ядра; 24 ГБ оперативной памяти; NVidia Tesla M2050.

На рис. 2-4 представлены временные результаты и ускорение параллельных алгоритмов размещения окружностей, моделирования люминесценции и экситонного переноса (максимально использовалось 7168 графических ядер).

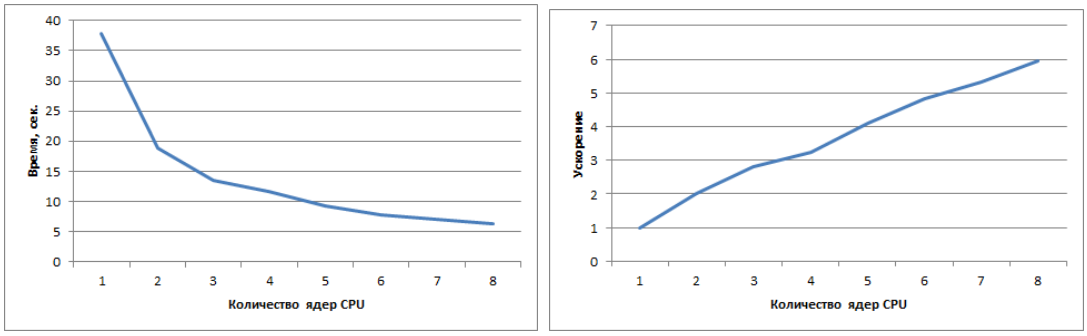


Рис. 2. Время работы и ускорение алгоритма размещения окружностей

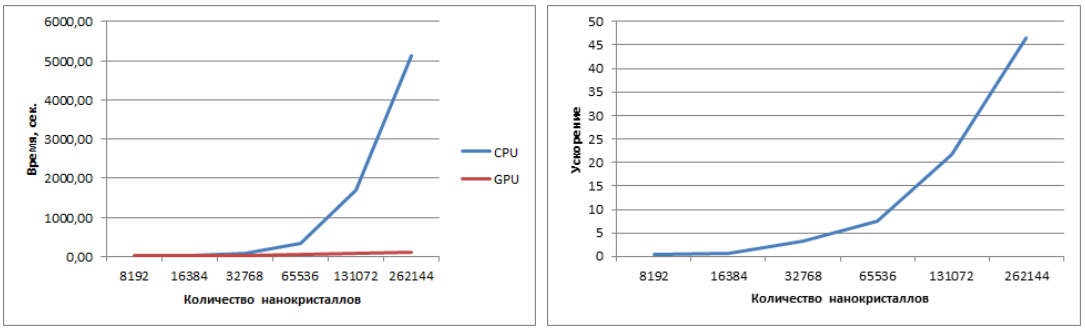


Рис. 3. Время работы и ускорение алгоритма моделирования люминесценции и экситонного переноса (параллельная CPU-версия и параллельная GPU-версия)

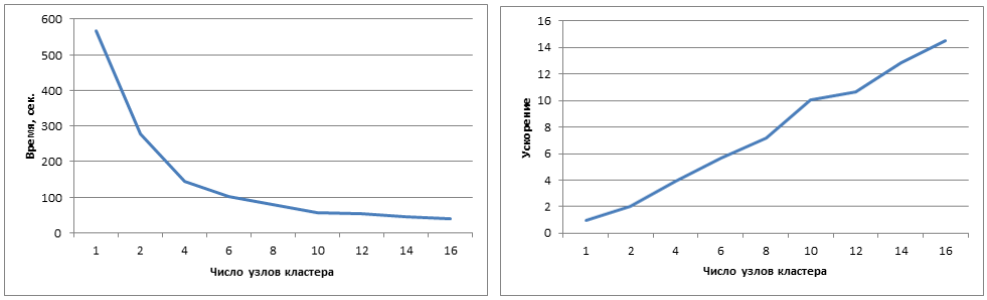


Рис. 4. Маштабируемость алгоритма моделирования люминесценции и экситонного переноса на системе с распределенной памятью, использующей графические ускорители

Заключение

Авторами была исследована возможность выполнения численного моделирования релаксации массива кремниевых нанокристаллов по методу Монте-Карло на гибридных вычислительных системах с распределенной памятью, использующих графические ускорители. Были предложены и реализованы схемы распараллеливания алгоритмов формирования массива нанокристаллов и моделирования люминесценции. Результаты экспериментов показывают, что предложенные параллельные решения обладают хорошей масштабируемостью.

Результаты вычислительных экспериментов были использованы для обоснования модели энергообмена между кремниевыми квантовыми точками посредством переноса элементарных возбуждений (экситонов).

Литература

1. Takagi H., Ogawa H., Yamazaki Y., Ishizaki A., and Nakagiri T. Quantum size effects on photoluminescence in ultrafine Si particles // Appl. Phys. Lett. 56, 1990. P. 2379–2380.
2. Pavesi L., Dal Negro L., Mazzoleni C., Franzo G., and Priolo F. Optical gain in silicon nanocrystals // Nature 408, 2000. P. 440–444.
3. Forster T. Versuche zum zwischenmolekularen. Ubergang von elektronenanregungsenergie // Zeitschrift fur Elektrochemie 53, 1949. S. 93.
4. Forster T. Experimentelle und theoretische Untersuchung des zwischenmolekularen Ubergangs von Elektronenanregungsenergie // Zeitschrift fur Naturforschung A 4, 1949. 321 s.
5. Dexter D.L. A theory of sensitized luminescence in solids // J. Chem. Phys. 21, 1952. 836 p.
6. Беляков В.А., Конаков А.А., Бурдов В.А. Миграция возбужденных носителей в ансамблях нанокристаллов кремния, легированных фосфором // ФТП 44, 2010. С. 1466.
7. Lagae A., Dutre P. A Comparison of Methods for Generating Poisson Disk Distributions // Computer Graphics, 2008. Vol. 27, № 1, P. 114–129.
8. Wei L.-Y. Parallel Poisson Disk Sampling // Microsoft Research Asia, 2008.

ОЦЕНКИ ВРЕМЕНИ В МОДЕЛИ ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ ПРОГРАММ

В.А. Биллиг

Тверской государственный технический университет

Рассматривается модель параллельных вычислений в вычислительной системе с общей памятью. Выполняемая программа рассматривается как множество модулей, связанных по данным. Граф зависимостей отражает эту связь. Целью работы является получение оценок времени выполнения программы одним процессором – T_1, конечным числом процессоров –T_p и для идеализированного случая, когда число процессоров не ограничивается – T_∞. Оценки получены для случая, когда времена выполнения модулей программы различны.

Параллельные вычисления становятся одним из магистральных направлений развития информационных технологий. Можно указать на две причины, определяющие важность этого направления. Первая состоит в том, что стратегически важные для развития государства задачи могут быть решены только с применением суперкомпьютеров, обладающих сотнями тысяч процессоров, которые нужно заставить работать одновременно. Вторая причина связана с другим полюсом компьютерной техники, на котором находятся обычные компьютеры, ориентированные на массового пользователя. И эта техника становится многоядерной, и ее требуется эффективно использовать, так что параллельные вычисления требуются и здесь.

Для поддержки параллельных вычислений сделано достаточно много, от архитектуры вычислительных систем, операционных систем, языков программирования до разработки специальных параллельных алгоритмов. Тем не менее для программиста, решающего сложную задачу, построение и отладка эффективной параллельной программы все еще остается не простым занятием. В ключевом докладе В.В. Воеводина [1] отмечалось, что к началу активного внедрения вычислительных систем параллельной архитектуры в практику решения больших прикладных задач нужный теоретический фундамент не был построен, так же как и не был развит математический аппарат исследований.

Целью данной работы является рассмотрение одной из моделей параллельного вычисления. Для этой модели мы хотим получить оценки времени выполнения программы одним процессором – T_1, конечным числом процессоров –T_p и для идеализированного случая, когда число процессоров не ограничивается – T_∞.

Рассмотрим программу P, состоящую из n модулей:

$$P = \{M_1, M_2, \ldots M_n\}$$

Будем предполагать, что программа P выполняется на компьютере, обладающем некоторым числом процессоров, работающих на общей памяти. Выходные данные, полученные в результате работы модуля M_i, могут являться входными данными для модуля M_j. Так естественным образом возникает зависимость между модулями, определяющая возможный порядок их выполнения.

Множество модулей разобьём на k уровней. К уровню i отнесем те модули, для начала работы которых требуется завершение работы модулей, из которых хотя бы один принадлежит уровню $i - 1$. Модуль уровня i с номером k будем обозначать как M_k^i.

Модули, принадлежащие уровню 1, имеют все необходимые данные, полученные от внешних источников. Они не требуют завершения работы других модулей и в принципе могут выполняться параллельно, будучи запущенными в начальный момент выполнения программы.

Свяжем с программой P ориентированный граф зависимостей модулей. Граф не содержит циклов и отражает разбиение модулей на уровни. Модули являются вершинами графа, а дуги отражают зависимости между модулями. Дуга ведет от модуля M_k^i к модулю M_l^j, если для начала выполнения модуля M_k^i требуется завершение работы модуля M_l^j. В узлах графа содержится информация об ожидаемом времени выполнения модуля, где время измеряется в некоторых условных единицах. На рис.1 показан пример графа зависимостей.

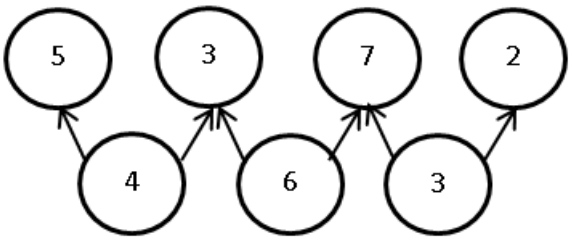


Рис. 1. Пример графа зависимостей

Обозначим через T_1 время, требуемое для выполнения программы P одним процессором, T_p – p процессорами, T_∞ – время, требуемое в случае, когда число процессоров неограниченно. В последнем случае достаточно n процессоров, по числу модулей нашей программы.

Предполагается, что все эти характеристики рассчитываются при соблюдении двух условий:

- Выполняются зависимости между модулями, заданные графом зависимостей.
- Характеристики вычислены для оптимального расписания работы процессоров.

В случае одного процессора достаточно выполнения только первого условия. Обычно предполагается естественный порядок выполнения модулей – последовательное выполнение модулей одного уровня, затем переход к выполнению модулей следующего уровня.

Для случая неограниченного числа процессоров оптимальным является такое расписание, когда каждый модуль начинает выполняться, как только завершены все модули, необходимые для его работы.

Для случая p процессоров можно распределить модули по процессорам, задав для каждого процессора P_i множество модулей, выполняемых этим процессором:

$$D(P_i) = \{M_{i,1}, M_{i,2}, \dots M_{i,r}\}.$$

Распределение модулей по процессорам совместно с графом зависимостей однозначно определяет расписание работ и время выполнения программы при данном расписании. Предполагается, что каждый процессор выполняет модули из распределения $D(P_i)$. После завершения очередного модуля он сразу же переходит к выполнению следующего модуля, если для этого модуля выполнены все зависимости, заданные графом зависимостей. В противном случае процессор ждет окончания работы требуемых модулей. Время завершения последнего модуля в распределении $D(P_i)$ задает время работы данного процессора. Тот процессор, который последним заканчивает работу, и определяет общее время решения задачи T_p^D для данного расписания. Введенная ранее характеристика T_p предполагает оптимальное расписание:

$$T_p = \min_D T_p^D.$$

Задача составления оптимального расписания относится к сложным задачам [3]. На практике для программ большого размера не удается явно вычислить значение T_p. По этой причине несомненный интерес представляет получение оценок для T_p.

Для введенных характеристик выполняется естественное соотношение:

$$T_\infty \le T_p \le T_1. \quad (1)$$

Нас будет интересовать получение более точных оценок для T_p. В работе В.П. Гергеля [2] показано, что для случая, когда время выполнения всех модулей одинаково, имеют место следующие соотношения:

$$\frac{T_1}{p} \le T_p \le \frac{T_1}{p} + T_\infty. \quad (2)$$

Рассмотрим более интересный для практики случай, когда модули программы для своего выполнения требуют разного времени. Пусть M^i – множество модулей уровня i:

$$M^i = \{M_1^i, M_2^i, \dots M_{k_i}^i\}. \quad (3)$$

Для каждого из этих модулей известно время, требуемое на его выполнение, – $t_{i,j}$.

И в этом случае нетрудно рассчитать время T_1 – время, требуемое на выполнение всей работы одним процессором:

$$T_1 = \sum_{i=1}^{k} \sum_{j=1}^{k_i} t_{i,j}. \quad (4)$$

Как рассчитать время T_∞ в этой ситуации, когда мы располагаем неограниченным числом процессоров? Введем для каждого модуля время окончания его работы – t_j^i. Это время будем рассчитывать по следующей формуле:

$$t_j^i = t_{i,j} + t_{jmax}^{i-1}. \quad (5)$$

Здесь t_{jmax}^{i-1} – время окончания работы того модуля уровня i-1, который:

- необходим для работы модуля M_j^i ;
- из всех необходимых модулей завершает свою работу последним.

Тогда время T_∞ можно рассчитать следующим образом:

$$T_\infty = \max_j\{t_j^k\} \quad (6)$$

Формула (6) говорит, что время завершения последнего модуля уровня k и является временем T_∞ при оптимальном расписании работ. Справедлива следующая теорема:

Теорема 1

Время T_∞ задается максимально нагруженным путем в графе зависимостей.

Оценка снизу

Лемма 1

Для T_p справедлива оценка:

$$T_p \ge \frac{T_1}{p}. \quad (7)$$

Оценка сверху

Пусть работу выполняют p процессоров. Составление расписания означает, что граф зависимостей разбивается на p непересекающихся подграфов. Все модули каждого из подграфов выполняются одним процессором. Подграф с максимальным временем выполнения для данного разбиения будем называть максимально нагруженным подграфом. Оптимальное расписание предполагает такое разбиение, при котором максимально нагруженный подграф выполняется за минимально возможное время.

Лемма 2

Для T_p справедлива оценка:

$$T_p \le \frac{T1}{p} + T_\infty. \quad (8)$$

Дадим графическую интерпретацию. Задание нижней и верхней оценки для $T_p(p)$ означает, что эта функция ограничена двумя гиперболами. Функция убывающая. В начальной точке при p =1 по определению $T_p(1) = T_1$, так что функция находится в за-

данном коридоре. Это же справедливо и для конечных точек, для всех p, больших некоторого значения p^*, при котором $T_p = T_\infty$. Рисунок 2 иллюстрирует поведение T_p.

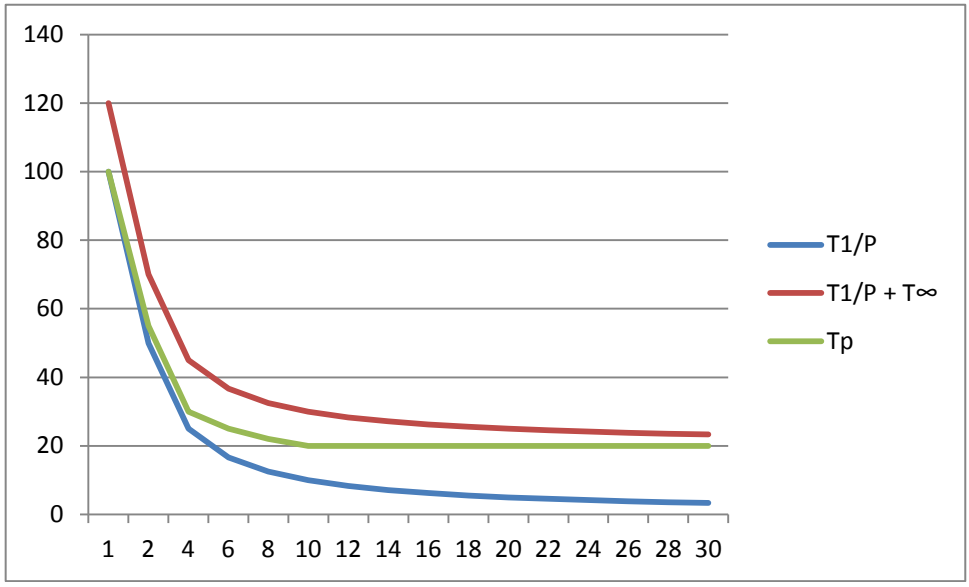


Рис. 2. Поведение функции $T_p(p)$

В заключение дадим некоторые практические рекомендации, следующие из полученных оценок. Выигрыш, который можно получить, используя дополнительные процессоры, зависит от разницы между общим временем выполнения всех модулей программы – T_1 и временем выполнения критического пути в графе зависимостей – T_∞.

Эта разница максимальна для крайнего случая, когда все модули могут выполняться независимо и в графе зависимостей все модули находятся на одном первом уровне. Критический путь в этом случае состоит из одного модуля, требующего максимального времени своего выполнения. Так что T_1 – это время выполнения всех модулей, а T_∞ – это время выполнения одного модуля. Привлечение p процессоров может дать существенный эффект, уменьшая время выполнения практически до среднего времени выполнения одного модуля T_1/p.

Эта разница минимальна для другого крайнего случая – строго последовательной программы, когда N модулей программы расположены на N уровнях и критический путь задает выполнение всех модулей. В этом случае T_1 и T_∞ совпадают и, как следствие, T_p равно T_1 при любом числе процессоров, так что привлекать дополнительные процессоры в этом случае бессмысленно.

Для строго последовательной программы дополнительные процессоры не позволяют уменьшить время выполнения программы в сравнении со временем выполнения этой же программы одним процессором. Так, например, задача о «Ханойской башне» на суперкомпьютере с сотнями тысяч процессоров будет решаться столь же долго, как и на компьютере с одним процессором. Это вытекает из сути задачи. Перенос следующего кольца требует завершения переноса предыдущего кольца, параллельно эту работу выполнять нельзя.

Литература

1. Воеводин В.В. Математические проблемы параллельных вычислений // Доклад на Всероссийской конференции «Научный сервис в сети Интернет: технологии распределенных вычислений», 2012.
2. Гергель В.П. Теория и практика параллельных вычислений. М.: Изд. «Бином», 2007.
3. Кормен Т. и др. Алгоритмы. Построение и анализ. Изд. «Вильямс», 2012.

«ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ И МНОГОПОТОЧНОЕ ПРОГРАММИРОВАНИЕ» – УЧЕБНЫЙ КУРС И УЧЕБНОЕ ПОСОБИЕ

В.А. Биллиг

Тверской государственный технический университет

Три важных направления развития ИТ достаточно тесно связаны друг с другом – суперкомпьютерные технологии, большие данные (Big Data) и интеллектуальный (глубинный) анализ данных (Data Mining). Параллельные вычисления объединяют все три направления.

Учебный курс и разрабатываемое учебное пособие посвящены основам параллельных вычислений и практике разработки параллельных программ.

Важной особенностью курса является его ориентация на язык C# и среду разработки Visual Studio. Цель курса – научить C#-программистов, работающих на традиционных многоядерных компьютерах, писать параллельные программы, эффективно используя возможности современных компьютеров. Учитывая особенности предлагаемого подхода к построению курса, в стороне остаются такие вопросы, как программирование в стандарте MPI и Open MP. Тем не менее автор уверен, что программист, понимающий проблемы параллельных вычислений, сможет достаточно просто переходить от одного стандарта к другому и перейти без труда к программированию с использованием специальных средств Intel.

Первая глава пособия является введением в мир параллельных вычислений. Здесь рассматривается некоторая модель параллельных вычислений. Вводятся характеристики таких вычислений – временные, ускорение, эффективность, упущенная эффективность. Рассматривается закон Амдаля, Густавсона-Барсиса, доказываются некоторые теоремы, связывающие временные характеристики. Здесь же рассматриваются проблемы сложности вычислений, возможности распараллеливания по данным и по задачам. Анализируются проблемы, возникающие при распараллеливании, – проблемы синхронизации, гонки данных, клинча.

Во второй главе анализируется взгляд на вычисления с другой стороны – с позиций вычислительной системы, ее операционной системы. Рассматривается классификация вычислительных систем. Рассматриваются понятия процессов и потоков, возникающих в операционной системе.

Третья глава посвящена алгоритмической природе параллельных вычислений. Рассмотрен ряд классических алгоритмов – их последовательные версии и версии, позволяющие вести вычисления параллельно при наличии нескольких процессоров. При рассмотрении параллельных алгоритмов не рассматриваются вопросы обмена данными между процессорами. Предполагается, что алгоритмы могут выполняться на системах с общей памятью. Выбранный набор алгоритмов достаточно широк и традиционен для демонстрации параллельных вычислений. Все начинается с вычисления суммы элементов, пирамидального алгоритма, алгоритма, ориентированного на конечное число процессоров. Затем рассматриваются более сложные алгоритмы – вычисления сходящихся рядов, интегралов, алгоритмы решения задач линейной алгебры, алгоритмы сортировки. Для всех рассматриваемых алгоритмов приводятся программы, написанные на языке C#.

Вторая часть учебника и курса посвящена практическим вопросам программирования в среде Visual Studio на языке C#. Рассматриваются проблемы создания параллельных программ с использованием возможностей класса Parallel. Обсуждаются достоинства и недостатки такого подхода. Показано, как алгоритмы, описанные в третьей главе, могут быть реализованы с использованием методов этого класса – Parallel.For и других.

Существенную часть курса занимает многопоточное программирование – пространство имен Threading и многочисленные классы этого семейства. Подробно рассматриваются на примерах все проблемы параллельных вычислений – создание и уничтожение потоков, их синхронизация, проблемы гонки данных, блокировки, клинча.

Тема параллелизма и многопоточности обсуждается и в более широком аспекте – отделение интерфейса приложения от бизнес-логики, создание приложений, моделирующих производственные процессы, допускающих визуализацию и управление на основе поступающих данных. Игры являются хорошим примером подобных приложений.

На создание этого курса большое влияние оказали работы В.П. Гергеля, выступления на конференциях и его учебники, доступные на сайте Интернет-университета ИТ – www.intuit.ru.

МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ И ДИНАМИЧЕСКОЙ БАЛАНСИРОВКИ НАГРУЗКИ В СИСТЕМЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

И.П. Болодурина, Д.И. Парфёнов

Оренбургский госуниверситет

Повседневное распространение информационных технологий все чаще находит свое применение в образовательных сетевых мультимедийных системах. При этом актуальной задачей является исследование эффективных методов управления производительностью и оптимизации использования программных и аппаратных ресурсов. В рамках представленного исследования построена многоуровневая модель системы дистанционного обучения, проведен анализ характеристик и приведен алгоритм для повышения эффективности использования имеющихся ресурсов с целью улучшения качества предоставления услуг в распределенных информационных системах дистанционного обучения.

Введение

Повседневное распространение информационных технологий все чаще находит свое применение в образовательных сетевых мультимедийных системах. При этом наиболее активно развивающимся направлением является дистанционное обучение. В последнее время широкое распространение получили такие интерактивные веб-сервисы, построенные на базе современных информационных технологий, как:

- цифровое телевидение (TV over IP – TVoIP);
- видео по запросу (Video on Demand – VoD);
- Интернет-трансляции;
- вебинары;
- веб-конференции.

Однако, несмотря на широкий круг решаемых задач, повсеместное внедрение перечисленных ранее сервисов в образовательных учреждениях остается весьма затруднительным. Для обеспечения необходимого качества предоставляемых услуг требуется выделение специализированных аппаратных ресурсов, а также каналов связи.

Университеты обладают собственными аппаратными (серверными) ресурсами для обеспечения внутренней деятельности и арендуют каналы связи для обмена информацией посредством сети Интернет. Существует определенная статистическая закономерность потребления имеющихся вычислительных мощностей, которая показывает, что 80% ресурсов необходимы лишь в 20% времени (справедливо и обратное утверждение). Как правило, нагрузка на аппаратные и программные ресурсы образовательного учреждения носит плавающий характер, при этом периоды пиковой нагрузки имеют прямую связь с происходящими в вузе событиями и процессами. Большинство событий носят систематический характер, что позволяет рассчитать необходимую нагрузку и подготовить требуемые ресурсы.

В настоящее время для обеспечения хранения мультимедиаконтента и доступа к ресурсам наиболее выгодным является применение гибридных облачных систем. Масштабируемость и другие характеристики, присущие облачным вычислениям, являются одним из немаловажных факторов, влияющих на тенденции размещения и предостав-

ления информационных услуг в образовательных учреждениях. Это особенно актуально для дистанционного обучения, при котором основная часть учебного процесса и взаимодействие обучающегося и преподавателя осуществляется посредством сети Интернет.

На факультете дистанционных образовательных технологий (ФДОТ) Оренбургского государственного университета (ОГУ) накоплен определенный опыт в автоматизации задач организационно-методического и программно-технического сопровождения дистанционного обучения в вузе. Кроме того, на факультете разработан консолидированный сервис – «Видеопортал дистанционного обучения», обеспечивающий доступ к перечисленным ранее услугам с целью реализации образовательных программ высшего профессионального образования, позволяющий организовать взаимодействие преподавателя и студентов на новом уровне путем создания интерактивной обратной связи. Мультимедийный сервис такого класса, помимо обеспечения постоянной доступности, требует высокого качества обслуживания при передаче данных [1].

Ежегодный прирост числа потребителей сетевых мультимедийных услуг приводит к значительному росту трафика и, как следствие, повышению нагрузки на оборудование и каналы связи. Узким местом подобных сервисов является точка вещания видеопотока ввиду ограниченности пропускной способности выходного канала. Особенно эта проблема актуальна для пользователей, осуществляющих доступ к веб-приложениям из сети Интернет. Сама по себе передача видеоконтента требует особого подхода. При доступе к уже существующему контенту создается высокая нагрузка на систему хранения данных. При онлайн-вещании (например, видеоконференции) создается высокая нагрузка на службу сжатия и обработки контента. Кроме того, специфика работы Интернета заключается в том, что для каждого клиента при обращении к сервису трансляции создается персональный поток (точка–точка), что при большом количестве обращений приводит к исчерпанию пропускной способности канала связи.

1. Постановка задачи

В рамках исследования нами выделено несколько отличительных особенностей обеспечения доступа к мультимедийным образовательным ресурсам в распределенной сети вуза:

1. Нагрузка на серверы периодическая, одновременно происходят обращения к нескольким ресурсам с разными типами. В большинстве случаев существующее оборудование не позволяет без использования распределения нагрузки обслужить всех клиентов, причем загрузка серверов носит неодновременный и наравномерный характер.
2. До 90% нагрузки предопределено, поскольку для доступа к ресурсам используются пререгистрация (подписка на сервисы), например запись на вещание лекции, а также статистические данные оценки использования информационных ресурсов, полученные на основе ежегодного отчета «об информатизации вуза». При этом использование стандартных средств не позволяет учесть предопределенную нагрузку и распределить ее в условиях ограниченных ресурсов.
3. В пределах локальной сети присутствуют различные категории полезного трафика, но при обращении к корпоративным сервисам не учитывается приоритет обслуживания и выделение полосы пропускания для критически важного трафика.

Для эффективного использования ресурсов необходимо их динамическое выделение в рамках решаемых задач для исключения простоя и перегрузки аппаратного обеспечения. Для повышения надежности и улучшения качества предоставляемых сетевых мультимедийных услуг требуется внедрение эффективных методов обеспечения распределения нагрузки аппаратно-программных ресурсов университетского комплекса.

Традиционно оптимизация использования вычислительных ресурсов осуществляется при помощи процедуры балансировки нагрузки. Как правило, балансировка заключается в распределении запросов определенным компонентам, обработчикам облачной системы на основе оценки загруженности и их состояния. Так как облачная система управляется из единого контроллера, это подразумевает, что поступивший запрос может быть передан на обработку любому из активных устройств, поддерживающих работу выбранного приложения. Однако работа приложений часто зависит не только от объема оперативной памяти и процессорного времени, требуемых для выполнения запроса пользователя. В настоящее время высоконагруженные приложения, направленные на обработку больших объемов данных, например видео- и мультимедиаконтента, невозможно представить без использования масштабируемых систем управления баз данных и распределенных систем хранения данных. Проведенный анализ публикаций по теме исследования показал [2,3,4,5], что на сегодняшний день нет достаточно эффективных универсальных, комплексных методов балансировки и распределения нагрузки, включающих в себя: выделение процессорного времени, оперативной памяти, управление потоком SQL-запросов к базе данных, а также динамическое распределение размещения файлов в системе хранения данных (СХД).

2. Решение задачи

Для детального анализа ресурсов системы дистанционного обучения нами разработана уровневая модель на основе базовых высоконагруженных доступных внешним пользователям подсистем:

- подсистема контроля знаний (уровень 1);
- подсистема предоставления учебно-методических комплексов (электронная библиотека) (уровень 2);
- подсистема трансляции и публикации видео- и аудиоматериалов (видеопортал ДО) (уровень 3).

Выделенные нами базовые компоненты могут быть представлены как комплекс, обеспечивающий работу мультисервисного набора услуг для физически *распределенных пользователей* [10]. Каждая из подсистем, используемая в системе дистанционного обучения, предъявляет собственные требования к прикладному программному обеспечению оборудования и качеству обслуживания (QOS), что позволяет проводить моделирование с использованием многокритериальных показателей и как следствие создать базу знаний для управления и распределения поступающей нагрузки.

Практика показывает, что большинство информационных систем, работающих с внешними пользователями, при большом количестве обращений испытывают недостаток в потребляемых ими ресурсах. Причем отказ в обслуживании для любой из систем напрямую зависит от объема выделенных для ее работы ресурсов. Как отмечалось ранее, прогнозирование нагрузки от клиентов позволяет подготовить оборудование и каналы связи для приема трафика. Однако это не решает проблему непрогнозируемых экстремальных нагрузок, а применение метода, основанного на увеличении времени отклика системы, приводит к удлинению очереди заявок, что снижает динамику работы системы. Такой подход невозможно организовать для сервисов реального времени, таких как потоковая передача видео- и аудиоданных. К тому же большинство систем работает по принципу First In, First Out (FIFO).

В рамках нашего исследования для системы дистанционного обучения разработан алгоритм приоритетного обслуживания клиентов высоконагруженных приложений с критичным временем отклика. В связи с этим нами решены следующие задачи:

- выделено прикладное программное обеспечение, влияющее на работу каждой из подсистем;

- определена наиболее ресурсоемкая подсистема;
- выставлены индикаторы приоритетов обработки запросов при одновременном функционировании подсистем;
- построена математическая модель для максимизации числа обработанных обращений к СДО.

Работу Интернет-приложений часто рассматривают как систему массового обслуживания с ограниченным временем пребывания в очереди и пуассоновским потоком заявок [8, 9]. Для формализации работы Интернет-приложений механизм обработки запросов будем рассматривать как многоканальную СМО с несколькими очередями (рис. 1).

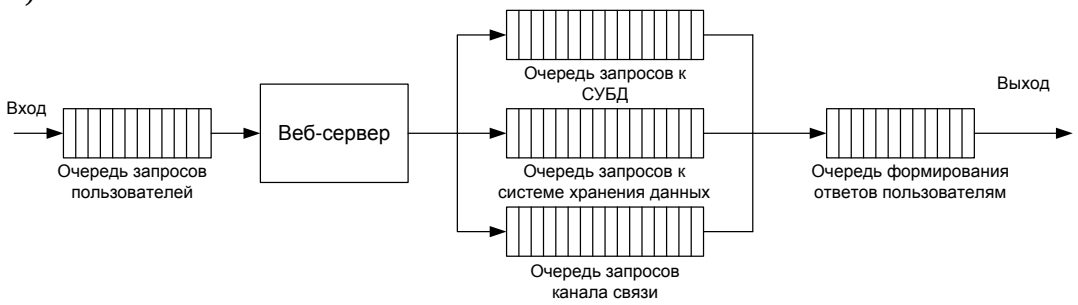


Рис. 1. Модель работы Интернет-приложения как СМО

В ходе исследования нами установлено, что на всех трех уровнях модели основными факторами, влияющими на скорости обработки запросов пользователей программным обеспечением системы дистанционного обучения, являются:

- обращение к СУБД для получения необходимых данных;
- обращение как к дисковому пространству самого сервера, так и к системе хранения данных для записи или чтения необходимых данных;
- использование приложением канала связи заданной пропускной способности в единицу времени для приема и передачи требуемого объема данных.

Для указанных выше факторов нами введены численные показатели классификационных признаков каждого из уровней построенной модели:

- количество запросов в единицу времени, отправленных к СУБД (SQL-запросов/с);
- использование дискового пространства серверного оборудования (Мб/с);
- интенсивность использования входящего/исходящего канала связи (Мбит/с).

Для каждого из уровней численные показатели в процентном соотношении к суммарному показателю использования данного ресурса всеми уровнями модели определяются выражением:

$$R_{\text{i исп}} = \frac{R_i \cdot 100}{(R_1 + \ldots + R_n)}, \qquad (1)$$

где R_1, …, R_n – численные показатели использования ресурса по каждому из классификационных признаков, полученные в результате измерений на интервале времени ΔT.

Индикаторы приоритета обслуживания уровней модели определим на основе рейтинга востребованности ресурсов системы в целом. Анализируя интенсивность использования каждого из компонентов ресурсов в СДО, получим рейтинг востребованности ключевых сервисов и аппаратного обеспечения, лежащих в основе каждой из подсистем.

Общую ресурсоемкость системы дистанционного обучения определим как суммарную площадь $U_{\text{сдо}}$, занимаемую всеми уровнями модели (U_i). При этом максимально возможные ресурсы сервера обозначим как площадь, полученную при использовании 100% всех ключевых сервисов [11].

Так как работа подсистем осуществляется непрерывно, поступление заявок к ресурсам системы (СУБД, каналы связи, дисковое пространство) можно описать в дискретном времени: $I_j(T_j)=\{j{:}t \in (0,T_i)\}$ – множество номеров заявок, пришедших в интервал времени (0, T_i) на подсистему i (i – уровень подсистемы, i = 1, …, M).

Статус обработки j-й заявки, поступившей на i-й уровень, обозначим x_{ij}, причем отказ в обслуживании будем считать x_{ij}=0, успех x_{ij}=1.

Интенсивность поступления и обработки заявок на каждый из уровней модели обозначим λ_i, при этом она напрямую зависит от ресурсоемкости подсистемы. Кроме того, введем показатель приоритета (P_i) для каждого из уровней, распределение которого зависит от количества одновременно используемых ресурсов. Тогда на нагрузку, создаваемую каждым из уровней, можно наложить ограничение:

$$\sum_{I_j(T_j)} U_i x_{ij} \le H_i \text{ , i=1,…,}M. \qquad (2)$$

При задании целевой функции введены следующие ограничения, связанные с предметной областью исследования:

- время обработки (T) любого запроса ограничено;
- мощность сервера (H) фиксирована.

Ввиду неравномерности использования основных ресурсов сервера каждым из уровней системы дистанционного обучения следует определить условия максимальной загрузки сервера, при которой возможна безотказная работа всех приложений:

$$\sum_{i=1}^{M} \sum_{j \in I_j(T_j)} U_i x_{ij} \le H \text{ , } x_{ij}=\{0,1\}. \qquad (3)$$

Таким образом, для обработки максимального количества запросов пользователей в единицу времени получим целевую функцию вида:

$$\sum_{i=1}^{M} \sum_{I_j(T_j)} \lambda_i x_{ij} P_i \to \max. \qquad (4)$$

При выборе приоритетов оцениваются следующие характеристики заявки:

- время нахождения заявки в очереди;
- текущая длина очереди заявок;
- интенсивность обращения к каждому из компонентов ресурса, необходимых для выполнения заявки.

Выбор приоритетов и оценка текущей ресурсоемкости задачи производится на основе компонентов ресурса, имеющих индивидуальные пороговые значения, связанные с физическими ограничениями оборудования.

В ходе реализации предложенной модели в распределенной информационной системе дистанционного обучения нами получены следующие показатели работы, позволяющие оценить эффективность применения разработанного алгоритма расстановки приоритетов. Анализ производился на промежутке времени ΔT= 60 секунд. Ограничение по времени обусловлено техническими параметрами (максимально допустимым временем отклика) работы приложения. Эффективность работы алгоритма приоритетов будем оценивать путем сравнения очереди заявок (общего их количества), одновременно находящихся в системе, и количества отброшенных заявок. На рис. 2 представлена диаграмма обслуживания заявок в реально работающей системе без использования предложенного алгоритма.

Рис. 2. Диаграмма обслуживания заявок без использования алгоритма расстановки приоритетов

Применив алгоритм выбора и расстановки приоритетов для каждого из ресурсов в рамках всей системы дистанционного обучения, получим снижение количества отброшенных заявок в каждый момент времени примерно в 2,7 раза, при этом общее число необработанных заявок по истечении времени обработки ΔT снизилось с 12 до 5 (рис. 3).

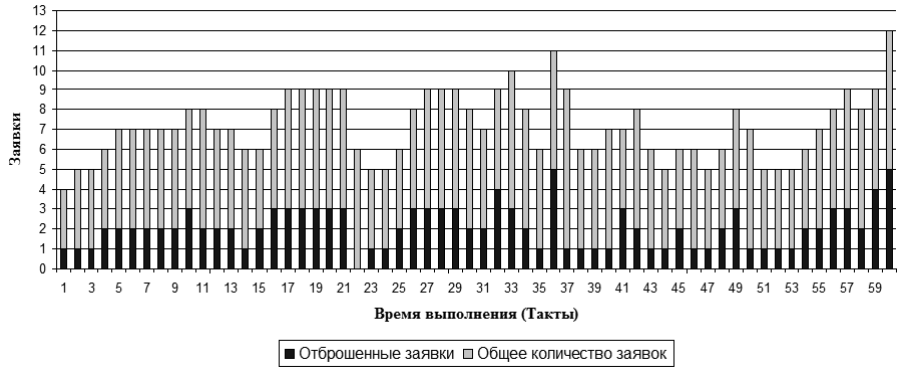


Рис. 3. Диаграмма обслуживания заявок с использованием алгоритма расстановки приоритетов

Как можем заметить, наблюдается самоподобие графиков обслуживания заявок в информационной системе. Нами проведено дополнительное исследование по оценке времени отклика системы, показавшее прирост скорости обработки заявок, по сравнению с обычной обработкой, так как средняя длина очереди снизилась с 8,6 до 5,1.

Экспериментальная апробация алгоритма проведена на симуляторе, моделирующем распределение нагрузки с использованием имитационной модели процесса взаимодействия пользователей с мультимедийными сервисами. Построенная модель и приведенный алгоритм могут применяться для повышения эффективности использования аппаратных и программных ресурсов с целью улучшения качества предоставления услуг в распределенных информационных системах дистанционного обучения, а также предотвращения перегрузки сервисов в момент пиковой нагрузки.

Работа выполнена при финансовой поддержке программы «Научные и научно-педагогические кадры инновационной России», грант № 14.132.21.1801.

Литература

1. Парфёнов Д.И. Технологии и инструментальные средства организации и проведения вебинаров в системе дистанционного обучения // Труды IX Всероссийской научно-практической конференции с международным участием «Современные ин-

формационные технологии в науке, образовании и практике». – Оренбург: [Би], 2010. – С. 109–113.
2. Вашкевич Н.П., Зинкина Н.С. Активные инфологические модели обработки данных на основе иерархических сетей фреймов // Вопросы радиоэлектроники. Серия ЭВТ. 2009. Вып. 4. С. 54–63.
3. Зинкина Н.С. Агентно-ориентированный подход к проектированию распределенных систем управления базами данных // Перспективы науки. 2011. № 2. С. 80–86.
4. Бойченко И.В., Корытников С.В. Управление ресурсами в сервис-ориентированных системах типа «приложение как сервис» // Доклады Томского государственного университета систем управления и радиоэлектроники. 2010. Вып. 1-2. С. 156–160.
5. Жевнерчук Д.В., Николаев А.В. Методика моделирования нагрузки на сервер в открытых системах облачных вычислений // Информ. и её примен. 2012. С. 43–50.
6. Петров Д.Л. Оптимальный алгоритм миграции данных в масштабируемых облачных хранилищах // Управление большими системами. 2010. Вып. 30. С. 180–197.
7. Петров Д.Л. Динамическая модель масштабируемого облачного хранилища данных // Известия ЛЭТИ. №4, 2010. С. 17–21.
8. Гусев О.В., Жуков А.В., Поляков В.В., Поляков С.В. Проблема адекватной оценки производительности веб-серверов в корпоративных сетях на предприятиях ЦБП // Материалы 6-й научно-технической конференции «Новые информационные технологии в ЦБП и энергетике». Петрозаводск, 2004. – С. 84–87.
9. Жуков А.В. Некоторые модели оптимального управления входным потоком заявок в интранет-системах // Материалы 6-й научно-технической конференции «Новые информационные технологии в ЦБП и энергетике». Петрозаводск, 2004. – С. 87–90.
10. Парфёнов Д.И. Программно-аппаратный комплекс видеопортала как эффективное средство информационного взаимодействия субъектов образовательного процесса // Труды V Международной научно-практической конференции «Информационная среда вуза XXI века». – Петрозаводск: [Би], 2011. – С. 141–144.
11. Парфёнов Д.И., Болодурина И.П. Моделирование востребованности ресурсов в распределенной информационной системе дистанционной поддержки образовательного процесса // Сборник статей под реакцией А.П. Кудинова «Высокие технологии, экономика, промышленность». – СПб.: [Би], 2012. – С. 30–34.

ПАРАЛЛЕЛЬНЫЕ РАСЧЕТЫ СЛОЖНЫХ ЗАДАЧ МЕХАНИКИ СПЛОШНЫХ СРЕД НА БАЗЕ ПРОГРАММНО-ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА STAR-CCM+

П.П. Борисейко

ООО «Саровский инженерный центр»
E-mail: pboriseyko@saec.ru

Основная идея

Контроль компонентов и температуры системы остается одной из наиболее значимых проблем в дизайне электронных систем. От чипа к шасси и за его пределами, от чрезмерной тепловой нагрузки, появляются ограничения максимальной производительности электронных устройств и значительно повышается энергетический след в системе.

Одним из путей решения этих проблем является STAR-CCM+, в частности, инструмент wrapper, который позволяет игнорировать многие недостатки 3D-модели и создавать геометрическое представление, максимально адаптированное и готовое к моделированию.

Например, что касается существующей функциональности эйлеровой многофазности, в STAR-CCM+ имеется возможность тепло- и массопереноса между фазами, а также целый ряд новых опций межфазного взаимодействия, позволяющих имитировать широкий спектр различных ситуаций. К примеру, опция фаз проницаемых стенок позволяет определять эффективность захвата частиц на границах, что является ключевым фактором при изучении загрязнения автомобилей и прогнозировании обледенения самолетов.

Научно-техническая задача и ее преимущества

В отличие от других CFD-пакетов, STAR-CCM+ специально предназначен для выполнения масштабируемых параллельных вычислений на сколь угодно больших расчетных сетках. Также комплекс позволяет решать относительно простые задачи прочности (совместно с газодинамикой).

STAR-CCM+ является наиболее полным и разработанным средством инженерного моделирования, заключенным в единый интегрированный пакет. Это не просто CFD-решатель, это целый инженерный комплекс для решения задач, связанных с расчетом потоков, теплопереноса и прочности.

STAR-CCM+ предоставляет большой набор физических моделей: ламинарные и турбулентные течения, ньютоновские и неньютоновские жидкости, многофазные потоки, кавитация, излучение, горение, развитие пограничного слоя, течения с большими числами Маха, сопряженный теплообмен, а также новые модели для расчетов теплообменников и вентиляторов.

Использование произвольных многогранных ячеек в комплексе позволяет экономить объем расходуемой памяти и ускоряет решение задачи. Комплекс содержит мощные средства визуализации и управления параметрами моделирования в процессе расчета.

Отличия от существующих решений

Параллельное построение сеток

STAR-CCM+ имеет возможность распараллеливания процесса построения многогранных сеток и призматических слоев. Параллельное построение сеточных моделей приводит к уменьшению требуемого объема памяти и значительно сокращает время построения сеток.

DEM-слияние и разъединение частиц

Метод дискретных элементов (DEM) позволяет моделировать движение гранулированных потоков песка, сыпучих веществ, металлического порошка, капсул, растворов и т.д. В новой версии STAR-CCM+ этот метод усовершенствован за счет возможности моделирования слияния – «слипания» – частиц и их разъединения под давлением, а также за счет соединения гибкими связями.

Образование, перенос и удаление жидкой пленки

При моделировании разбрызгивания жидкости требуется умение прогнозировать результат взаимодействия капель разбрызганной жидкости с поверхностью твердого тела. В STAR-CCM + v6.04 встроена новая модель для прогнозирования образования и переноса тонкой пленки жидкости на поверхности твердого тела, а также ее последующего удаления и распада под воздействием аэродинамических сил.

РАЗРАБОТКА ТЕХНОЛОГИИ ХРАНЕНИЯ ДАННЫХ ДЛЯ ПАРАЛЛЕЛЬНЫХ СХД

К.В. Бородулин

Южно-Уральский госуниверситет, Челябинск

Описывается создание технологии параллельной системы хранения данных для высокопроизводительных вычислительных комплексов с пиковой производительностью до 0.5 Пфлопс.

Введение

На текущий момент большинство суперкомпьютеров имеют кластерную архитектуру, состоящую более чем из 1000 узлов [1]. Все узлы связаны высокопроизводительной сетью, позволяющей передавать до 40 Гбит/с, что обеспечивает высокую производительность на вычислительных задачах типа LINKPACK. Но при расчете инженерных задач производительность вычислений может сильно снижаться из-за того, что инженерные вычисления производят большое число обменов (сохранение состояния итераций, контрольных точек, чтение исходных данных) с системой хранения данных (СХД) [2].

Идея проекта состоит в увеличении скорости обмена данными между узлами суперкомпьютера и системой хранения данных путем организации параллельной системы хранения. Таким образом, цель – создание технологии параллельной системы хранения данных для высокопроизводительных вычислительных комплексов с пиковой производительностью до 0.5 Пфлопс. Для достижения данной цели необходимо решить следующие задачи:

- разработать алгоритмы обработки и хранения данных в параллельной СХД;
- разработать методы хранения и обработки метаданных в параллельной СУБД;
- реализовать прототип программной системы, использующий разработанные алгоритмы.

1. Обзор существующих технологий

Современные параллельные системы хранения данных (например, Panasas [2]) используют выделенный сервер для хранения метаданных и атрибутов файлов, и при повреждении данного сервера вся информация о файлах (например, на каких узлах сохранялся данный файл) будет утеряна. Также такие системы не позволяют установку сторонних компонентов, что значительно повышает стоимость ремонта и обслуживания системы.

Недостатки распределенных ФС, например lustre [4], заключаются в том, что при использовании высокопроизводительной сети скорость передачи данных с СХД ограничивается контроллером. К примеру, СХД суперкомпьютера «СКИФ-Аврора» использует контроллер, подключенный к сети суперкомпьютера по сети Infiniband и к полке хранения по интерфейсу SAS 6 Гбит/с. Таким образом, производительность сети Infiniband (40 Гбит/с) не используется полностью, т.к. контроллер не может записать такое количество данных на полку хранения за один момент времени и вынужден использовать очередь для данных.

Технология параллельной системы хранения данных позволяет увеличить скорость обмена данных между суперкомпьютером и СХД за счет использования большого числа параллельных операций чтения/записи.

2. Описание технологии

Параллельная система хранения данных состоит из следующих компонентов (рис. 1):

- FS – клиент, установленный на узле суперкомпьютера, служащий для соединения с СХД;
- Director – узел СХД, служащий для связи клиентов с СХД;
- Storage – узлы хранения, на которых хранятся блоки данных;
- СУБД – параллельная СУБД (MongoDB [6]), в которой хранятся метаданные и атрибуты файлов.

Принцип работы состоит в следующем: модуль *FS* соединяется с узлом *Director* и передает команды для чтения/записи и работы с атрибутами файлов. При записи блока *FS* посылает команду WriteBlock() на узел *Director*. Он выбирает незанятый узел *Storage* и передает *FS* номер выбранного узла, на который передается блок данных. Узел *Storage*, на который поступили данные, разбивает блок на сегменты и по высокопроизводительной сети передает их на узлы *Storage*, выбираемые по определенным критериям. После получения сегментов узлы *Storage* записывают метаинформацию и хеш сегмента о них в *СУБД*.

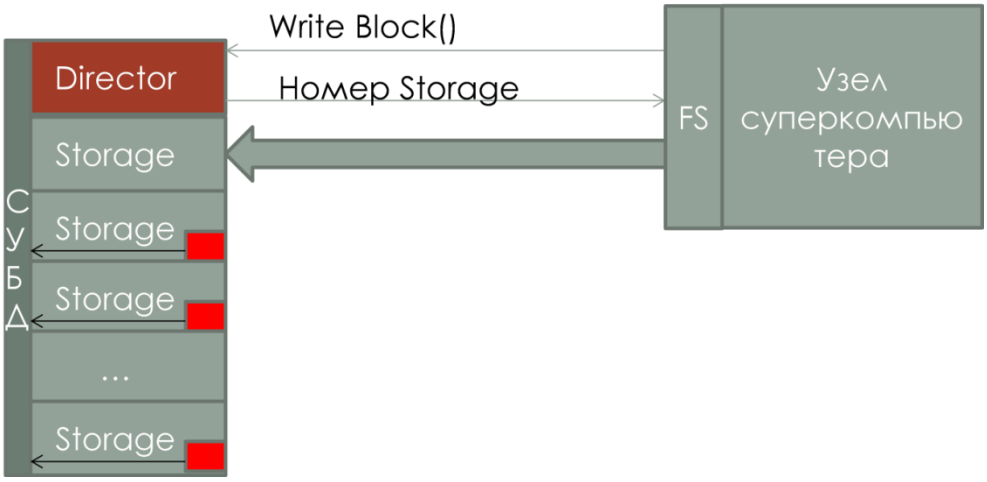


Рис. 1. Архитектура параллельной системы хранения данных

При операции чтения данных FS посылает команду ReadBlock() узлу Director. Он запрашивает информацию о сегментах блока в СУБД и посылает команды узлам Storage отправить данные сегменты в FS. Далее Director посылает контрольную сумму сегментов на FS для проверки целостности переданных данных.

Предложенная технология позволяет устранить точку отказа сервера метаданных за счет использования параллельной СУБД, что повышает надежность системы (каждый из узлов системы может автоматически выступить в роли Director, если основной узел выйдет из строя). Также повышается скорость работы с атрибутами, что важно при работе большого количества пользователей. Использование хешей позволяет обеспечить блочную дедупликацию данных за счет удаления сегментов, хеши которых совпадают. Также технология позволяет создавать моментальные снимки (снапшоты) за счет разрешения метаинформации о сегментах.

Литература

1. TOP500 List – November 2011 (1-100) – [http://top500.org/list/2011/11/100].
2. Panasas parallel storage: hpc benefit for computer aided engineering applications – [http://www.panasas.com/sites/default/files/uploads/docs/panasas_parallelstoragecae_sb_lr_1016.pdf].
3. Welch B., Unangst M. Clustered and Parallel Storage System Technologies // 7th USENIX Conference on File and Storage Technologies (FAST '09).
4. Сайт ФС lustre – [http://wiki.lustre.org/index.php/Main_Page].
5. Сайт проекта ОМЕГА – [http://omega.susu.ru/].
6. Сайт СУБД MongoDB – [http://www.mongodb.org/].
7. Российский рынок систем хранения данных за последний год увеличился наполовину – [http://expert.ru/expert/2012/20/kamera-hraneniya/].

ПРИМЕНЕНИЕ МНОГОСЕТОЧНЫХ МЕТОДОВ ДЛЯ РЕШЕНИЯ ЭЛЛИПТИЧЕСКИХ ЗАДАЧ С ПОМОЩЬЮ ПАКЕТА OPENFOAM

Д.В. Быстриков

Кемеровский госуниверситет

Представлены результаты экспериментального исследования применения многосеточных методов для решения эллиптических задач с помощью пакета OpenFOAM. В качестве задачи, при решении которой исследуется применимость многосеточных методов, выбрана задача обрушения столба жидкости в прямоугольном резервуаре без перегородки. При решении подобных задач используется метод объема жидкости, реализованный в решателе interFoam пакета OpenFOAM.

Введение

По мере того как суперкомпьютеры становятся все более популярным инструментом для решения задач в различных областях науки, у исследователей появляется возможность решать ресурсоемкие задачи с большей точностью. При решении задач гидродинамики очень важным является выбор метода решения систем линейных алгебраических уравнений (СЛАУ). Для больших задач время решения этих систем может покрывать существенную долю общего времени расчёта. Правильный выбор метода может значительно сократить время расчета на суперкомпьютерах.

Методы оценивались по двум критериям:

- времени счета;
- эффективности распараллеливания в системах с общей и распределенной памятью.

OpenFOAM (англ. Open Source Field Operation And Manipulation CFD ToolBox) – открытая интегрируемая платформа для численного моделирования задач механики сплошных сред. Большим преимуществом пакета OpenFOAM является то, что использование данного пакета на суперкомпьютерах не требует от пользователя больших усилий. Это позволяет пользователям решать все более сложные и ресурсоемкие задачи. Запуск OpenFOAM на нескольких процессорах заключается в разбиении расчетной области на части, каждая из которой решается отдельным процессором. Разбиение производится четырьмя различными способами:

- метод simple. Область разбивается на части в направлениях x, y и z. Количество областей в каждом направлении задается пользователем;
- метод hierarchical. Отличается от метода simple тем, что пользователь задает порядок разбиения, например сначала направление y, потом x, потом z;
- метод scotch. Пользователь указывает правило, по которому разбивается сетка, и указывает вес каждого процессора. Очень полезный метод, если задача решается на процессорах с разной производительностью;
- метод manual. Ручное разбиение, в котором пользователь явно задает расположение каждой подобласти для определенного процессора.

Во многих задачах для решения системы уравнений Навье-Стокса используются методы, в которых для нахождения давления необходимо решать уравнение Пуассона. Как правило, соответствующая матрица линейных алгебраических уравнений является симметричной и положительно определенной. Весьма эффективными для решения СЛАУ являются многосеточные методы.

Многосеточный метод – метод решения системы линейных алгебраических уравнений, основанный на использовании последовательности уменьшающихся сеток и операторов перехода от одной сетки к другой. Многосеточные методы эффективны для задач эллиптического и параболического типов. В работах [1,2] теоретически обосновывается применимость многосеточных методов для симметричных и положительно определенных матриц. Многосеточные методы являются оптимальными относительно числа операций, затрачиваемых на получение решения с заданной точностью. Важной особенностью является то, что, обладая высокой эффективностью, многосеточные методы допускают наиболее естественное распараллеливание, что позволяет отнести данные методы к перспективному разделу современных высокопроизводительных алгоритмов.

В качестве методов решения СЛАУ использовались следующие:

- метод сопряженных градиентов с предобуславливанием типа неполной факторизации Холецкого (DICPCG);
- геометро-алгебраический многосеточный метод (GAMG) с предобуславливанием типа неполной факторизации Холецкого. В качестве сглаживателя выбран метод Гаусса-Зейделя;
- метод сопряженных градиентов с геометро-алгебраическим многосеточным предобуславливателем (GAMGPCG).

В качестве многосеточного цикла использовался V-цикл. Идея данного подхода заключается в том, что сначала ищется решение на подробной сетке, затем проводится последовательно вычисление невязок на все более грубых сетках, затем – несколько коррекций с грубой сетки и сглаживающие итерации на все более и более подробных сетках.

Результаты

Первый этап экспериментального исследования заключался в проведении расчетов на одном узле вычислительного кластера (в системе с общей памятью). Исследование проводилось на различном числе процессоров (1, 2, 4 и 8) и сетке, состоящей из различного числа ячеек (от 10 тысяч ячеек до 100 тысяч ячеек с шагом 10 тысяч). Каждый расчет был произведен несколько раз, представленные результаты являются средним арифметическим проведенных экспериментов.

Использование различных методов разбиения области показало, что методы simple и hierarchical показывают лучшие результаты по времени, чем метод scotch. Это связано с тем, что, во-первых, геометрия области простая, а во-вторых, производительность каждого процессора одинаковая. На рис. 1 представлены результаты расчетов на 1 процессоре с использованием различных методов решения СЛАУ. На всех представленных рисунках по оси абсцисс – количество ячеек сетки, по оси ординат – время расчета.

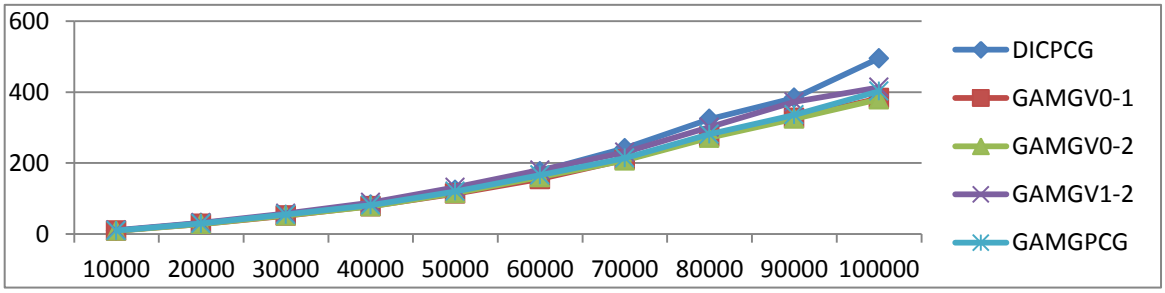


Рис. 1. Результаты расчета на 1 процессоре

Из данных, представленных на рис. 1, видно, что методы GAMG с предобуславливанием типа неполной факторизации Холецкого с V-циклом 0-1 и V-циклом 0-2 по-

казывают лучшие результаты по скорости и позволяют сократить время расчета на 20% при достаточно подробной сетке по сравнению с методом DICPCG.

На рис. 2 представлены результаты расчетов на 2 процессорах с использованием различных методов решения СЛАУ.

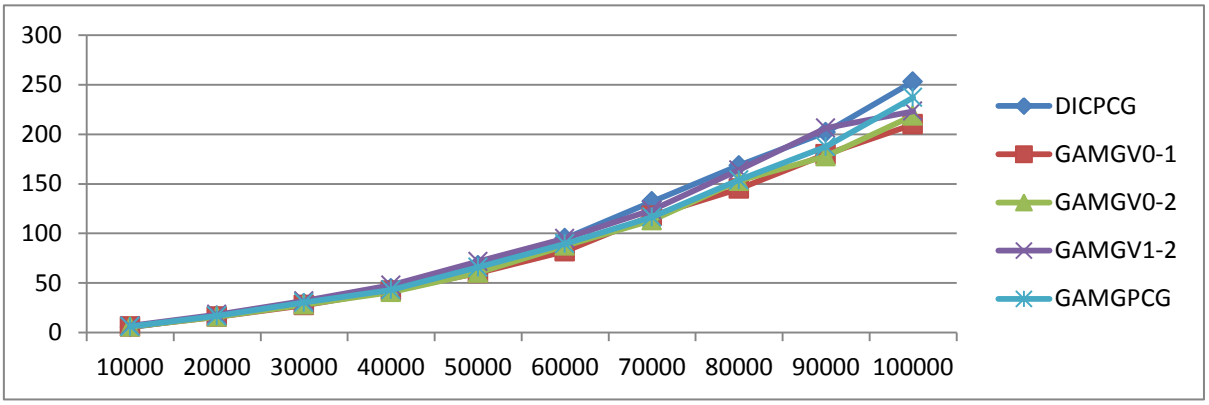


Рис. 2. Результаты расчета на 2 процессорах

Из результатов, представленных на рис. 2, видно, что те же методы показывают наилучшие результаты по времени расчета. Стоит отметить, что все алгоритмы распараллеливания, используемые в пакете OpenFOAM, показали высокую эффективность.

Результаты расчета ускорения и эффективности алгоритмов распараллеливания при расчете на 2 процессорах представлены в таблице 1. В таблицах 1-3 по оси ординат отложено количество ячеек сетки, S – ускорение, E – эффективность.

Таблица 1. Ускорение и эффективность расчетов на 2 процессорах

	DICPCG		**GAMGV0-1**		**GAMGV0-2**		**GAMGV1-2**		**GAMGPCG**	
	S	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**
10000	1,704	85,2	1,628	81,4	1,58	79,4	1,624	81,2	1,604	80,2
20000	1,809	90,5	1,789	89,5	1,779	88,9	1,753	87,7	1,75	87,5
30000	1,824	91,2	1,873	93,6	1,849	92,5	1,815	90,7	1,815	90,7
40000	1,85	92,5	1,853	92,6	1,903	95,2	1,861	93,1	1,887	94,3
50000	1,832	91,6	1,898	94,9	1,913	95,6	1,832	91,6	1,827	91,4
60000	1,859	92,7	1,889	94,5	1,831	91,6	1,899	95	1,858	93
70000	1,829	91,5	1,779	88,9	1,828	91,4	1,863	93,2	1,839	92
80000	1,922	96,1	1,891	94,6	1,773	88,7	1,837	92	1,828	91,4
90000	1,9	95	1,821	91	1,829	91,4	1,807	90,4	1,792	89,6
100000	1,957	97,86	1,824	91,2	1,736	86,8	1,852	92,6	1,697	84,9

Анализ табл. 1 показывает, что параллельные версии многосеточных методов дают меньшее ускорение, чем метод DICPCG, по сравнению со своими последовательными аналогами, однако время расчета по-прежнему на 20% меньше, чем при использовании метода DICPCG.

На рис. 3 представлены результаты расчета на 4 процессорах с использованием различных методов решения СЛАУ.

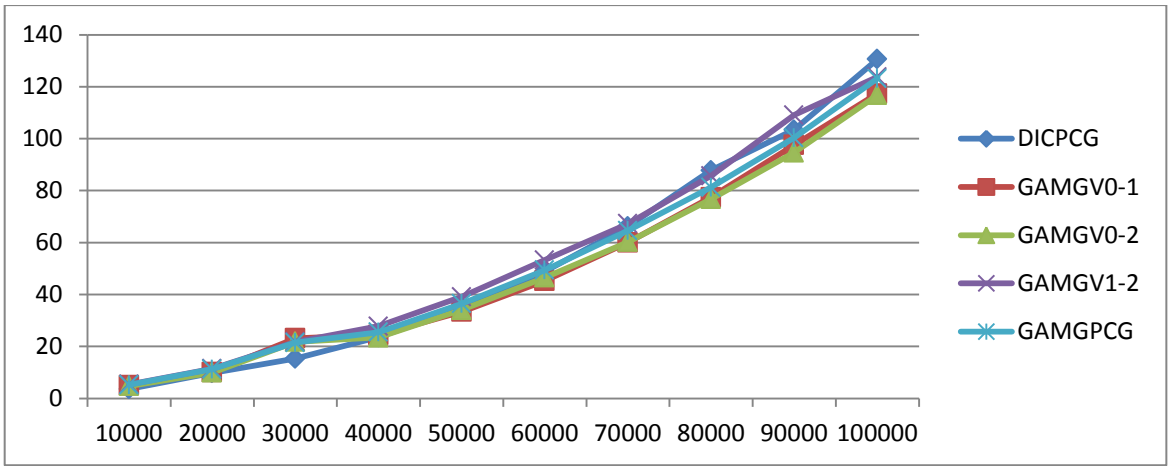


Рис. 3. Результаты расчета на 4 процессорах

Тенденция временных затрат используемых методов аналогична случаю использования 1, 2 процессоров. Кроме того, можно заметить, что при размерности сетки 30000 ячеек наблюдается скачок у всех многосеточных методов. Данное явление объяснить затруднительно. Стоит отметить, что все алгоритмы распараллеливания, используемые в пакете OpenFOAM, показали эффективность ниже, чем при расчете на 2 процессорах. Это объясняется уменьшением счета на каждом процессоре и увеличением коммуникаций в параллельной программе. Результаты расчета ускорения и эффективности алгоритмов распараллеливания при расчете на 4 процессорах представлены в табл. 2.

Таблица 2. Ускорение и эффективность расчетов на 4 процессорах

	DICPCG		**GAMGV0-1**		**GAMGV0-2**		**GAMGV1-2**		**GAMGPCG**	
	S	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**
10000	2,581	64,5	1,934	48,4	2	50	2,051	51,3	1,883	47,1
20000	3,163	79,1	2,805	70,1	2,871	71,8	2,798	69,9	2,626	65,7
30000	3,507	87,7	2,232	55,8	2,405	60,1	2,672	66,8	2,566	64,2
40000	3,491	87,3	3,207	80,2	3,317	82,9	3,204	80,1	3,189	79,7
50000	3,573	89,3	3,415	85,4	3,389	84,7	3,360	84	3,302	82,5
60000	3,62	90,5	3,423	85,6	3,443	86,1	3,382	84,5	3,368	84,2
70000	3,649	91,2	3,488	87,2	3,44	86	3,424	85,6	3,315	82,9
80000	3,69	92,2	3,544	88,6	3,535	88,4	3,515	87,9	3,463	86,6
90000	3,71	92,7	3,353	83,8	3,433	85,8	3,414	85,4	3,347	83,7
100000	3,79	94,8	3,262	81,5	3,251	81,3	3,339	83,5	3,268	81,7

На рис. 4 представлены результаты расчета на 8 процессорах с использованием различных методов решения СЛАУ.

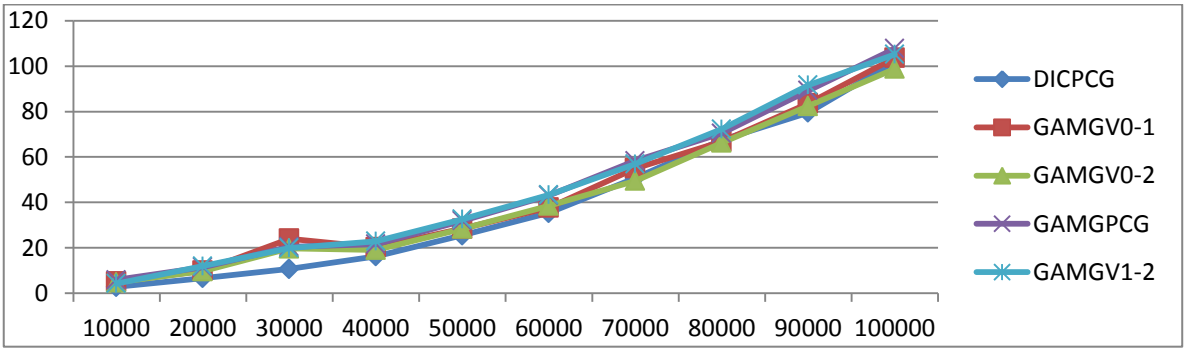


Рис. 4. Результаты расчета на 8 процессорах

Результаты, представленные на рис. 4, показывают, что время расчета на 8 процессорах примерно равно времени расчета на 4 процессорах, что говорит о плохой масштабируемости параллельных алгоритмов. Следовательно, увеличивать число процессоров не имеет смысла, так как это может привести даже к увеличению времени расчета. Кроме того, у всех многосеточных методов снова наблюдается скачок времени расчета при размерности 30000 ячеек.

Результаты расчета ускорения и эффективности алгоритмов распараллеливания при расчете на 8 процессорах представлены в таблице 3.

Таблица 3. Ускорение и эффективность расчетов на 8 процессорах

	DICPCG		**GAMGV0-1**		**GAMGV0-2**		**GAMGV1-2**		**GAMGPCG**	
	S	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**	**S**	**E(%)**
10000	3,34	41,7	1,915	23,9	2	25	2,583	32,3	1,663	20,8
20000	4,632	57,9	2,89	36,1	2,939	36,7	2,674	33,4	2,578	32,2
30000	5,048	63,1	2,149	26,9	2,658	33,2	2,922	36,5	2,695	33,7
40000	5,062	63,3	3,877	48,5	4,077	50,9	3,891	48,6	3,766	47,1
50000	4,875	60,9	4,008	50,1	4,065	50,8	4,028	50,3	3,771	47,1
60000	4,997	62,5	4,111	51,4	4,163	52,1	4,163	52,1	3,87	48,4
70000	4,764	59,5	3,809	47,6	4,187	52,3	4,047	50,1	3,672	45,9
80000	4,796	59,9	4,103	51,3	4,093	51,1	4,167	52,1	3,993	49,9
90000	4,825	60,3	3,912	48,9	3,944	49,3	4,064	50,8	3,763	47
100000	4,858	60,7	3,693	46,2	3,84	48	3,929	49,1	3,735	46,7

Расчеты проводились на кластере МСЦ РАН.

Результаты показали, что использование многосеточных методов может существенно сократить время расчета в системах с общей памятью.

В настоящее время проводятся эксперименты для вычисления эффективности распараллеливания методов решения СЛАУ при более подробном разбиении расчетной области (и соответственно большем числе ячеек) в системах с общей и распределенной памятью.

Литература

1. Федоренко Р.П. Введение в вычислительную физику: учеб. пособие. М.: Издательство Московского физико-технического института, 1984. – 526 с.
2. Федоренко Р.П. Релаксационный метод решения разностных эллиптических уравнений // Журнал вычислительной математики и математической физики.– 1961. – 1(5). С. 922-927.

ИСПОЛЬЗОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ С КРИТЕРИЕМ ДЖЕФРИСА-МАЦУСИТЫ ДЛЯ ВЫБОРА ОПТИМАЛЬНОГО НАБОРА ВХОДНЫХ КАНАЛОВ В ЗАДАЧЕ ДЕШИФРАЦИИ НАЗЕМНЫХ ОБЪЕКТОВ ПО ДАННЫМ ДЗЗ

А.Ф. Валиуллин

Казанский национальный исследовательский технический университет им. А.Н. Туполева

Введение

В современном мире с ростом производства и деятельности человека растет степень воздействия на окружающую среду. Когда происходят техногенные аварии или природные катаклизмы, необходимо незамедлительно получить данные для оценки ситуации. Эти данные необходимы для составления мероприятий, направленных на ликвидацию последствий.

Для получения данных и контроля огромных площадей земной поверхности наиболее эффективными являются методы ДЗЗ, когда по каким-либо причинам наблюдение непосредственно с поверхности Земли не обосновано с экономической точки зрения либо невозможно ввиду ряда причин.

Данные дистанционного зондирования Земли (ДЗЗ), получаемые с оптико-электронных космических аппаратов ДЗЗ, обычно представляют собой цифровые изображения излучения, зарегистрированного в различных спектральных диапазонах. Это могут быть данные как видимого спектра, так и инфракрасного и/или микроволнового диапазонов излучения. Получению, обработке и анализу подобных изображений посвящена обширная литература [1, 2, 3]. Наиболее распространенными сенсорами на данный момент являются различные мультиспектральные спектрорадиометры, регистрирующие излучение в нескольких (обычно до десяти) спектральных диапазонах (каналах).

При автоматизированной дешифрации таких данных (т. е. построении тематических карт на основе данных ДЗЗ) хорошо зарекомендовали себя различные параметрические и непараметрические методы статистики [1], а также другие методы обработки данных, например нейронные сети (обзор литературы по применению нейронных сетей при анализе данных ДЗЗ представлен в [3]).

В настоящее время начинают получать распространение гиперспектральные спектрорадиометры, которые позволяют регистрировать от нескольких десятков до нескольких сотен каналов. Например, общедоступные данные EO1/Hyperion [4] представляют собой сцены, состоящие из 220 спектральных полос от 0.4 до 2.5 мкм, со спектральным разрешением от 0.1 до 0.11 мкм. Гиперспектральные спектрорадиометры призваны увеличить точность и надежность ДЗЗ благодаря съемке в широком радиометрическом разрешении.

Основная проблема обработки гиперспектральной информации заключается в том, что классические методы анализа данных ДЗЗ разрабатывались для применения к мультиспектральным снимкам, т. е. для работы с данными относительно малой размерности. Поэтому использование этих методов для обработки данных существенно большей размерности сопряжено с определенными трудностями. В первую очередь это свя-

зано со сложностью точной оценки параметров моделей на основе обучающих данных. Например, количество обучающих примеров, необходимых при оценке параметров для метода максимального правдоподобия (в предположении о нормальном законе распределения данных), растет как квадрат размерности обрабатываемых данных, а в случае использования непараметрических методов между требуемым объемом обучающей выборки и размерностью данных возникает экспоненциальная зависимость [1].

1.Описание работы

Для решения проблемы высокой размерности данных необходима методика, дающая рекомендацию о выборе наилучшей комбинации входных каналов для уверенной дешифрации космических снимков. В работе [5] была описана методика, в которой использовалась имитационная модель, предсказывающая средний уровень излучения для интересующего объекта. Однако в примере (см. рис. 1) видно, что два различных класса объектов могут иметь равные средние. В данном случае модель даст низкую оценку возможности дешифрации на данном наборе исходных каналов при наличии практической возможности разделимости.

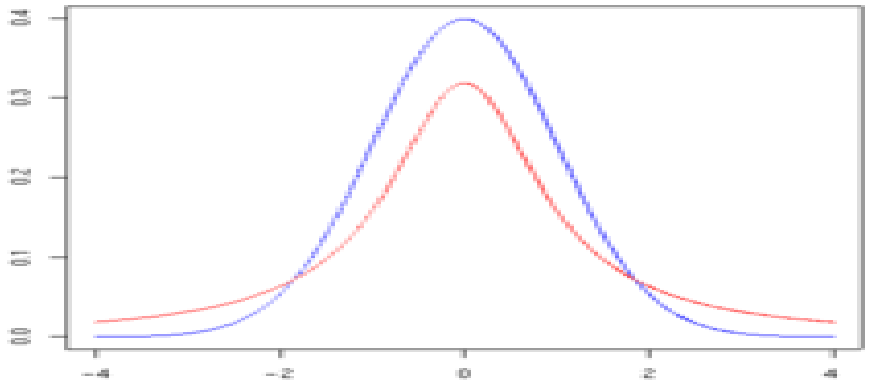


Рис. 1. График плотности распределения двух объектов

Для разрешения этого недостатка необходима оценка, которая будет учитывать расстояние между плотностями вероятности.

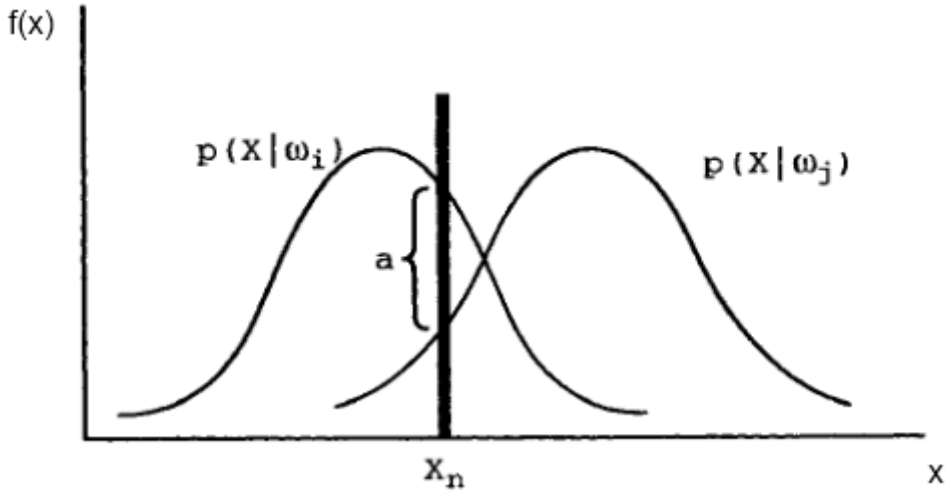


Рис. 2. Геометрический смысл учета расстояния между графиками плотности распределения вероятности

В работе будут использоваться генетические алгоритмы [7]. В качестве фитнес-функции будет выступать оценка Джефриса-Мацуситы [1], которая максимизируется в процессе поиска. На рис. 2 изображена идея использования расстояния между плотностями распределения для получения более объективной оценки комбинаций входных каналов. Генетический алгоритм будет использоваться в качестве механизма эвристического поиска набора входных каналов с максимальным значением оценки разделимости. В качестве исходных данных будут установлены пределы доступных каналов, затем будут рассматриваться закодированные последовательности номеров каналов. Критерием качества найденных решений будет служить расстояние Джефриса-Мацуситы, которое будет рассчитываться для каждого найденного решения. Для проверки адекватности найденной комбинации каналов на них будет обучена нейронная сеть с топологией 2-5-2-1. Критерием качества обучения нейронной сети будет ошибка на тестовом и валидационном множестве.

2. Постановка задачи

В работе необходимо найти оптимальный набор входных каналов для наиболее уверенной дешифрации снимков нейронной сетью. Входные каналы получены со спутника MODIS [6] части заповедника «Черные земли», находящиеся в Республике Калмыкия. В наличии 22 канала.

В работе рассматривается задача дешифрации областей, пострадавших от пожара. Также обращается внимание на довольно схожий класс «тени от облаков», т.к. на графике плотности распределения два этих класса имеют примерно равные средние значения. Математическая форма показателя разделимости классов [1] указана ниже:

$$J = \sqrt{\int_{-\infty}^{\infty} \left\{ \sqrt{g(x)} - \sqrt{f(x)} \right\}^2 dx},$$

где $g(x)$ и $f(x)$ – функции распределения двух классов.

Однако ввиду отсутствия $g(x)$ и $f(x)$ в работе будет использоваться формула для расчета оценки при условии нормального распределения вероятностей рассматриваемых классов:

$$J = \sqrt{2\left\{1 - \frac{\sqrt[4]{|\Sigma_f||\Sigma_g|}}{\sqrt{\frac{1}{2}|\Sigma_f + \Sigma_g|}} \exp\left\{-\frac{1}{8}\left[\mu_f - \mu_g\right]^T \left[\frac{\Sigma_f + \Sigma_g}{2}\right]^{-1} \left[\mu_f - \mu_g\right]\right\}\right\}},$$

где $\sum$ – ковариационная матрица, μ – средние.

3. Результаты

Критерий Джефриса-Мацуситы показал свою эффективность в условиях данной задачи. На рис. 3 визуально видно, что более разнесенные по оси ординат распределения получили более высокую оценку (слева оценка 1.22, справа 0.76). Слева отображены кривые плотности распределения для 8-го и 9-го каналов, где зеленая кривая – плотность распределения сгоревшей территории 9-го канала, синяя кривая – плотность сгоревшей территории 8-го канала, серая кривая – тень от облаков 8-го канала и красная – тень от облаков 9-го канала соответственно. Таким образом получается сравнение двух кривых одного класса обоих каналов с двумя кривыми второго класса обоих каналов.

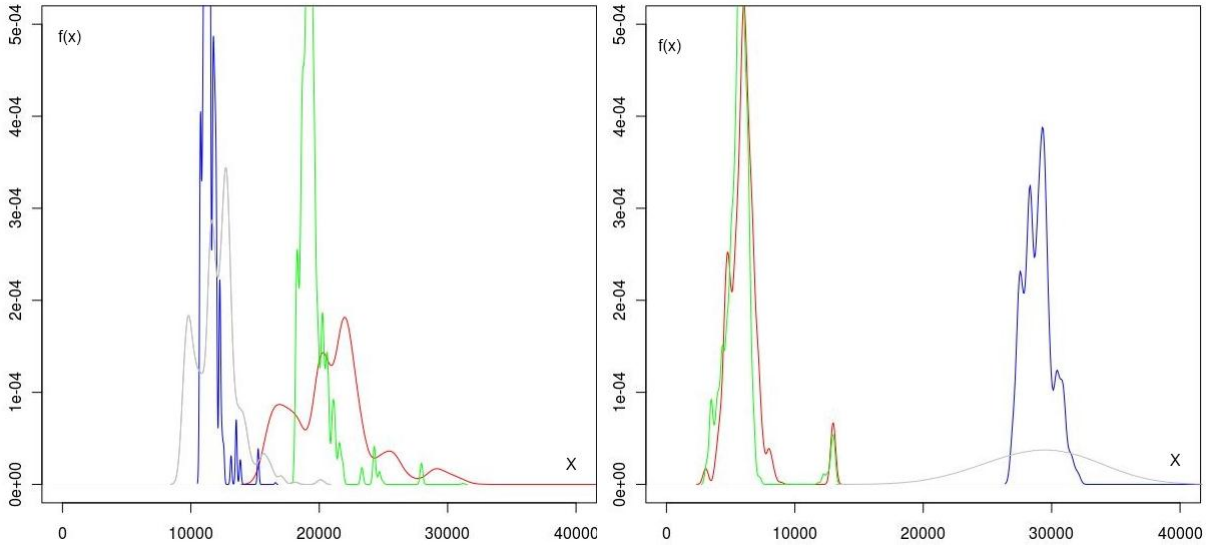


Рис. 3. Графики плотности распределения: зеленый и синий – плотности распределения класса «сгоревшая площадь», красный и серый – «тени от облаков»

Таблица 1. Результаты обучения нейронной сети

1 канал	2 канал	J	Train	Val
8	**9**	**1.221064**	**1.6915883488419e-06**	**2.42829360273056e-06**
8	10	1.018898	1.69167854095155e-06	0.000881937217142461
1	2	0.9509699	0.012281471583721	0.0161234563422114
5	11	0.7674143	0.025140684272252	0.0218377315961682

Выделенная строка в таблице результатов – это каналы, рекомендуемые генетическим алгоритмом. Видно, что их выбор дает наиболее низкую ошибку как на тестовом, так и на обучающем множестве. Также видно, что при снижении оценки *J* падает и качество обучения. Выбор оптимальных каналов занял 2.5 минуты.

Результаты показали, что использование генетических алгоритмов с критерием Джефриса-Мацуситы дает хорошую рекомендацию при небольших временных затратах.

Литература

1. Landgrebe D.A. Signal theory methods in multispectral remote sensing. Wiley-Interscience, 2003. 508 p.
2. Richards J.A., Jia X. Remote sensing digital image analysis: an introduction. Springer Verlag, 2006. 454 p.
3. Шовенгердт Р.А. Дистанционное зондирование. Модели и методы обработки изображений. М.: Техносфера, 2010. 582 с.
4. Earth Observing 1 (EO-1) Hyperion – [http://edcsns17.cr.usgs.gov/eo1/sensors/hyperion].
5. Валиуллин А.Ф. Имитационная модель системы излучение–датчик для спектрорадиометров дистанционного зондирования Земли // Туполевские чтения XX (принятно в печать). Казань, КНИТУ-КАИ.
6. Modis – [http://modis.gsfc.nasa.gov/].
7. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. 2006. 385 с.

МОДЕЛИРОВАНИЕ ДИНАМИКИ ВЗАИМОДЕЙСТВУЮЩИХ КВАНТОВЫХ ЧАСТИЦ В ПРОСТРАНСТВЕННО-НЕОДНОРОДНЫХ РЕШЕТКАХ

А.В. Вильдеманов[1], М.В. Иванченко[1], Т.В. Лаптева[2]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского

[2]Max Planck Institute for the Physics of Complex Systems, Dresden, Germany

Введение

Локализация колебаний и распространение волн в зависимости от характера пространственных неоднородностей среды – одно из фундаментальных явлений волновой физики. С 50-х годов XX века известен феномен андерсоновской локализации, заключающийся в том, что пространственная неоднородность в *линейном* периодическом потенциале приводит к локализации волн, экспоненциально слабому транспорту (электрического заряда, механических колебаний). Проблема андерсоновской локализации в *нелинейных* системах, системах с *взаимодействием* между квантовыми частицами до сих пор оставалась открытой.

Развитая нами теория нелинейной андерсоновской локализации предсказывает, что для разрушения андерсоновской локализации достаточно взаимодействия между двумя частицами, а в классическом (многочастичном) пределе локализация приобретает вероятностные свойства [1,2]. Ее верификация в численных экспериментах представляет собой исключительно сложную задачу, поскольку достоверное измерение делокализации волновых пакетов и характеристик диффузии требует численного интегрирования больших систем дифференциальных уравнений (на решетке как минимум $(5x10^3)x(5x10^3)$) на больших временных интервалах (как минимум 10^6 шагов интегрирования) для большого набора реализаций пространственной неоднородности (порядка 100 реализаций). На современных вычислительных ресурсах типичное время работы последовательной программы для одной такой реализации составляет несколько недель.

Применение технологий параллельного программирования позволило нам существенно продвинуться вперед в решении этой проблемы. Поскольку задача относится к классу задач SIMD (single instruction multi data), здесь целесообразно использовать графический ускоритель. Дополнительным плюсом использования GPU является то, что на одном вычислительном узле, как правило, находится несколько графических карт, которые можно использовать параллельно. В качестве технологии программирования была выбрана CUDA, а схема численного интегрирования – из класса симплектических методов (pq-метод [3]). Для двухчастичной задачи на одномерной решетке пространство данных представляет собой двумерный симметричный массив значений амплитуд вероятности волновой функции (так как частицы, в силу физической постановки задачи, обладают бозе-статистикой и неразличимы). Для вычисления функции-интегратора можно использовать ядро с двумерной сеткой нитей, что позволяет обработать все элементы массива. Поскольку взаимодействие между частицами локально, в CUDA ядре можно ограничиться использованием только глобальной памяти. В результате длительность выполнения вычислительной задачи с теми же параметрами, что и в последовательной версии, на одной графической карте Tesla M2070 удалось снизить до 1 дня.

Работа выполнена при поддержке фонда «Династия», грантов Президента РФ МК-4028.2012.2 и РФФИ 12-02-31403, а также в сотрудничестве с Институтом Макса Планка физики сложных систем, Дрезден.

Литература

1. Flach S., Ivanchenko M.V., Khomeriki R. Correlated metallic two-particle bound states in quasiperiodic chains // Europhysics Letters. 98. 66002 (2012).
2. Ivanchenko M.V., Laptyeva T.V., Flach S. Anderson localization or nonlinear waves: A matter of probability // Physical Review Letters. 107. 240602 (2011).
3. Skokos Ch., Krimer D.O., Flach S., Komineas S. Delocalization of wave packets in disordered nonlinear chains // Phys. Rev. E 79. 056211 (2009).

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МНОГОКРИТЕРИАЛЬНОГО МЕТОДА ПОСТРОЕНИЯ КАРТ ДИСПАРАТНОСТИ

А.Н. Волкович

Объединенный институт проблем информатики НАН Беларуси

Рассматриваются проблемы увеличения производительности и качества построения карт диспаратности в задачах восстановления трехмерных моделей. Описывается оригинальный многокритериальный подход к процедуре построения карты диспаратности. Рассматриваются особенности параллельной реализации, и анализируется эффективность предложенных методов.

Введение

Задача восстановления информации об объемных характеристиках объектов на основе нескольких изображений, полученных в одно время с некоторым параллаксом, является одной из фундаментальных задач в области компьютерного зрения. Попытки построения электронной системы, функционально повторяющей систему объемного зрения человека, происходят в течение многих лет. Выработана серия приемов с использованием различных вспомогательных средств (лазерные дальномеры, сканеры, системы структурированной подсветки), которые позволяют получать дополнительные характеристики для построения трехмерной модели сцены. Особый интерес вызывают подходы, основанные на получении данных в пассивном режиме (без дополнительных излучателей) и получении на их основе информации об объеме.

Построение объемной модели на основе стереоизображений традиционно было и остается одним из наиболее актуальных направлений в развитии компьютерного зрения. Последние исследования в этой области значительно продвинули область знания в вопросах качества и адекватности построений. К сожалению, на современном этапе исследований требования к производительности значительно превышают возможности элементной базы – алгоритмы стереовосстановления обычно требуют от нескольких секунд до нескольких минут машинного времени для построения единственной карты диспаратности. Однако существует значительное количество актуальных приложений, таких как задачи навигации и виртуальной реальности, которые требуют построения карт диспаратности с частотой близкой или эквивалентной стандартному видео. Кроме того, обработка больших изображений (таких как аэрофотоснимки и т.п.) существующими методами требует неприемлемо больших временных затрат.

1. Методы стереовосстановления объемных сцен

Широкие исследования в области компьютерного стереовидения были начаты в середине 90-х годов XX века. За этот период был сформирован ряд методов, включающих в себя следующие шаги: получение данных, калибровка системы, задание точек привязки и сопряженных точек, вычисление матриц камер и моделей камер, ректификация изображений, вычисление плотных карт диспаратности, триангуляция, визуализация.

Наиболее сложным и неоднозначным в решении является этап построения карт диспаратности. Сложность решения задачи поиска соответствий заключается в том, что изображения, получаемые с камер, имеют определенную длину фокуса, ГРИП и угол обзора и не имеют возможности уточнения тех или иных характеристик за счет сакка-

дических движений, аккомодации, конвергенции и перцептивности, используемых в «живых» системах. В мировой практике выработан ряд методов построения плотных карт диспаратности, которые условно делятся на локальные и глобальные. Глобальные методы позволяют получить достаточно адекватную карту диспаратности, но при этом обладают слабой распараллеливаемостью и требуют значительных затрат машинного времени, что приводит к малой приемлемости их использования в задачах, требующих оперативной обработки данных. Локальные методы, в свою очередь, обладают хорошей перерабатываемостью для их использования на параллельных вычислителях (MPI, GPGPU), что позволяет использовать их для быстрой обработки стереоизображений, однако они обладают сильной чувствительностью к однородным областям изображений, на которых результат вычисления диспаратности дает значительные ошибки поиска соответствий. Отсюда очевидно возникает задача поиска возможностей снижения числа ошибок при вычислении соответствий в локальных областях.

2. Использование цветовой метрики при поиске локальных соответствий

В мировой практике при работе с изображениями в задачах стереовосстановления обычно используется только информация о яркости как критерии сравнения точек изображений. Недостатком данного подхода является множественность интерпретации цветов для точек с одинаковым значением яркости. Кроме того, следует учитывать факт неравномерности восприятия цветного и монохромного изображения. Данная особенность учитывается в методах деградации цветовой модели изображения к 256 оттенкам серого за счет введения коэффициентов, применяемых к соответствующим каналам.

Учитывая то, что большинство изображений изначально формировались цветным датчиком в цветном виде, в целях повышения эффективности авторами проекта была проведена модификация алгоритма с целью организации его работы с цветовой информацией.

При переходе к работе с тремя компонентами изображения можно представить в виде «облака» точек в трехмерном пространстве с осями, соответствующими цветовым каналам изображения. Однако RGB-пространство не является ортогональным, что обусловлено спецификой зрительного анализатора, имеющего различное число палочек и колбочек, восприимчивых к тому или иному цвету.

Поскольку мерой сравнения точек в трехмерном пространстве выступает евклидово расстояние, которое применимо к ортогональным системам, следует производить ортогонализации пространства RGB в пространство XYZ. Представление базисных цветов RGB, согласно рекомендациям ITU, в пространстве XYZ:

$$\text{Красный}: x = 0{,}64 \;\; y = 0{,}33;$$

$$\text{Зеленый}: x = 0{,}29 \;\; y = 0{,}60;$$

$$\text{Синий}: x = 0{,}15 \;\; y = 0{,}06.$$

Таким образом, систему преобразований для перевода цветов между системами RGB и XYZ можно представить в виде:

$$X = 0{,}431 * R + 0{,}342 * G + 0{,}178 * B,$$

$$Y = 0{,}222 * R + 0{,}707 * G + 0{,}071 * B,$$

$$Z = 0{,}020 * R + 0{,}130 * G + 0{,}939 * B.$$

После приведения пространств к точкам могут применяться операции, справедливые для ортогональных систем. Следует отметить, что максимальное значение длины вектора в пространстве, построенном для 8-битного цветового канала (441 единица), в

1,7 раза превосходит по информативности сравнение точек в полутоновом представлении изображения, а количество уникальных значений увеличивается в 65 тысяч раз по сравнению с монохромным.

3. Использование функций градиента с целью уточнения диспаратности

В свою очередь, остается проблема обработки участков изображений, расположенных не в фокусе, объектов с большими однородными областями, а также бесконечно удаленных объектов (например, небо). Такие области изображений практически не имеют разрывов (перепадов) по яркости, которые, в свою очередь, несут максимальное количество информации, используемой при обработке изображения.

В целях классифицирования точки как находящейся на перепаде яркости изменение яркости, ассоциированное с данной точкой, должно быть существенно большим, чем изменение яркости в точке фона. Определение точки изображения как точки перепада происходит в том случае, если ее двумерная производная первого порядка превышает некоторый заданный порог. Один из способов нахождения первых частных производных в конкретной точке состоит в применении следующего градиентного оператора Собеля. В большинстве случаев, встречающихся в практике, оператор Собеля применяется к монохромному варианту изображения. Однако преобразование цветовой модели может исказить информацию, поэтому авторами предложено производить обработку оператором Собеля каждого цветового канала и их ансамбля при построении карты вычислений.

На основе полученной карты градиентов возможно получить следующие дополнительные характеристики точки: значение функции локального градиента, расстояние до ближайшего пика функции градиента, конфигурация пиков в некоторой окрестности, которые можно использовать для уточнения соответствия локальной области.

4. Параллельная реализация процедуры построения карты диспаратности

Задача построения плотной карты диспаратности как одна из задач обработки изображений по своей сути может быть справедливо отнесена к классу итерационных. В принципе, задача распараллеливания любого итерационного метода не относится к разряду сложных. В случае обработки изображений в локализованных регионах очевидна возможность разделения задачи.

Однако при адаптивной обработке комбинированным методом возникает значительное увеличение объемов вычислений в рамках выбранной итерации. Кроме того, возникает проблема неравномерности объемов вычислений, а также объемов используемых данных, вызываемая разноплановостью подходов к обработке изображения и карты градиентов. Данная задача, по своей сути, решаема загрузкой в каждый вычислительный узел полного изображения, что может быть приемлемо при обработке монохромных исходных данных, но невозможно при работе с цветными изображениями, т.к. размерность в большинстве случаев начинает превышать объемы памяти графических ускорителей.

Учитывая взаимную независимость методов поиска соответствий на основе цветовых характеристик и карты градиентов, задача была разделена на два соответствующих прохода в вычислениях. Управление вычислительными проходами производится собственным менеджером задачи на основе «карты вычислений».

«Карта вычислений» формируется предварительно и представляет собой матрицу целочисленных значений. Заполнение матрицы производится на основе анализа региона вокруг каждой точки изображения и указывает минимальный размер региона, при котором пространство вокруг точки на карте градиентов замкнуто.

В ходе работ было определено, что при значениях в ячейках менее 20 следует использовать метод поиска соответствий на основе цветовых характеристик, а при боль-

ших значениях (нахождение точки на большом однородном поле) наилучший результат достигается при поиске соответствий на основе карты градиентов.

Основываясь на данных в «карте вычислений» менеджер задачи проводит в первую очередь вычисление методом поиска соответствий при помощи анализа цветовых компонентов в регионе. Обработка выполняется параллельно в доступных узлах графического ускорителя. Следующим шагом производится дозаполнение карты диспаратности на втором проходе. С целью повышения скорости обработки и загрузки памяти устройств менеджером задачи на основе «карты вычислений» производится загрузка минимально необходимого объема исходных данных, а также пропуск участков изображений, не относящихся к соответствующему проходу.

В рамках выполнения работ по созданию системы автоматической реконструкции трехмерных сцен по нескольким изображениям была протестирована описанная реализация метода построения плотной карты диспаратности.

Для определения эффективности выполнения метода в последовательной и параллельной реализации проведен вычислительный эксперимент. В ходе проведения эксперимента сравнивалось время выполнения метода при различных подходах к программной реализации параллелизма на различных вычислительных системах.

Вычислительный эксперимент проводился на вычислительных системах со следующими конфигурациями:

- Персональный компьютер на базе процессора Intel Core 2 Duo 2.6 ГГц, 2048 Mb ОЗУ.
- Персональный компьютер на базе процессора Intel i7 3.4 ГГц, 16384 Mb ОЗУ.
- Персональный компьютер на базе процессора Intel Core 2 Duo 2.6 ГГц, 2048 Mb ОЗУ, графическая карта NVidia GeForce 9600GT с поддержкой технологии GPGPU.

В ходе эксперимента по тестированию алгоритма производилось по десять замеров времени исполнения алгоритма на центральном процессоре для каждой стереопары. В свою очередь, вычислительный эксперимент с использованием вычислений на графической карте привел к значительным (более 10%) колебаниям времени исполнения. Безусловно, это связано с тем, что абсолютные значения времени исполнения алгоритма на GPU значительно меньше и, в целом, сравнимы с погрешностью измерения. Для того чтобы получить наиболее адекватные средние данные выполнения алгоритма на графическом ускорителе, измерения проводились двадцать пять раз для каждой стереопары.

Средние результаты времени выполнения вычислений карт диспаратности представлены в табл 1.

Таблица 1. Средняя скорость выполнения алгоритма (секунды)

Стереоизображения	**Intel Core Duo 2.6 GHz Dual Mode**	**Intel i7 3.4 GHz**	**GPGPU**
Изображения местности	6,91	4,87	0,016
Тестовое изображение	1,64	1,64	0,009
Макроснимок	12,61	8,46	0,027
Город	10,45	7,13	0,024
Цветок	10,82	7,52	0,030
Интерьер	11,61	8,04	0,026

Таким образом, можно говорить о том, что параллельная GPGPU-реализация алгоритма позволяет увеличить его производительность в зависимости от сложности и размерности изображений на несколько порядков по сравнению с их последовательной реализацией.

Заключение

Исследования в области автоматического построения 3D-моделей объектов реального мира привели к необходимости поиска средств улучшения качества построения карт диспаратности. В ходе исследований был выполнен переход к трехканальной метрике, используемой для формирования поля соответствий, а также к поиску соответствия при помощи карт градиентов в крупных однородных областях в задаче построения карт диспаратности.

Совокупность предложенных подходов, вместе с фактической реализацией методики на персональном компьютере с помощью GPGPU-средств, может способствовать дальнейшему развитию исследований в области улучшения качества стереообработки в задачах компьютерного зрения реального или близкого к реальному времени.

Литература

1. Zach Ch., Gallup D., Frahm J.-M. Fast Gain-Adaptive KLT Tracking on the GPU. University of North Carolina Chapel Hill, NC, 2008. 7 p.
2. Волкович А.Н., Жук Д.В., Тузиков А.В. Методы построения трехмерных моделей местности и их реализация для параллельных систем // Доклады 5-й Международной конференции «Обработка информации и управление в чрезвычайных и экстремальных ситуациях», 24 – 26 октября, Минск, Беларусь, 2006. С. 100-104.
3. Ляховский В.В., Волкович А.Н., Жук Д.В., Тузиков А.В. Система автоматической реконструкции трехмерных сцен по нескольким изображениям // Материалы V Белорусского космического конгресса, 25-27 октября 2011 г. Мн.: ОИПИ НАН Беларуси, 2011. Т.2. С. 129-133.

TEMPLET-СЕРВИС НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ ДЛЯ РАЗРАБОТКИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРИЛОЖЕНИЙ

С.В. Востокин, Ю.С. Артамонов, Ю.П. Назаров

Самарский государственный аэрокосмический университет

Система Templet – это веб-инструмент, призванный помочь программистам, которым требуется управление полным циклом разработки параллельного приложения, от шаблона до реализации и развёртывания.

Веб-сервис предназначен для предоставления следующих новых возможностей: (1) создания базы данных примеров и шаблонов в различных программных/аппаратных архитектурах; (2) создания новых проектов путем копирования-вставки существующих проектов или совместного использования проектов; (3) сотрудничества в области разработки параллельных приложений на базе системы контроля версий; (4) автоматической сборки и развертывания исходных программ для исполнения на суперкомпьютере; (5) совместного использования ресурсов облака или суперкомпьютера.

Сервис работает на нескольких VPS/VDS-серверах в частном облаке суперкомпьютера «Сергей Королев» Самарского государственного аэрокосмического университета. Он использует Subversion-совместимые репозитории, установленные на суперкомпьютере или в Интернете. Структура проекта с фиксированными изменениями проверяется нашим сервисом. Если структура корректна, по запросу пользователя проект может быть скомпилирован и запущен на суперкомпьютере или в частном окружении, тип которого поддерживается сервисом.

Поддерживаются проекты в модели программирования Templet [1], а также проекты с определённой структурой файлов, необходимой для унифицированной процедуры развёртывания.

В настоящее время служба работает в тестовом режиме. Она доступна с сайта http://graphplus.ssau.ru. Для развёртывания программы в частном окружении доступен тип окружения «Linux-сервер», он может быть использован в любых проектах, независимо от частного облака СГАУ и суперкомпьютера. Необходимым условием является лишь наличие доступа к частному окружению по протоколу SSH из сети Интернет. Для работы на суперкомпьютере доступен тип окружения, который может использоваться при наличии доступа к суперкомпьютеру «Сергей Королев».

Литература

1. Востокин С.В. Templet – метод процессно-ориентированного моделирования параллелизма // Программные продукты и системы. 2012. №3. С. 9–12.

ПРИБЛИЖЕННОЕ РЕШЕНИЕ НЕЛИНЕЙНЫХ ОПЕРАТОРНЫХ УРАВНЕНИЙ С ПОМОЩЬЮ ИТЕРАТИВНОГО АГРЕГИРОВАНИЯ

В.В. Гавриленко[1], А.Ф. Обшта[2], Б.А. Шувар[2], А.А. Галкин[1]

[1]Национальный транспортный университет, Киев, Украина

[2]Национальный университет «Львовская политехника», Львов, Украина

Рассматриваются проблемы разработки алгоритмов решения сложных задач, согласованные с новыми архитектурами вычислительных систем. Получены достаточные условия сходимости построенных авторами общих итерационных алгоритмов решения операторных уравнений. Утверждения не содержат ограничений по полуупорядоченности пространств, в которых рассматриваются заданные операторные уравнения. Результаты исследований могут, в частности, иметь применение при решении систем линейных алгебраических уравнений большой размерности и сложной структуры.

Введение

Разработке параллельных алгоритмов систем и программ уделяется большое внимание в связи с тем, что известные на сегодня методы построения параллельных алгоритмов для решения сложных задач на многопроцессорных вычислительных системах не позволяют построить достаточно эффективные и быстродействующие программы для решения этих задач. Основным фактором, сдерживающим моделирование сложных процессов, является недостаточная производительность компьютеров и недостаточная производительность алгоритмов и программ. Организация параллельных вычислений является сложным процессом в связи с тем, что необходимо согласовывать решения проблем, связанных с конфигурацией компьютера, эффективностью системного программного окружения, структурной эффективностью прикладной программы параллельных вычислений. Что касается конфигурации компьютера, то известно, что при выполнении ресурсоемких задач нужно обеспечивать хранение массивов данных, поступающих из многих каналов, одновременно со считыванием в многоблочных операционных устройствах специализированных процессоров, выполнять операции реорганизации и упорядочения данных в массивах. Для существующих типов памяти это очень сложная, а часто и не решаемая с приемлемыми характеристиками задача.

Одним из подходов, улучшающих характеристики компьютера, является разработанный А.А. Мельником [1] метод построения памяти с упорядоченным доступом к данным, который позволил реализовать параллельный доступ к данным в памяти и стал основой для создания специализированных процессоров на основе этой памяти с новой архитектурой и значительно лучшими по сравнению с существующими характеристиками процессоров. В связи с этим актуальными являются проблемы разработки алгоритмов, согласованных с новой архитектурой вычислительной системы. С нашей точки зрения, значительно улучшить вычислительные характеристики могут согласованные с новой архитектурой вычислительной системы алгоритмы итеративного агрегирования решения сложных задач.

В прикладной математике немало численных методов, особенно касающихся сложных задач, которые не получили строгого обоснования, хотя успешно применяются на практике. К таким методам относятся, в частности, методы итеративного агрегирования, которые изучены мало и условия сходимости которых неизвестны. Несмотря

на многочисленные теоретические исследования методов итеративного агрегирования, указанный факт не потерял актуальности [2]. Методы итеративного агрегирования формально можно описать как проекционно-итеративные методы [3]. Существенным отличием от проекционно-итеративных методов является их особенность, которая означает, что на каждом шаге итерационного процесса методы итеративного агрегирования являются одновременно алгоритмами выбора проекционных операторов. В предлагаемой работе построены и исследованы общие итерационные алгоритмы, которые охватывают многопараметрические методы итеративного агрегирования. Использована методика построения и исследования методов итеративного агрегирования для линейных и нелинейных операторных уравнений, предложенная в [4], а также методика построения «синтетических» аналогов этих методов, получаемая сочетанием идеи итеративного агрегирования с идеей проекционно-итеративных методов.

Построение алгоритма для линейного уравнения

Пусть E – банахово пространство, в котором рассматриваем уравнение

$$x = Ax + b \ \left(b \in E\right) \tag{1}$$

с линейным непрерывным оператором $A: E \to E$. Зададим оператор $P\left(P^2 = P\right)$, который проектирует элементы банахова пространства E в элементы банахова пространства E'. Обозначим $Q = I - P$ (I – тождественный оператор). Пусть заданы линейные непрерывные операторы $S: E \to E'$ и $\Lambda: E' \to E'$. Будем считать также, что выполняется равенство

$$S\left(A + \tilde{A}\right)P = \Lambda S\,. \tag{2}$$

Полагаем, что этим равенством определен линейный непрерывный оператор $\tilde{A}: E \to E$. Если речь идет об однопараметрическом случае, то вместо (2) получим

$$\left(\varphi, \left(A + \tilde{A}\right)x\right) = \lambda(\varphi, x), \tag{3}$$

где (φ, x) – значение линейного функционала $\varphi \in E^*$ на элементах $x \in E$ (E^* – банахово пространство, сопряженное с E). Если $\tilde{A}$ – нулевой оператор, то число λ и элемент φ являются собственным числом и соответствующим ему собственным элементом сопряженного с A оператора A^*. В более общем виде равенство $SAP = \Lambda S$, которое получается из (2), если $\tilde{A}$ является нулевым оператором, можно интерпретировать как обобщение спектральной задачи для оператора AP. Если при этом Λ является матрицей диагонального вида, то ее элементы являются собственными числами оператора AP.

Рассмотрим систему, составленную из уравнения (1) и вспомогательного уравнения

$$y = \Lambda y - SAQx - Sb + S\tilde{A}Px \tag{4}$$

с дополнительным неизвестным $y \in E'$. Предполагаем, что существует обратный оператор $\left(I' - \Lambda\right)^{-1}$, где I' – единичный оператор в E'. Множество пар $\{x, y\}$ элементов $x \in E$, $y \in E'$, которые удовлетворяют равенству

$$Sx + y = \theta', \tag{5}$$

где θ' – нулевой элемент в E', обозначим через ε_0. Это множество является подпространством в банаховом пространстве $E_0 = E \times E'$ с нормой, введенной с помощью формулы

$$\|x, y\|_0 = \sqrt{\|x\|_E^2 + \|y\|_{E'}^2}\,. \tag{6}$$

Используя этот способ погружения пространства E в пространство E_0, построим итерационный процесс с помощью формул

$$x^{(n+1)} = APx^{(n+1)} - AQx^{(n)} + a^{(n)}(y^{(n)} - y^{(n+1)}) + b, \tag{7}$$

$$y^{(n+1)} = \Lambda y^{(n+1)} - SAQx^{(n)} + \alpha_0^{(n)}(y^{(n)} - y^{(n+1)}) - Sb. \tag{8}$$

Здесь $a^{(n)} : E' \to E$, $\alpha_0^{(n)} : E' \to E'$ являются заданными операторами при каждом $n = 0,1,\ldots$. Для каждого $n = 0,1,\ldots$ постулируем выполнение условия

$$Sa^{(n)} + \alpha_0^{(n)} = \Lambda. \tag{9}$$

Выбор операторов, фигурирующих в (9), конкретизирует итеративный алгоритм, который описывают формулы (7), (8), при условии, что $x^{(0)}$, $y^{(0)}$ удовлетворяют равенство (5) при $x = x^{(0)}$, $y = y^{(0)}$. В случае когда речь идет об однопараметрическом методе итеративного агрегирования и выполняется равенство (3), итеративный процесс (7), (8) можно представить в виде

$$x^{(n+1)} = Ax^{(n)} + \frac{APx^{(n)}}{\left(\varphi, x^{(n)}\right)}\left(y^{(n)} - y^{(n+1)}\right) + b, \tag{10}$$

$$y^{(n+1)} = \lambda y^{(n+1)} - \left(\varphi, \tilde{A}Qx^{(n)}\right) + \frac{\left(\varphi, AQx^{(n)}\right)}{\left(\varphi, x^{(n)}\right)}\left(y^{(n)} - y^{(n+1)}\right) - \left(\varphi, b^{(0)}\right) + S\tilde{A}Px^{(n)}. \tag{11}$$

Равенство (5) в этом случае имеет вид

$$(\varphi, x) + y = 0. \tag{12}$$

Выбирая $x^{(0)}$ произвольным способом, из этого равенства при $x = x^{(0)}$ находим $y^{(0)} = y$. При этом подтверждается равенство

$$\left(\varphi, a^{(n)}\right) + \alpha_0^{(n)} = \lambda \tag{13}$$

при $n = 0$ благодаря выбору $a^{(n)}$, $\alpha_0^{(n)}$ по формулам

$$x^{(n+1)} = \frac{\left(\varphi, Ax^{(n+1)}\right)}{\left(\varphi, x^{(n)}\right)}; \qquad \alpha_0^{(n)} = \frac{\left(\varphi, AQx^{(n)}\right)}{\left(\varphi, x^{(n)}\right)}. \tag{14}$$

Очевидно, что однопараметрический алгоритм (10), (11) можно представить в виде

$$x^{(n+1)} = \frac{\left(\varphi, Ax^{(n+1)}\right)}{\left(\varphi, x^{(n)}\right)} APx^{(n)} + AQx^{(n)} + b. \tag{15}$$

Используя [4, 5], получено строгое доказательство сходимости итерационного процесса для линейного случая, то есть для приведенных выше алгоритмов (7) и (8).

Построение алгоритма для нелинейного уравнения

Исследуем один из вариантов алгоритмов итеративного агрегирования для нелинейного уравнения

$$x = Ax + Fx, \tag{16}$$

в котором линейный непрерывный оператор A и, вообще говоря, нелинейный непрерывный оператор F действуют из E в E, где E – банахово пространство.

Зададим операторы $S : E \to E'$, $\Lambda : E' \to E'$, где E' – банахово пространство, вообще говоря, не совпадает с E. Пусть оператор $\tilde{A}$ ($\tilde{A} : E \to E$) таков, что выполняется равенство

$$S(A + \tilde{A}) = \Lambda S. \tag{17}$$

Вместе с уравнением (16) будем рассматривать вспомогательное уравнение

$$y = \Lambda y + S\tilde{A}x - SFx. \tag{18}$$

Построим итерационный процесс

$$x^{(n+1)} = Ax^{(n)} + Fx^{(n)} + a^{(n)}(y^{(n)} - y^{(n+1)}), \quad (19)$$

$$y^{(n+1)} = \Lambda y^{(n+1)} + S\tilde{A}x^{(n)} - SFx^{(n)} + a_0^{(n)}(y^{(n)} - y^{(n+1)}). \quad (20)$$

Используя [4, 5], получено строгое доказательство сходимости итерационного процесса для нелинейного случая, то есть для приведенных выше алгоритмов (19) и (20).

Выводы

Результаты исследований могут, в частности, иметь применение при решении на специализированных параллельных вычислительных комплексах систем линейных алгебраических уравнений большой размерности и сложной структуры, описывающих балансовые задачи в математической экономике.

Литература

1. Мельник А.О. Архітектура комп’ютера. Луцьк: Волинська обласна друкарня, 2008. – 470 с.
2. Гробова Т.А., Стеценко В.Я. Методы итеративного агрегирования для приближенного решения линейных и нелинейных алгебраических систем и интегральных уравнений: Монография. – Ставрополь: Изд-во СГУ, 2003. – 87 с.
3. Лучка А.Ю. Проекционно-итеративные методы решения обыкновенных дифференциальных уравнений. – Киев: Наук. думка, 1980. – 264 с.
4. Шувар Б.А. О сходимости многопараметрических методов итеративного агрегирования // Вестник Львовск. политехн. ин-та. Прикладная математика. – 1989. №232. – С. 140-142.
5. Шувар Б.А., Копач М.І., Ментинський С.М., Обшта А.Ф. Двосторонні наближені методи. – Івано-Франківськ: В-во Прикарпатського національного університету ім. В.Стефаника, 2007. – 515 с.

КАЧЕСТВЕННАЯ И ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ МЕДИЦИНСКАЯ ВИЗУАЛИЗАЦИЯ МЕТОДОМ VOLUME RAY CASTING

Н.И. Гаврилов, В.Е. Турлапов

Нижегородский госуниверситет им. Н.И. Лобачевского

Графические процессоры (GPU) сегодня широко применяются в научной, и прежде всего медицинской, визуализации. Алгоритмы для GPU становятся всё более сложными по своей структуре и всё более производительными. Как следствие, визуализация становится качественной и быстрой, увеличивается объём визуализируемых данных. В ННГУ им. Н.И. Лобачевского создан комплекс программ для научной и медицинской визуализации, предоставляющий аппарат трехмерной визуализации и исследования томограмм любого типа на рабочий стол каждого врача-клинициста. В работе представлена отечественная реализация технологии 3D-реконструкции томограмм, конкурирующая с реализациями от Fovia и Anatomage. Обсуждаются детали, связанные с качеством и производительностью 3D-реконструкции, приведены результаты вычислительных экспериментов на программном комплексе. Обсуждаются перспективы применения такого комплекса в медицине.

Введение

Томография в целом, и особенно появившаяся в последние годы многосрезовая компьютерная томография (МСКТ), является методикой объемного исследования всего тела человека. Так, длительность МСКТ всего тела с толщиной среза менее 1 мм составляет около 10-15 секунд, а результатом исследования являются от нескольких сотен до нескольких тысяч изображений (http://www.presidentmed.ru/).

В 2007 году компанией Toshiba впервые представлены 320-срезовые компьютерные томографы. Они позволяют не только получать изображения, но и дают возможность наблюдать почти что «в реальном» времени физиологические процессы, происходящие в головном мозге и в сердце [1]. Поэтому томография широко применяется сегодня: как скрининговый тест; для диагностики по экстренным показаниям; как средство плановой диагностики; для контроля результатов лечения; для проведения лечебных и диагностических манипуляций, например пункция под контролем компьютерной томографии и т.д. Полученные аксиальные томограммы составляют трехмерный массив данных, позволяющий выполнить любые реконструкции изображений, в том числе многоплоскостные реформации, объемную (и, при необходимости, стерео) визуализацию, виртуальные эндоскопии, делая эту методику наглядной как для врача-диагноста, так для врача-клинициста. Трехмерная реконструкция томограмм может быть также эффективным инструментом обучения в медицинском образовании. Нельзя переоценить возможности совместного использования нескольких томограмм разного типа: рентгеновской компьютерной (КТ), магниторезонансной (МРТ), позитрон-эмиссионной (ПЭТ) и так далее – в диагностике и сопровождении лечения онкологических заболеваний в ядерной медицине.

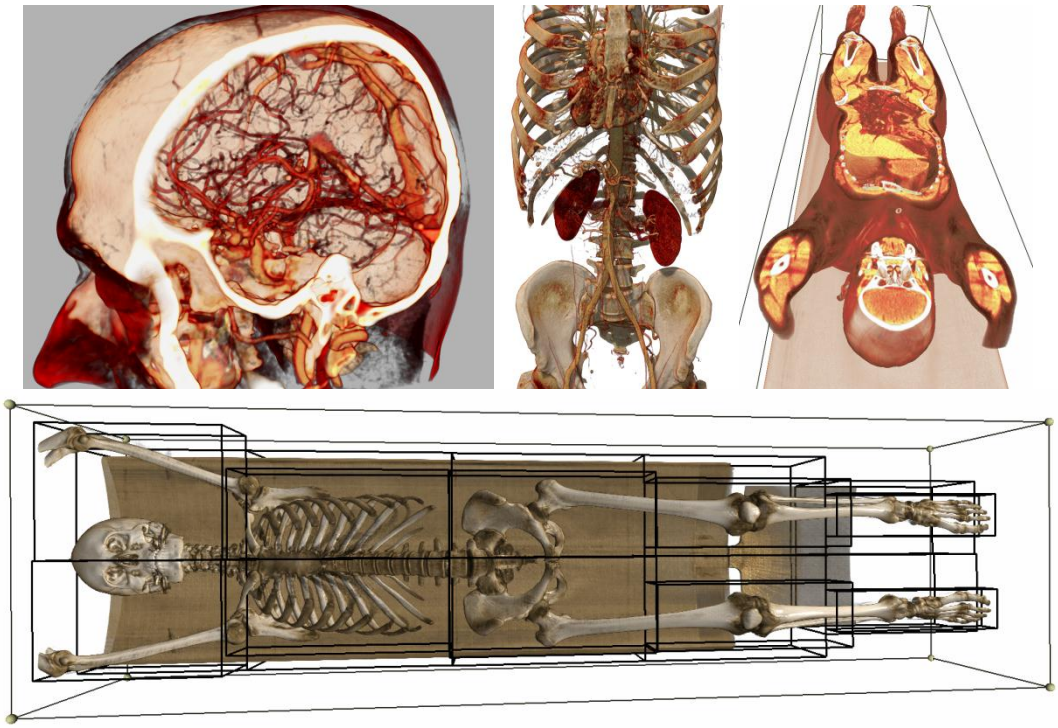


Рис. 1. Примеры трёхмерной реконструкции томограмм через «3D-визуализатор томограмм»

1. Объёмная визуализация в медицине

В направлении мировых тенденций по 3D-реконструкции томограмм в реальном времени и использованию 3D-модели пациента, как основы его медицинской карты, с 2009 года активно работает и Нижегородский госуниверситет им. Н.И. Лобачевского [5]. Наибольших успехов в этом направлении достиг проект «3D-визуализатор томограмм», разрабатываемый в составе госконтракта ФЦП «Научные и научно-педагогические кадры инновационной России». Проект направлен на достижение наивысших производительности и качества 3D-реконструкции результатов томографии внутренних органов человека и, на этой основе, на сведение воедино результатов томограмм всех типов в одной трехмерной модели с целью увеличения точности диагностики и качества лечения. Основным конкурентом в этой области является компания Fovia (http://fovia.com), мировой лидер в производительности и качестве 3D-реконструкции томограмм.

В результате создан программный комплекс со следующими, интересными для врачей-диагностов и клиницистов, свойствами и характеристиками:

- Реализован алгоритм визуализации, который позволяет добиться реального времени для слияния 3D-визуализации нескольких томограмм, заданных в разных системах координат.
- Реализованы одно- и двухпараметрические функции трансформации, несколько стереорежимов.
- Возможно как одновременное отображение нескольких томограмм разного типа в одном объеме, так и одной томограммы в разных техниках визуализации (более 10 техник, среди них: «рентген», MIP, MinIP, MIDA, DVR и т.д.).
- Поддерживается входной формат DICOM, бинарный формат RAW для 2D-3D-массивов данных.
- Есть возможность отсечения визуализируемой области данных трехмерной сеткой, управляемой пользователем.

Реализован целый ряд приемов улучшения качества визуализации: 1) трикубическая фильтрация данных; 2) накопление кадра во времени; 3) реструктуризация и обработка данных средствами видеокарты; 4) метод адаптивного шага луча [6]; 5) прединтегрированный объёмный рендеринг; 6) метод разбиения шага луча [6, 7].

Преодолено ограничение для GPU на максимальный объем томограммы в 512^3 вокселей. Использование блочного представления данных позволяет обойти это ограничение, а использование блоков разных размеров и покрытие только видимой части данных обеспечивает дополнительный прирост производительности и экономию памяти GPU.

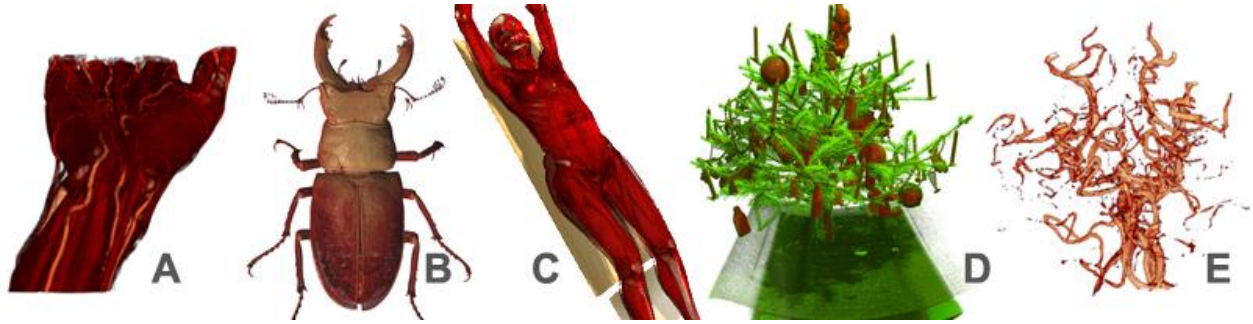


Рис. 2. Тестовые данные: A) hand 244x124x257 (КТ-данные); B) Beetle 832x832x494; C) melanix 512x512x1203 (КТ-данные); D) x-mas 512x512x999 (КТ-ёлки); E) vessels 512^3 (МРТ сосудов головного мозга)

2. Вычислительные эксперименты

Ниже представлены результаты экспериментов по замеру производительности рендеринга при выборе различных размеров блоков, на которые мы делим массив данных. Мы используем кубические блоки размерами 32, 48, 64, 96, 128 и 256. Для выбранных тестовых массивов данных и видеокарты GeForce GTS 250 оптимальными размерами оказались 64 и 128. Использовано 5 тестовых массивов данных, которые общедоступны в сети Интернет, и три различные техники визуализации: объёмный рендеринг без затенения; объёмный рендеринг с затенением по локальной модели освещения Фонга; объёмный рендеринг без затенения с трикубической выборкой. В данных экспериментах мы перекрываем наши блоки на толщину в два вокселя. Размеры окна вывода: 800x600 пикселей. Шаг луча в алгоритме Ray Casting: 0,34 от длины диагонали вокселя.

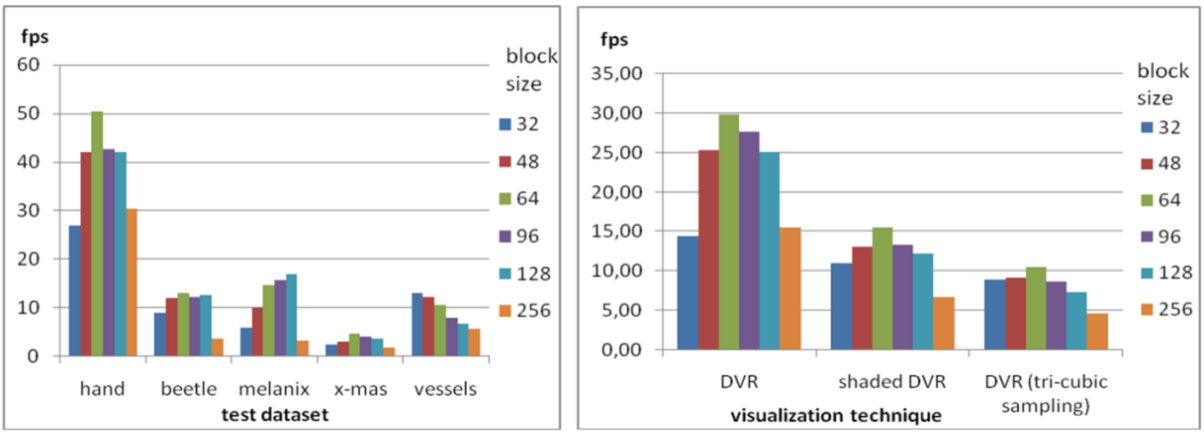


Рис. 3. Производительность рендеринга при различных разбиениях данных

С ростом производительности GPU меняются подходы к оптимизации алгоритмов визуализации. Поэтому нами был исследован и реализован популярный прединтегрированный рендеринг и предложен самостоятельно разработанный метод разбиения ша-

га для постклассификационной модели визуализации. Наш метод показал себя столь же эффективным в устранении артефактов, как и прединтегрированный рендеринг. Применяя основные подходы к оптимизации рендеринга блочных данных, сегодня можно добиться интерактивной и высококачественной объёмной визуализации томограмм размером порядка 2 Гб. Результаты экспериментов показали высокую производительность рендеринга таких данных для видеокарты GeForce GTX 580 3Gb и его готовность к использованию для виртуального анатомического стола [6,7].

Благодаря стратегиям устранения артефактов рендеринга качество интерактивной визуализации стало эквивалентным качеству CPU-реализации от компании Fovia. Итоговые вычислительные эксперименты показали, что производительность созданного программного обеспечения и качество 3D-визуализации томограмм находятся на уровне и выше лучших мировых образцов и достаточны для эксплуатации 3D-визуализатора на рабочем месте врача-клинициста, обеспеченном вычислительной техникой среднего класса. На базе данного ПО могут быть созданы такие образовательные системы и сервисы в области медицины и биологии, как «виртуальный анатомический стол» фирмы Anatomage (www.anatomage.com).

Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839 и гранта Президента РФ № НШ-1960.2012.9

Литература

1. Johns Hopkins Medicine (Hospital) news – [http://www.hopkinsmedicine.org/Press_releases/2007/11_26_07.html].
2. Charette R.N. Visualizing Electronic Health Records With "Google-Earth for the Body" (IBM researchers develop 3-D visualization tool for electronic health records) Jan.2008 (http://spectrum.ieee.org/biomedical/diagnostics/visualizing-electronic-health-records-with-googleearth-for-the-body).
3. IBM Research, Smarter Planet. Thy-Mors hospital uses three-dimensional models of human anatomy to help speed diagnoses and treatments. Published on 10-Dec-2009 / Validated on 26 Sep 2011 (http://www-01.ibm.com/software/success/cssdb.nsf/CS/LMCM-7YKVEJ?OpenDocument&Site=default&cty=en_us).
4. Charette R.N. Using Avatars to Understand Adverse Drug Reactions / March 06, 2012 (http://spectrum.ieee.org/riskfactor/biomedical/diagnostics/using-avatars-to-understand-adverse-drug-reactions/).
5. Сайт межфакультетской лаборатории компьютерной графики ННГУ им. Н.И. Лобачевского – [http://graph-unn.ru], видеоматериалы – [http://www.youtube.com/user/ngavrilov86].
6. Гаврилов Н.И., Турлапов В.Е. Подходы к оптимизации GPU-алгоритма volume raycasting для применения в составе виртуального анатомического стола // Научная визуализация. 2012. Кв.2. Т.4. №2. С.21-56.
7. Nikolay Gavrilov, Vadim Turlapov. Advanced GPU-based Ray Casting for Bricked Dataset // Proc. of SIGGRAPH’2012, 5-9 Aug 2012.

ПАРАЛЛЕЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ ПОЛНООБЪЕКТНЫМИ ЗАЩИЩЕННЫМИ БАЗАМИ ДАННЫХ КАРТОГРАФИЧЕСКИХ СЦЕН

Р.Ф. Гибадуллин, С.В. Пыстогов

Казанский национальный исследовательский технический университет им. А.Н. Туполева

Анализируются тенденции универсализации баз данных картографических сцен (БД КС) и их защиты. Рассматривается исследуемая архитектура СУБД КС. Приводятся разработки: схемы универсальной БД КС с ассоциативной защитой, метода учета погрешностей векторизации, параллельных алгоритмов и программ синтеза БД КС на вычислительном кластере.

Введение

В настоящее время системы управления картографическими базами данных развиваются на стыке многих научных дисциплин и применяются в различных сферах человеческой деятельности, например при изучении природных ресурсов и управлении ими, при создании и ведении кадастров и управлении муниципальными образованиями и в других сферах.

Картографическая продукция имеет следующие особенности:

- Отличается высокой себестоимостью работ по ее получению. Топографами и геодезистами затрачиваются немалые усилия на формирование этой продукции.
- Для последующей обработки требуются векторизация растровых изображений, устранение дефектов топологической структуры при создании векторных карт и организация хранения их в базе данных картографической сцены и соответствующей системы управления.
- Картографическая сцена содержит контурные изображения объектов разных типов с их привязкой к координатам.
- Картографические базы данных характеризуются большой объемностью. Согласно общим сведениям о геоинформационной системе «Панорама», разработанной с учетом широкого круга пользовательских потребностей, векторная карта может содержать несколько тысяч листов. Всего один лист карты может содержать до 4 миллиардов объектов. Например объем векторной кадастровой карты Российской Федерации без учета семантического описания объектов может достигать нескольких десятков гигабайт [1].

Коммерческая и другая ценность картографической продукции требует организации ее информационной безопасности. Обработка запросов к защищенным базам данных картографических сцен связывается со значительными временными затратами. Это приводит к необходимости организации параллельной обработки: распределения картографических баз данных по узлам вычислительного кластера, применения технологий MPI, NVIDIA CUDA и др. Присутствие на тематических слоях карты различных типов объектов (точечных, линейных и площадных) и применение к ним единого механизма защиты требует разработки полнообъектной структуры хранения таких данных. Все это обуславливает необходимость разработки полнообъектной защищенной системы управления базами данных картографических сцен (СУБД КС).

1. Существующие решения

На сегодняшний день для организации хранения и управления пространственными данными в СУБД широко применяется серверное программное обеспечение ArcSDE в составе ArcGIS Server [2]. ArcSDE тесно интегрирована с пакетами ArcEditor, ArcInfo, ArcView для работы с многопользовательскими картографическими базами данных. Например, ArcView действует как клиентское приложение для просмотра и анализа картографических данных.

ArcSDE использует сжатый бинарный формат для хранения картографических данных. Этот бинарный формат был разработан для нескольких платформ СУБД. Важно отметить, что многопользовательская база данных хранится не в собственной структуре, а в стандартных таблицах, управляемых СУБД. В результате пользователи могут обращаться к многопользовательской базе данных посредством ArcSDE.

Ведущие места среди СУБД, обладающих встроенными механизмами защиты баз данных, занимают: *Oracle, Microsoft SQL Server, Sybase Server*. Каждая из них имеет свои определенные преимущества. *Oracle* обладает такими возможностями, как защита и ограничения доступа пользователей к важным данным и приложениям; защита данных при передаче по сети. Но при этом она имеет серьезный недостаток – отсутствие шифрования данных непосредственно в базе данных, что делает систему уязвимой при внутренних атаках (в случае кражи жестких дисков). Большинство средств защиты *Microsoft SQL Server* обеспечивается функциональными возможностями операционной системы, а не с помощью инструментов СУБД. Например, файловая система *NTFS* операционной системы *Windows* обеспечивает полное шифрование каталогов и файлов. Эта функциональная возможность *Windows* используется в СУБД для защиты файлов базы данных от прямых атак. Такой подход не обеспечивает высокой производительности при работе с защищенными базами данных. *Sybase Server* предлагает надежную защиту, включая стандарт тройного шифрования данных (*triple DES*) в базе данных. Это хорошо защищает СУБД от внутренних атак на данные. Защита, предлагаемая в СУБД компании *Sybase*, превосходит защиту *Oracle* и является более интегрированной по сравнению с защитой *Microsoft SQL Server*.

Перечисленные СУБД достаточно универсальны, однако построение СУБД КС, ориентированных на работу с защищенными БД картографии, может существенно повысить эффективность управления такими БД по критерию быстродействия при требуемом уровне стойкости защиты.

2. Базовые принципы формирования полнообъектной защищенной БД КС

Базовые принципы формирования базы данных картографических сцен для случая точечных объектов подробно описаны в статье [3].

Соответствующая структура базы данных показана на рисунке (рис.1).

Описание отношений:

- α, β, γ определяют тип слоя: слой точечных, слой линейных и слой площадных объектов соответственно.
- *Name* – название слоя.
- *Code** – код тематического слоя.
- *Obj_name* – имя точечного объекта.
- *Obj_code** – код точечного объекта.
- *Cluster_num** – номер кластера в тематическом слое.
- *Global_X*, Global_Y** – глобальные координаты нижнего левого угла данного кластера.

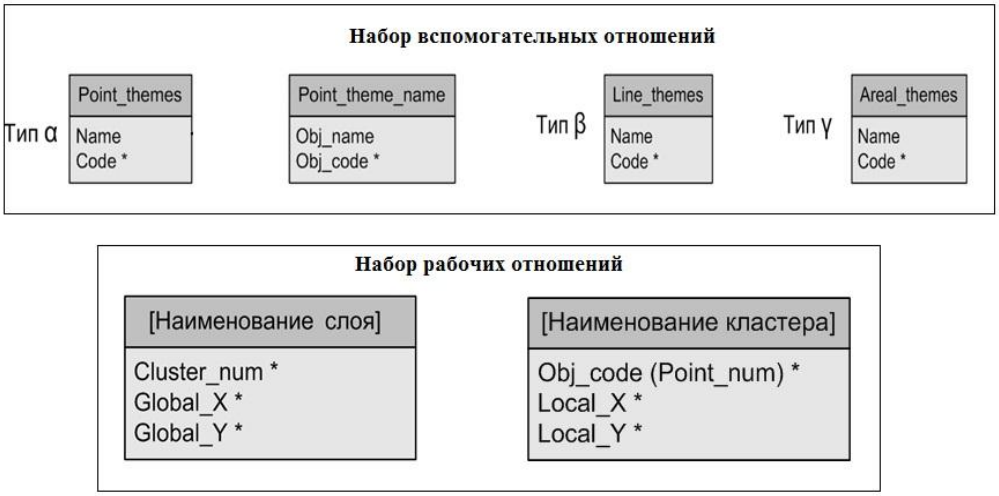


Рис. 1

- *Obj_code (Point_num)** – код точечного объекта (номер узловой точки линейного или площадного объекта).
- *Local_X*, Local_Y** – локальные координаты точки/узла внутри кластера.
- Данные по атрибутам, отмеченные звездочкой, хранятся в сокрытом *виде*.
- Наименование слоя и наименование кластера являются составными компонентами.
- [Наименование слоя] = <Тип_слоя> <Стегокод_слоя>.
- [Наименование кластера]=<Наименование слоя> <№ кластера>.

Различают рабочие и вспомогательные отношения.

Рабочие отношения – набор записей одинакового формата, которые несут в себе основную информацию об объектах картографируемого участка местности в рамках предлагаемого подхода хранения данных в терминах «объект–координаты».

Вспомогательные отношения – это набор записей одинакового формата, несущих дополнительную информацию об объектах, тематических слоях картографируемого участка местности. Данная информация необходима для правильной работы системы с хранимыми данными.

Каждое рабочее отношение с именем типа [Наименование слоя] описывает отдельный тематический слой (фрагменты внутри слоя). Отношения с именем типа [Наименование кластера] описывают по одному фрагменту (точечные объекты и узловые точки внутри него).

В качестве пользовательского приложения для анализа карт в СУБД КС выбрана геоинформационная система (ГИС) MapInfo. Так как она оперирует координатными данными в виде 31-битных целочисленных переменных (знаковый разряд не используется), то минимальная единица деления экстента карты, полученной посредством картографической проекции Гаусса-Крюгера (рис. 2), составит:

$$(X_{\max} - X_{\min})/(2^{31} - 1) = 2\times10^{7} / 2\times10^{9} = 0{,}01\text{м}.$$

Погрешность векторизации по координатным осям в таком случае составляет

$$\mu = 0{,}01/2 = 0{,}005 \text{ м}.$$

Неравенство $\varepsilon > \mu$ неприемлемо из практических соображений, где ε – погрешность представления точечного объекта в разрабатываемой параллельной системе. Равенство погрешностей ($\varepsilon = \mu$) достигается сужением границ защищаемого участка картографической сцены до 10×10 км^2 с шагом локальной координатной сетки, равным 0,01 м.

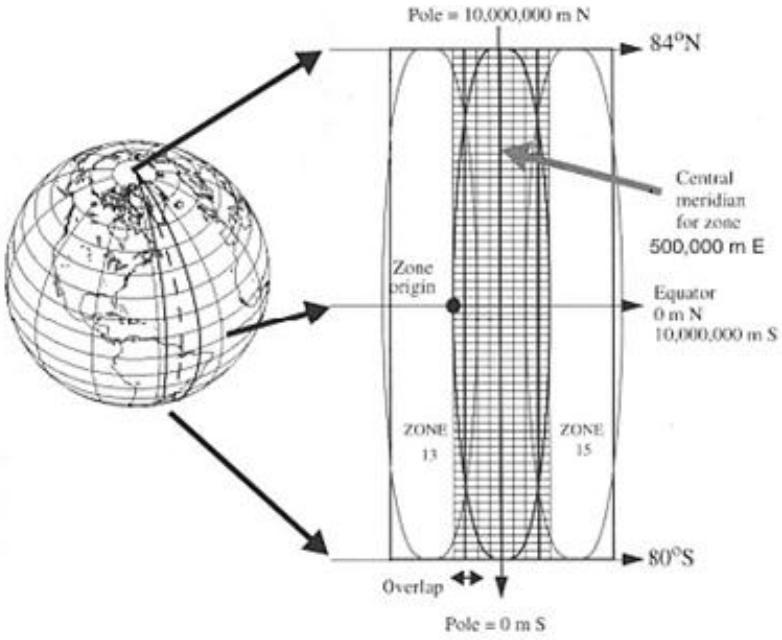


Рис. 2. Геодезическая зона в проекции Гаусса-Крюгера

Таким образом, шаг локальной координатной сетки при формировании баз данных картографических сцен должен определяться минимальной единицей деления экстента карты пользовательской ГИС.

3. Архитектурные особенности системы

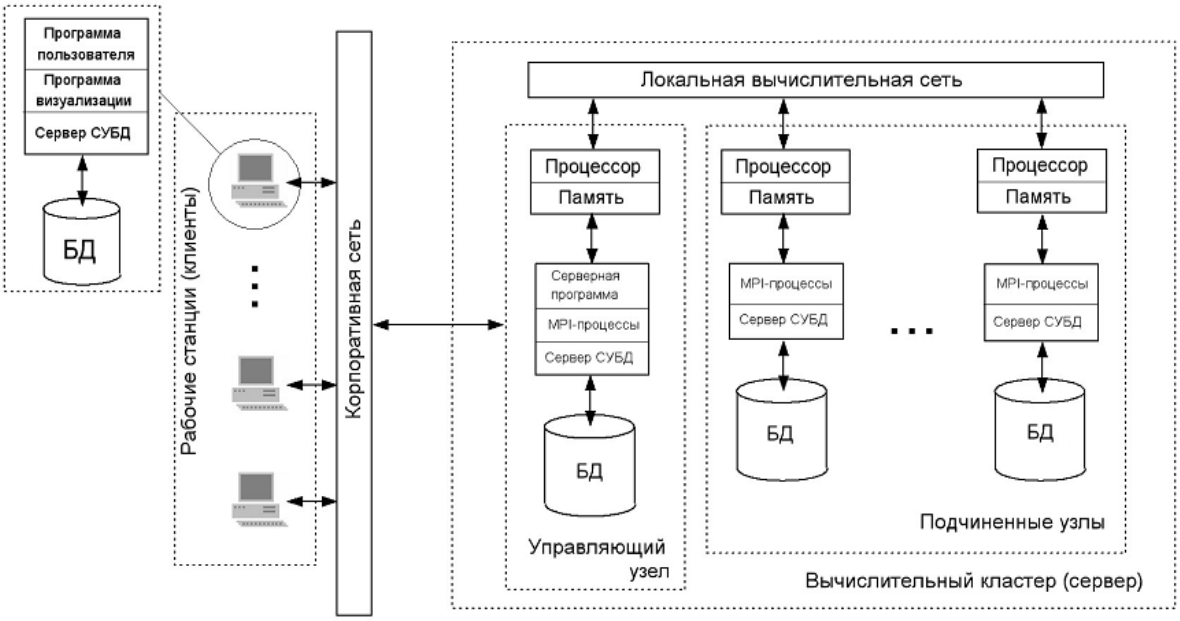


Рис. 3

В системе за основу берется «интеллектуальная» файл-серверная организация взаимодействия. Под интеллектуальностью в данном случае понимается выполнение сервером не только функции хранения, но и функции предварительной обработки поступающих запросов и «интеллектуальный» сбор результатов. Файловый сервер принимает запросы, поступающие по сети от компьютеров-клиентов, и, в соответствии с запросами, передает им требуемые данные. Клиентская часть получает от сервера ответ (ре-

зультат). Данный ответ должен быть предварительно обработан программой клиента для возможности дальнейшей работы пользователя с полученными данными.

Серверная часть состоит из множества однородных узлов, среди которых выделяется один узел, выполняющий функции управления. Именно этот узел является ключевым в задачах получения запросов от клиентов, распределения задач между счетными узлами кластера, сбора результатов и т.д. Остальные узлы кластера являются вычислительными.

К серверу по корпоративной сети подключается множество пользователей. На их рабочих местах должно быть установлено специальное программное обеспечение для работы с БД КС.

4. Синтез БД КС

1. Считывание информации об объектах (точечных, линейных и площадных) картографируемого участка местности из файлов формата MIF/MID в текстовый файл.
2. Создание на управляющем узле кластера структуры будущей базы данных на основе информации, вводимой пользователем или считываемой из входных файлов.
3. Равномерное распределение полученного файла с информацией об объектах по всем узлам кластера (либо по определенным узлам согласно указаниям администратора).
4. Параллельная обработка информации об объектах узлами кластера. Заполнение таблиц базы данных на управляющем узле.
5. После формирования базы данных с открытой информацией на управляющем узле происходит распределение таблиц по узлам кластера для последующих хранения и обработки запросов. Распределению методом горизонтального деления подвергаются рабочие отношения, в то время как вспомогательные отношения полностью дублируются в базе данных каждого узла.
6. После распределения выполняется операция маскирования всеми узлами. Её можно назвать параллельной, т.к. все узлы в совокупности обрабатывают полностью всю БД КС, хотя каждый узел маскирует свой набор записей.

Литература

1. Сообщество специалистов в области ГИС и ДЗЗ [Электронный ресурс]. GIS-Lab © 2012. URL: http://www.gis-lab.info.
2. Картографическая платформа для вашей организации [Электронный ресурс]. Esri © 2012. URL: http://www.arcgis.com.
3. Райхлин В.А., Вершинин И.С., Гибадуллин Р.Ф. Конструктивное моделирование систем в приложении к защите данных картографии // Методы моделирования: Труды Респ. научн. семинара АН РТ. – Казань: «ФЭН» (Наука), 2010. Вып. 4. С. 68–95.

ДЕТЕКТИРОВАНИЕ ЛИЦ ЛЮДЕЙ НА ИЗОБРАЖЕНИИ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ

А.С. Голубев, И.И. Зиновьев

Владимирский госуниверситет им. А.Г. и Н.Г. Столетовых

Предложен вариант алгоритма для детектирования лиц людей на изображении с использованием графических ускорителей. Данный алгоритм был проверен на практике в реальных условиях и подтвердил свою работоспособность при анализе видеопотока высокого разрешения в реальном времени.

Введение

С развитием современной техники видеонаблюдения повышаются требования к быстродействию алгоритмов детектирования лиц людей на изображении в связи с увеличением разрешения анализируемых кадров. Современные центральные процессоры (CPU) уже не справляются с этой задачей в реальном времени. Одним из возможных решений этой проблемы является использование графических ускорителей (GPU). В работе предлагается вариант реализации алгоритма детектирования лиц людей на изображении, представленного в 2001 году Полом Виолой (Paul Viola) и Майклом Джонсом (Michael Jones) [5], для архитектуры NVIDIA CUDA.

1. Особенности реализации алгоритма

Подходы к распараллеливанию алгоритма Виола-Джонса подробно рассмотрены в работе [1]. Мы выбрали вариант, в котором каждый поток анализирует свой участок изображения. В качестве основы взята реализация алгоритма Виола-Джонса для CPU из открытой библиотеки OpenCV [6].

Рассмотрим особенности предлагаемой нами реализации алгоритма детектирования.

1.1. Размещение исходных данных в памяти GPU

Исходное изображение загружается в текстурную память графического ускорителя, что позволяет реализовать эффективный алгоритм его масштабирования путем выборки из текстуры с аппаратной билинейной фильтрацией.

Т.к. каскад классификаторов является неизменным и необходимым всем потокам одновременно, то наиболее эффективно хранить его в константной памяти. Размещение классификатора в памяти графического ускорителя осложняется сложной иерархией его структур в библиотеке OpenCV и ограниченным размером константной памяти. Поэтому формат хранения каскада был преобразован с использованием одномерных массивов и минимально возможных типов данных. Сам процесс приведения каскада к новому формату не вносит значительных временных затрат, т.к. производится один раз при загрузке каскада. Такая реорганизация каскада позволила хранить в константной памяти графического ускорителя до 2150 слабых классификаторов.

Следует заметить, что все буферы в памяти GPU выделяются один раз на этапе инициализации детектора, что немаловажно при обработке видеопотока.

1.2. Использование вычислительных ресурсов GPU

Особенностью нашей реализации по сравнению с другими [1, 2, 4] является использование цельного ядра детектирования (рис. 1). Это позволяет значительно снизить число обращений в медленную глобальную память за счет размещения данных в разделяемой памяти.

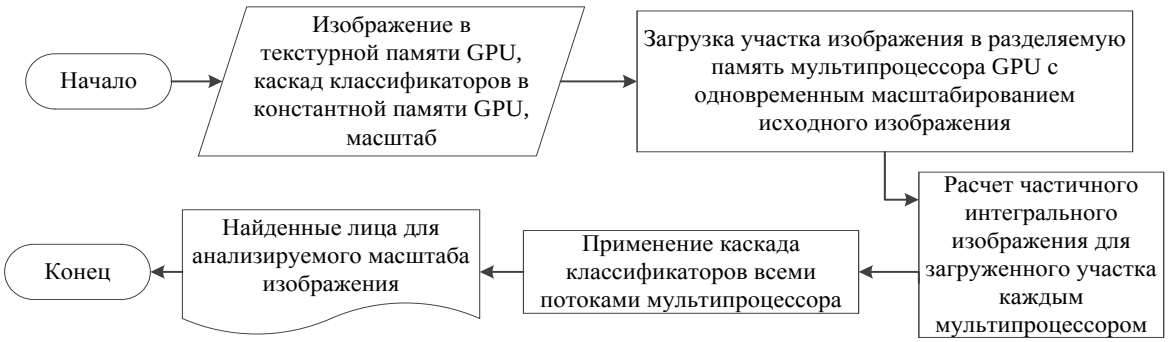


Рис. 1. Алгоритм работы ядра детектирования

Для достижения поставленной цели в разделяемую память мультипроцессора из текстурной памяти считывается участок изображения, достаточный для обеспечения исходными данными всех его потоков. Затем производится вычисление независимых интегральных изображений для частей изображения, уже размещенных в разделяемой памяти мультипроцессоров. Каждый мультипроцессор сначала выполняет суммирование своего участка изображения по строкам, затем – по столбцам. Затем каждый поток независимо от других анализирует свой участок изображения на наличие на нем лица.

Главным недостатком такого подхода является большой размер полученного ядра детектирования и, следовательно, большое число регистров, требуемое ему для работы. Но правильный выбор параметров для выполнения ядра позволяет снизить влияние этого фактора (рис. 2).

1.3. Получение результатов из памяти GPU

В процессе детектирования каждый поток сохраняет результат своей работы в выделенную на этапе инициализации область глобальной памяти. Для ускорения работы реализована их асинхронная загрузка в оперативную память компьютера (технология «zero copy»). После завершения анализа очередного кадра в оперативную память выгружаются все оставшиеся к этому моменту промежуточные результаты. Дальнейшая обработка полученных результатов, связанная с группировкой лиц, найденных в разных масштабах, производится на центральном процессоре.

2. Апробация предложенной реализации алгоритма детектирования

Для наглядной оценки скорости работы полученной реализации алгоритма (обозначим как gpu) проведем сравнение с многопоточными вариантами данного алгоритма, представленными в библиотеке OpenCV, как для центрального процессора (OCV-cpu), так и для графического ускорителя (OCV-gpu). Эксперимент состоит в следующем: один и тот же видеофайл продолжительностью 6 минут, полученный с камер видеонаблюдения, установленных на турникетах метрополитена, анализировался всеми вариантами алгоритма. Эксперимент проводился на компьютере, в состав которого входят 2 процессора Intel Xeon E5520 2.27 Ghz (в сумме 8 физических или 16 логических ядер) и видеокарта NVIDIA GTX 470 (448 CUDA-ядер). При этом фиксировались следующие показатели: минимальное (T_{min}), максимальное (T_{max}) и среднее (T_{avg}) время,

затраченное на анализ одного кадра, процент загрузки центрального процессора (P_{cpu}) и процессора графического ускорителя (P_{gpu}), объем используемой оперативной памяти (M_{cpu}) и памяти графического ускорителя (M_{gpu}). В ходе эксперимента для всех вариантов алгоритма были установлены следующие параметры: минимальный размер лица 20x20 пикселей, шаг масштабирования 1.2, коэффициент группировки 3, каскад классификаторов haarcascade_frontalface_alt2.xml из состава библиотеки OpenCV. Результаты эксперимента приведены в табл. 1.

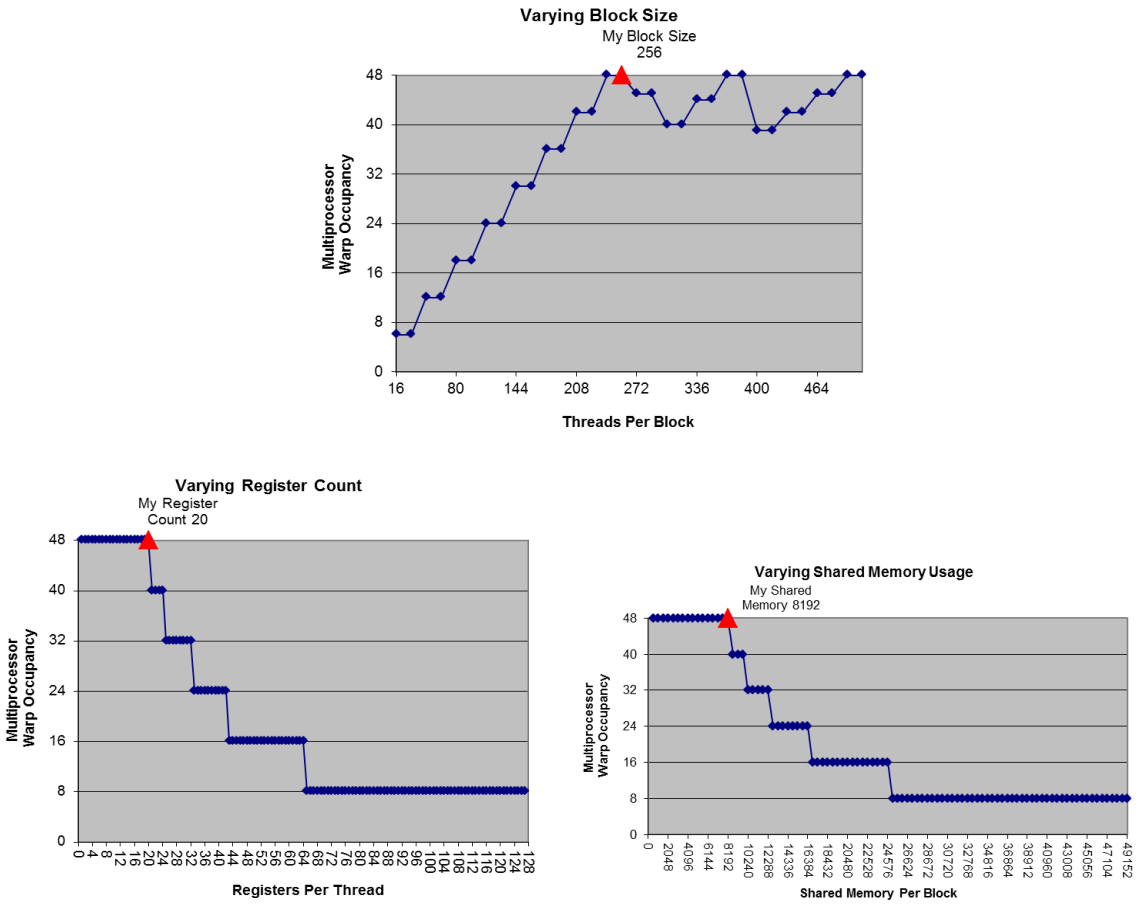


Рис. 2. Выбранные параметры выполнения ядра детектирования (с использованием NVIDIA Occupancy Calculator)

Таблица 1. Сравнительные характеристики разных реализаций алгоритма Виола-Джонса

Реализация	T_{min}, мс	T_{max}, мс	T_{avg}, мс	P_{cpu}, %	P_{gpu}, %	M_{cpu}, МБ	M_{gpu}, МБ
OCV-cpu	410,23	585,03	438,49	60	-	79,348	-
OCV-gpu	171,29	213,91	195,78	7	76	184,908	201
gpu	46,73	82,46	69,91	7	59	89,344	63

Следует отметить, что предложенная нами реализация алгоритма детектирования оказалась как быстрее, так и эффективнее в использовании ресурсов графического ускорителя.

Сравнение эффективности работы рассмотренных выше программных реализаций алгоритма Виола-Джонса проводилось по относительному показателю положительных (True positive rate) и ложных (False positives) обнаружений при анализе изображений из

базы Face Detection Data Set and Benchmark (FDDB) [3]. В ходе эксперимента для алгоритма Виола-Джонса были установлены параметры, аналогичные описанным выше, изменялся только коэффициент группировки в пределах от 1 до 7. Полученные результаты приведены на рис. 3.

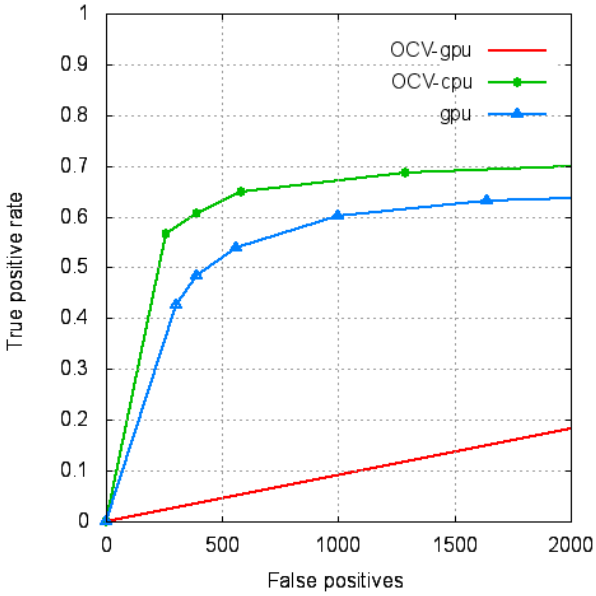


Рис. 3. Сравнение программных реализаций по базе FDDB (Discrete Score)

Предложенная в работе реализация алгоритма продемонстрировала сравнимые результаты с вариантом для центрального процессора.

Заключение

Полученные результаты демонстрируют, что предложенный нами вариант адаптации алгоритма Виола-Джонса для графического ускорителя применим на практике и успешно конкурирует с другими подходами.

Данная работа выполнена в рамках НИР по госзаданию «Наука» (Регистрационный номер: 8.3303.2011 от 23.11.2011).

Литература

1. Harvey J.P. GPU acceleration of object classification algorithms using NVIDIA CUDA. Master’s thesis, Rochester Institute of Technology. 2009.
2. Hefenbrock D., Oberg J., Thanh N.T.N., Kastner R., Baden S.B. Accelerating viola-jones face detection to FPGA-level using GPUs // In Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM ’10, Washington, DC, USA. IEEE Computer Society. 2010. P. 11–18.
3. Jain V., Learned-Miller E. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. 2010.
4. Obukhov A. Haar classifiers for object detection with CUDA. GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. Addison Wesley, 2004. P. 517–544.
5. Viola P., Jones M. Robust real-time face detection // International Journal of Computer Vision. 2004. 57(2). P. 137–154.
6. Документация открытой библиотеки OpenCV – [http://opencv.willowgarage.com/wiki/].

РАЗРАБОТКА ПАРАЛЛЕЛЬНОГО ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ ОЦЕНКИ СЕЙСМОСТОЙКОСТИ НАЗЕМНЫХ СООРУЖЕНИЙ ПУТЁМ ПРЯМОГО ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ

В.И. Голубев, И.Б. Петров, Н.И. Хохлов

Московский физико-технический институт

При землетрясении из его очага распространяются упругие волны с различными частотами и амплитудами. При достижении поверхности они вызывают как вертикальные, так и горизонтальные движения земли, что приводит к повреждению несущих стен строений и их разрушению. Результатами этого являются огромное число человеческих жертв и существенные материальные убытки. Недавние землетрясения в России и Японии, с сопутствующими им колоссальными разрушениями строений, показали, что существующие на сегодняшний день инструменты оценки сейсмостойкости недостаточны.

В рамках данной работы был разработан сеточно-характеристический численный метод для решения динамических задач упругого тела (распространение сейсмических волн в гетерогенных геологических средах) на структурных криволинейных трёхмерных расчётных сетках. В докладе будет представлена механико-математическая вычислительная модель очага землетрясения с методикой определения параметров модели по натурным измерениям. Также был реализован исследовательский код (C++), позволяющий проводить математическое моделирование динамических процессов в геологических средах. Особое внимание было уделено распараллеливанию с использованием архитектур (MPI + OpenMP) для ускорения расчётов с использованием высокопроизводительных вычислительных систем.

ОБОБЩЕНИЯ МЕТОДА DIRECT НА ЗАДАЧИ С ФУНКЦИОНАЛЬНЫМИ ОГРАНИЧЕНИЯМИ

С.Ю. Городецкий

Нижегородский госуниверситет им. Н.И. Лобачевского

Метод Direct был разработан в 1993 году для решения задач липшицевой глобальной оптимизации без вычисления оценок константы Липшица. Значения константы могли быть любыми от нуля до бесконечности. Метод не учитывал функциональные ограничения. В данной работе предлагаются два его специальных обобщения на случай наличия таких ограничений. Первое основано на сведении к задаче безусловной глобальной оптимизации с перестраиваемой целевой функцией. Второе – непосредственно распространяет принципы построения Direct на случай липшицевых ограничений при неограниченном диапазоне их констант Липшица. Приведены теоретические основы построения новых методов, обоснована сходимость.

Введение

Метод Direct предложили Jones D.R., Perttunen C.D., Stuckman B.E. в работе [1]. Метод предназначен для поиска на гиперинтервале $D=[a,b]\subset R^N$ глобального минимума липшицевой функции без использования значений константы Липшица L_f:

$$f^*=f(x^*)=\min f(x),\ \ x\in D=[a,b]\subset R^N. \qquad (1)$$

Относительно L_f предполагается, что ее значение может быть любым числом из диапазона $[0,\infty)$. Метод Direct использует разбиение области поиска D на гиперинтервалы $D_i=[a_i,b_i]$ по центральной схеме деления на три. Эта схема деления первоначально была предложена Евтушенко Ю.Г. и Ратькиным В.А. для задач с заданным значением константы L_f. В [2] опубликован предварительный вариант этого метода (в форме схемы деления на два). В их методе на каждом шаге выбирался, для последующего деления на три равные части (по большему ребру), наиболее *приоритетный* гиперинтервал D_t. Его выбор определялся тем, что нижняя оценка функции $f(x)$ в пределах D_t по измерению в его геометрическом центре $c_t=(a_t+b_t)/2$, имеющая вид

$$f_t^-(L_f)=f^-(L_f,D_t)=f(c_t)-L_f\cdot\|b_t-a_t\|/2, \qquad (2)$$

оказывалась наименьшей по сравнению с другими гиперинтервалами при заданном значении L_f. При этом из сравнения исключались гиперинтервалы, в которых значения $f_i^-(L_f)$ улучшали достигнутое к данной итерации наименьшее значение $f_{\min}$ функции менее, чем на $\varepsilon>0$. То есть выбор *приоритетного* D_t определялся правилами:

$$\forall D_i:\ f_t^-(L_f)\le f_i^-(L_f), \qquad (3)$$

$$f_t^-(L_f)\le f_{\min}-\varepsilon. \qquad (4)$$

В методе Direct [1], описание которого можно также найти в [3], значение L_f неизвестно, что приводит к следующему изменению принципа (3)-(4) выбора приоритетных гиперинтервалов (их при неизвестном L_f называют *потенциально оптимальными*). D_t

является *потенциально оптимальным*, если существует хотя бы одно значение $L_f \in [0,\infty)$, при котором условие (3) оказывается справедливым, и, кроме того, выполняется измененный аналог (6) условия (4).

Правило выбора *потенциально оптимальных* гиперинтервалов приобретает вид:

$$\exists L_f \in [0,\infty):\ \forall D_i:\ f_t^-(L_f) \le f_i^-(L_f), \tag{5}$$

$$f_t^-(L_f) \le f_{\min} - \varepsilon |f_{\min}|. \tag{6}$$

В работе [1] найдено простое геометрическое представление, позволяющее выделять множество гиперинтервалов D_t, удовлетворяющих условию (5). А именно, если изобразить гиперинтервалы D_i на плоскости точками с координатами (d_i, f_i), где $d_i = \| b_i - a_i \|$, $f_i = f(c_i)$, то множеству точек (d_t, f_t) для гиперинтервалов D_t соответствуют вершины правой нижней части границы выпуклой линейной оболочки множества всех точек (d_i, f_i). Для вычислительно эффективного выделения точек (d_t, f_t) в [1] предложено использовать правило Грэхема [4].

В методе Direct на каждой итерации выполняется разбиение по принципу деления на три (по большим ребрам) всех найденных потенциально оптимальных гиперинтервалов D_t с последующим проведением в новых центрах образовавшихся гиперинтервалов измерений функции. Вычисления значений функций в новых центрах могут выполняться параллельно. Останов вычислений выполняется при превышении предельного количества итераций или предельного количества измерений функции. Метод при тестировании демонстрирует хорошие характеристики.

1. Известные обобщения метода Direct

К настоящему времени разработаны следующие обобщения Direct. В монографии [3] построен более эффективный вариант метода, работающий по диагональной схеме деления на три. В [5] получено нетривиальное обобщение метода Direct на задачи с вычислимой липшицевой производной, константа Липшица которой неизвестна и может изменяться на промежутке $[0,\infty)$; метод построен для задач (1) с размерностью $N = 1$. В [6] метод [5] распространен на многомерные задачи с использованием новой схемы разбиения области.

Обобщений метода Direct на задачи с функциональными ограничениями к настоящему времени построено не было. В некоторых работах предлагалось для метода Direct учитывать ограничения с использованием штрафных функций, однако этот подход нельзя считать корректным, поскольку предлагаемые штрафы не являются точными.

2. Два подхода к обобщению метода Direct на задачи с функциональными ограничениями

Задача с функциональными ограничениями ставится следующим образом:

$$f^* = f(x^*) = \min f(x),\ x \in X, \tag{7}$$

$$X = \{x \in D = [a,b] \subset R^N : g_i(x) \le 0, (i = 1,..,m)\}. \tag{8}$$

Ограничения неравенства сводятся к одному обобщенному ограничению вида

$$g(x) \le 0, \tag{9}$$

$$g(x) = \max\{\ g_1(x),...,g_m(x)\ \}. \tag{10}$$

В двух предлагаемых подходах принимаются разные предположения о свойствах функций задачи.

Первый подход использует сведение исходной задачи (7)-(8) к задаче без функциональных ограничений с перестраиваемой целевой функцией. При этом исходные

предположения заключаются в том, что функции f и g являются липшицевыми с равными константами $L_f = L_g = L$, где значение L неизвестно и $L \in [0,\infty)$.

Пусть f_k^* – текущая оценка глобально-минимального значения функции в (7)-(8):

$$f_k^* = \begin{cases} \min\{\ f(c_i) : c_i \in D_i \ \&\ g(c_i) \le 0, i = 1,..,M_k\}, & \exists D_i : g(c_i) \le 0 \\ +\infty, \quad \forall D_i :\ g(c_i) > 0, \end{cases} \tag{11}$$

где M_k – число гиперинтервалов D_i на k-й итерации, а c_i – их центральные точки.

Введем перестраиваемую целевую функцию $f_k(x)$, зависящую от оценки f_k^*:

$$f_k(x) = \max\{\ f(x) - f_k^*; g(x)\ \}. \tag{12}$$

Ее константа Липшица равна L, значение L неизвестно, и $L \in [0,\infty)$. Вместо задачи (7)-(8) с функциями-ограничениями будем решать задачу без таких ограничений:

$$\min f_k(x),\ x \in D = [a,b] \subset R^N. \tag{13}$$

Заметим, что при $f_k^* = f^*$ множество глобальных минимумов в (12)-(13) совпадает с множеством глобальных минимумов исходной задачи. Отметим также, что предложенный способ учета ограничений использовался в работе [7] для триангуляционного метода с оцениванием константы Липшица производных по направлениям.

К решению (12)-(13) может быть применен модифицированный метод Direct, отличающийся от стандартного только учетом изменяемости функции в процессе поиска ее минимума. Поскольку метод Direct строит неравномерное, но в пределе всюду плотное покрытие области поиска D [3], значение f_k^* будет стремиться к f^* из (7) при $k \to \infty$, что гарантирует сходимость модифицированного метода.

Второй подход не использует сведения задачи с ограничениями к задачам без ограничений и предлагает прямое специальное обобщение принципов построения метода Direct на задачи с ограничениями.

При втором подходе предполагается, что в задаче (7)-(10) функция f является липшицевой с константой L, значение L неизвестно, и $L \in [0,\infty)$. Кроме того, принимается, что функция g из (10) липшицева с константой L_g, значение константы неизвестно, но взаимосвязано с L соотношением $L_g \in [0, \gamma \cdot L]$, где $\gamma > 0$. Поскольку значение L неограниченно, константа Липшица обобщенного ограничения потенциально также может принимать сколь угодно большие значения.

В рассматриваемой задаче каждый гиперинтервал D_i (его диаметр – d_i) с вычисленными в его центре значениями $f_i = f(c_i)$ и $g_i = g(c_i)$ характеризуется уже не парой, а тройкой чисел (d_i, f_i, g_i), что принципиально отличает ситуацию от задачи без функциональных ограничений (1). При выборе «лучших» гиперинтервалов D_t для их деления на три на очередной итерации необходимо сформулировать новый принцип отбора. Отбираемые «лучшие» гиперинтервалы по-прежнему будем назавать *потенциально оптимальными*. Они будут выбираться из категории гиперинтервалов, которые будем называть *недоминируемыми* по аналогии с терминологией, использованной в [3].

Если временно считать, что значения L и L_g известны, то минимально возможные значения функций f и g в пределах гиперинтервала $D_i = [a_i, b_i]$, построенные по результатам вычисления f и g в центральной точке c_i, определяются соотношениями

$$f_i^-(L) = f^-(L, D_i) = f(c_i) - L \cdot d_i / 2;\ g_i^-(L_g) = g^-(L_g, D_i) = g(c_i) - L_g \cdot d_i / 2. \tag{14}$$

Гиперинтервал D_t будем называть *недоминируемым*, если

$$\exists L \in [0,\infty):\ f_t^-(L) = \min\{ f_i^-(L) : \exists L_g \in [0, \gamma \cdot L] : g_i^-(L_g) \le 0, 1 \le i \le M_k \}, \tag{15}$$

$$\exists\, L_g \in [0, \gamma \cdot L]:\ \ g_t^-(L_g) \le 0. \tag{16}$$

Каждому недоминируемому D_t отвечает некоторый диапазон $\{L(D_t)\}$ соответствующих ему (в силу (15)-(16)) значений константы L. *Потенциально оптимальными* назовем такие недоминируемые гиперинтервалы D_t, для которых дополнительно

$$\exists L \in \{L(D_t)\}:\ f_t^-(L) \le f_k^* - \varepsilon \left| f_k^* \right|. \tag{17}$$

Введенное определение потенциальной оптимальности соответствует наличию ограничений в (7)-(8), однако является достаточно сложным. Для его практического использования необходимы простые конструктивные правила, последовательное применение которых приводило бы к отбору потенциально оптимальных D_t. Ниже формулируются правила отбора, включающие несколько этапов.

Этап 1. Гиперинтервалы текущего разбиения D_i разделяются на группы – слои. В каждый слой объединяются все D_i, имеющие один и тот же диаметр $d_i = c$ (будем обозначать их как D_i^c). На множестве гиперинтервалов одного слоя вводится понятие *недоминируемости в слое*, аналогичное (15)-(16). Каждому недоминируемому в слое $\{D_i^c\}$ гиперинтервалу (обозначим их через $\hat{D}_{i(s)}^c$, где $s = 1,2,...$ – введенная порядковая нумерация) будет соответствовать диапазон подходящих для $\hat{D}_{i(s)}^c$ в слое значений констант $\{L(\hat{D}_{i(s)}^c)\} = [L_s^c, L_{s+1}^c)$. Для выделения в слое гиперинтервалов $\hat{D}_{i(s)}^c$ элементы слоя удобно представить точками (g_i, f_i) на плоскости (g, f), как показано на рис. 1.

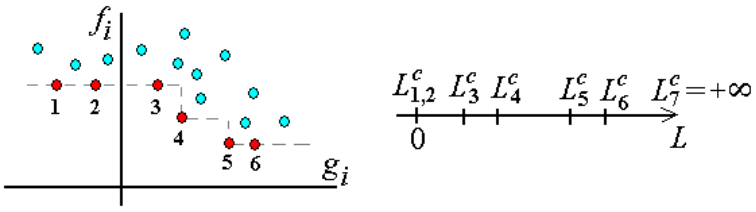


Рис.1. Набор точек соответствует слою $d_i = c$, красные точки отвечают недоминируемым в слое гиперинтервалам $\hat{D}_{i(s)}^c$, $s = 1,2,..,6$

Доказаны следующие леммы.

Лемма 1. Если в слое $\{D_i^c\}$ существуют точки (g_i, f_i) со значениями $g_i \le 0$, то гиперинтервалы D_j^c со значениями $g_j \le 0$, $f_j = f^{*c} = \min\{ f_i : g_i \le 0, D_i \in \{D_i^c\}\}$ являются недоминируемыми в слое (т.е. относятся к типу $\hat{D}_{i(s)}^c$) и для них $L_s^c = 0$.

Если в слое $\{D_i^c\}$ нет гиперинтервалов с $g_j \le 0$, то будем полагать $f_c^* = +\infty$.

Лемма 2. Для того чтобы в слое $\{D_i^c\}$ точка (g_{i^*}, f_{i^*}) с $g_{i^*} > 0$ соответствовала некоторому недоминируемому в слое гиперинтервалу $\hat{D}_{i(s)}^c$ (где $i(s) = i^*$), необходимо и достаточно, чтобы $f_{i^*} \le f^{*c}$ и в слое не существовало другого гиперинтервала D_j, для которого $g_j \le g_{i^*}$, $f_j < f_{i^*}$. Если это выполнено, $L_s^c = 2g_{i^*} / (\gamma \cdot d_{i^*})$, где $i^* = i(s)$.

Перенумеруем индексом s недоминируемые в слое гиперинтервалы $\hat{D}^c_{i(s)}$ в порядке возрастания значений L^c_s. Для гиперинтервала $\hat{D}^c_{i(s)}$ с наибольшим номером s положим $L^c_{s+1} = +\infty$. Правила отбора, выполняемые на этапе 1, иллюстрирует рис.1.

Этап 2. Выполняется упорядочение по возрастанию найденных значений L^c_s для всех s и всех слоев c. Полученный набор значений обозначим как $0 \le L_1 < L_2 <..$ $< L_{m_k} < L_{m_{k+1}} = +\infty$. Для каждого промежутка $[L_j, L_{j+1})$ выделяются все найденные гиперинтервалы $\hat{D}^c_{i(s)}$ из разных слоев c, для которых $[L_j, L_{j+1}) \cap [L^c_s, L^c_s) \ne \varnothing$. Набор таких гиперинтервалов обозначим $\{\tilde{D}^j_\ell\}$ $(\ell = 1,.., \bar{\ell}(j))$. По построению для значений $L \in [L_j, L_{j+1})$ недоминируемые по совокупности слоев гиперинтервалы могут содержаться только в наборе $\{\tilde{D}^j_\ell\}$.

Этап 3. Для всех $j = 1,..., m_k$ выполняется взаимное сравнение гиперинтервалов в пределах каждого из наборов $\{\tilde{D}^j_\ell\}$ $(\ell = 1,.., \bar{\ell}(j))$. Сравнение проводится на плоскости (d_i, f_i) с выделением *потенциально оптимальных* гиперинтервалов D_t по принципу:

$$D_t \in \{\tilde{D}^j_\ell : \ell = 1,.., \bar{\ell}(j)\},$$

$$\exists L \in [L_j, L_{j+1}):\ \forall \tilde{D}^j_\ell, (\ell = 1,.., \bar{\ell}(j)):\ f^-(L, D_t) \le f^-(L, \tilde{D}^j_\ell), \tag{18}$$

$$f^-(L, D_t) \le f^*_k - \varepsilon |f^*_k|. \tag{19}$$

Все выделенные таким образом гиперинтервалы D_t перед началом следующей итерации разделяются на три равные части по их большим ребрам с вычислением функций f и g в новых центрах вновь образованных гиперинтервалов.

Доказано, что для построенного метода $f^*_k \to f^*$ при $k \to \infty$. В докладе будут приведены примеры решения тестовых задач.

Работа выполнена при финансовой поддержке Минобрнауки России, государственное соглашение о предоставлении гранта № 14.B37.21.0878.

Литература

1. Jones D.R., Perttunen C.D., Stuckman B.E. Lipschitzian optimization without the Lipschitz constant // J. Optim. Theory and Appl. 1993. V. 79. № 1. P. 157–181.
2. Евтушенко Ю.Г., Ратькин В.А. Метод половинных делений для глобальной оптимизации функций многих переменных // Изв. АН СССР. Техническая кибернетика. 1987. Т.1. С. 119-127.
3. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008. – 352 с.
4. Препарата Ф.Ф., Шеймос М.И. Вычислительная геометрия. Введение. – М.: Мир, 1989. – 478 с.
5. Sergeyev Ya.D., Kvasov D.E. A univariate global search working with a set of Lipschitz constants for the first derivative // Optimization Lettes. 3. 2009. – P. 303-318.
6. Kvasov D.E., Sergeyev Ya.D. Lipschitz gradients for global optimization in a one-point-based partitioning scheme // Journal of Computational and Applied Mathematics. 2012. V. 236. P. 4042–4054.
7. Городецкий С.Ю. Триангуляционные методы параболоидов в задачах многоэкстремальной оптимизации с ограничениями для класса функций с липшицевыми производными по направлениям // Вестник Нижегородского университета им. Н.И. Лобачевского. № 1 (1). 2012. – С. 144–155.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ОПТИЧЕСКОЙ ДИФФУЗИОННОЙ СПЕКТРОСКОПИИ ДЛЯ ФУНКЦИОНАЛЬНОЙ ДИАГНОСТИКИ МОЗГА ЧЕЛОВЕКА

А.В. Горшков[1], М.Ю. Кириллин[2], А.Л. Коршунова[1]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]Институт прикладной физики РАН, Нижний Новгород

Представлен адаптированный алгоритм моделирования распространения зондирующего излучения в голове человека для функциональной диагностики мозга. Дана оценка эффективности и масштабируемости параллельной версии адаптированного алгоритма для систем с общей и распределенной памятью. Описан выбор оптимального алгоритма поиска пересечений применительно к задаче моделирования распространения зондирующего излучения в биологических тканях со сложной геометрией слоев.

Введение

В настоящее время в медицинских исследованиях существует потребность в развитии новых, безопасных для человека и доступных методов диагностики, поскольку используемые традиционные методы (МРТ, КТ, ПЭТ) имеют ряд ограничений, связанных с их небезопасностью, высокой стоимостью оборудования и значительными требованиями к инфраструктуре. Классом наиболее перспективных методов диагностики, которые могут применяться как в сочетании с существующими методами, так и, в некоторых случаях, вместо них, являются оптические методы.

Одним из таких методов является оптическая диффузионная спектроскопия (ОДС), основанная на регистрации многократно рассеянного объектом зондирующего излучения на нескольких длинах волн. Нужные длины волн определяются в соответствии со спектрами поглощения исследуемых компонентов организма. В частности, применение ОДС для функциональной диагностики мозга основано на облучении мозга человека источниками излучения различных длин волн, выбирающихся в соответствии с различиями спектров поглощения окси- и дезоксигемоглобина, концентрации которых в коре головного мозга изменяются при умственной активности. Взаимное расположение источников и детекторов определяет измерительный объем, в то время как информация о потерях излучения в среде позволяет определить соотношение окси- и дезоксигемоглобина в измерительном объеме.

Получение количественной диагностической информации на основании данных ОДС требует учета особенностей процессов распространения излучения в биологических объектах. Из-за сложной и неоднородной структуры таких объектов использование аналитических подходов к описанию этих процессов затруднительно. В этой связи эффективным выглядит применение компьютерного моделирования распространения зондирующего излучения в объектах со сложной геометрией.

1. Адаптированный алгоритм моделирования распространения зондирующего излучения в голове человека

Для решения задачи компьютерного моделирования оптической диффузионной спектроскопии был разработан адаптированный алгоритм моделирования распростра-

нения зондирующего излучения в многослойных биологических тканях (рис. 1). Алгоритм основан на применении метода Монте-Карло и учитывает преломление света на границах слоев со сложной геометрией.

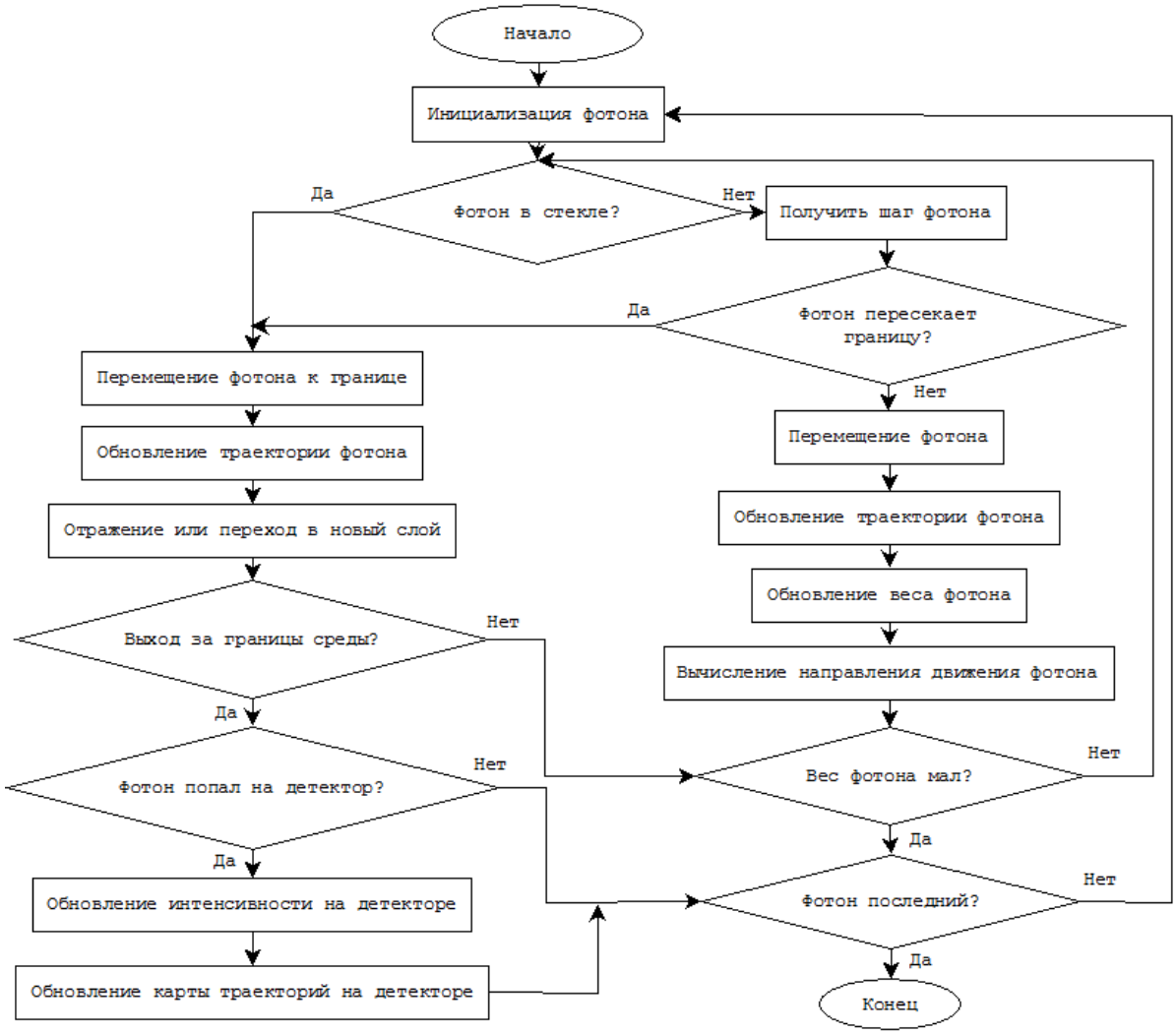


Рис. 1. Адаптированный алгоритм моделирования распространения зондирующего излучения в голове человека

В настоящее время метод Монте-Карло активно применяется для моделирования формирования сигналов и предсказания результатов применения методов оптической биомедицинской диагностики [1-3]. Метод основан на многократном повторении случайного испытания и последующем обобщении полученных данных. Применительно к распространению излучения оптического диапазона в биотканях он заключается в многократном моделировании случайных траекторий фотонов в исследуемой среде. Световые поля единичных фотонов являются независимыми, что делает возможным эффективное распараллеливание описываемого метода.

Предложенный алгоритм основан на работе [1], однако содержит ряд отличий:

- в алгоритме реализована поддержка произвольной геометрии слоев среды. Границы слоев описываются набором треугольников. Реализован алгоритм быстрого поиска пересечений фотонов с такого рода границами;
- для решения задачи функциональной диагностики мозга было введено понятие «детекторов»:

- детектор – это ограниченная замкнутая область на внешней границе исследуемого объекта;
- в процессе моделирования сохраняется информация о весе фотонов, прошедших сквозь каждый детектор, а также траектории их движения внутри объекта;

- предложены параллельные версии адаптированного алгоритма для систем с общей и распределенной памятью.

2. Оптимальный алгоритм поиска пересечений

Моделирование распространения зондирующего излучения в биологических тканях методом Монте-Карло требует существенных вычислительных ресурсов, а в случае учета сложной геометрии границ слоев эта проблема становится еще более актуальной в силу необходимости многократного поиска пересечений фотонов с границами слоев среды. Следует отметить, что реальную геометрию исследуемых объектов можно получить посредством МРТ. Примером может служить голова человека, состоящая из таких слоев, как кожа головы, череп, цереброспинальная жидкость, серое вещество и белое вещество головного мозга.

Поиск пересечений луча с набором примитивов хорошо изучен в области компьютерной графики, где предложены различные алгоритмы такого поиска. В рамках данной работы полезным является дополнительное исследование существующих алгоритмов с целью определения наиболее эффективного именно для задачи моделирования распространения зондирующего излучения. Данная задача обладает следующими особенностями:

- рассматриваются биологические ткани, состоящие из нескольких слоев;
- каждый слой обладает одной или несколькими границами, граница представляет собой произвольную триангулированную поверхность;
- осуществляется поиск пересечения траектории движения фотона с границами текущего слоя с учетом длины шага, на который смещается фотон в заданном направлении;
- среда для моделирования – головной мозг человека – с точки зрения геометрии представляет собой набор вложенных друг в друга непересекающихся поверхностей, причем аналитическое описание этих поверхностей представляется затруднительным;
- геометрия слоев среды не меняется в процессе моделирования.

В простейшем случае алгоритм поиска пересечений сводится к последовательному перебору всех треугольников, из которых состоят границы, и проверке на пересечение луча с текущим треугольником. Более оптимальным подходом здесь будет замена полного перебора на выбор небольшой группы треугольников, с которыми может пересечься траектория фотона, исходя из заданного направления его движения. Таким образом, в оптимизации нуждаются два этапа: тест на пересечение луча с треугольником и выбор набора треугольников для сокращения числа таких тестов. И если в качестве оптимального алгоритма поиска пересечения луча с треугольником большинство исследователей используют алгоритм, описанный в работе [4], то в плане сокращения числа тестируемых треугольников применяемые подходы различны.

Были рассмотрены такие подходы к сокращению числа тестов на пересечение луча и треугольника, как регулярная и иерархическая сетки, KD-, BSP- и BVH-деревья. Из них для реализации и проведения временных тестов были выбраны наиболее подходящие в рамках описанных выше условий алгоритмы на основе KD- и BVH-деревьев с SAH критерием качества [5, 6]. Также были реализованы и протестированы различные модификации алгоритмов построения данных структур, касающиеся способа выбора

разделяющей плоскости (по границам треугольников, биновый метод с различным числом бинов) и условий остановки построения дерева (минимальное количество треугольников в листе, ограничение на глубину дерева).

Для проведения экспериментов использовались три группы тестов с геометриями, аппроксимирующими границы тканей головного мозга человека:

1) плоскопараллельная геометрия (plane) – 6 плоскостей, каждая состоит из двух треугольников (всего 12 треугольников);
2) квазисферическая геометрия (sphere) – 5 концентрических квазиполусфер, ограниченных плоскостью, сферы аппроксимировались разным числом треугольников. Квазиполусфера представляет собой полусферу, на поверхности которой встречаются выпуклые места и впадины. Такая поверхность не описывается аналитически, не является гладкой или выпуклой и может считаться поверхностью достаточно общего вида;
3) геометрия коры головного мозга человека (cortex), полученная посредством данных МРТ, состоит из 686 348 треугольников.

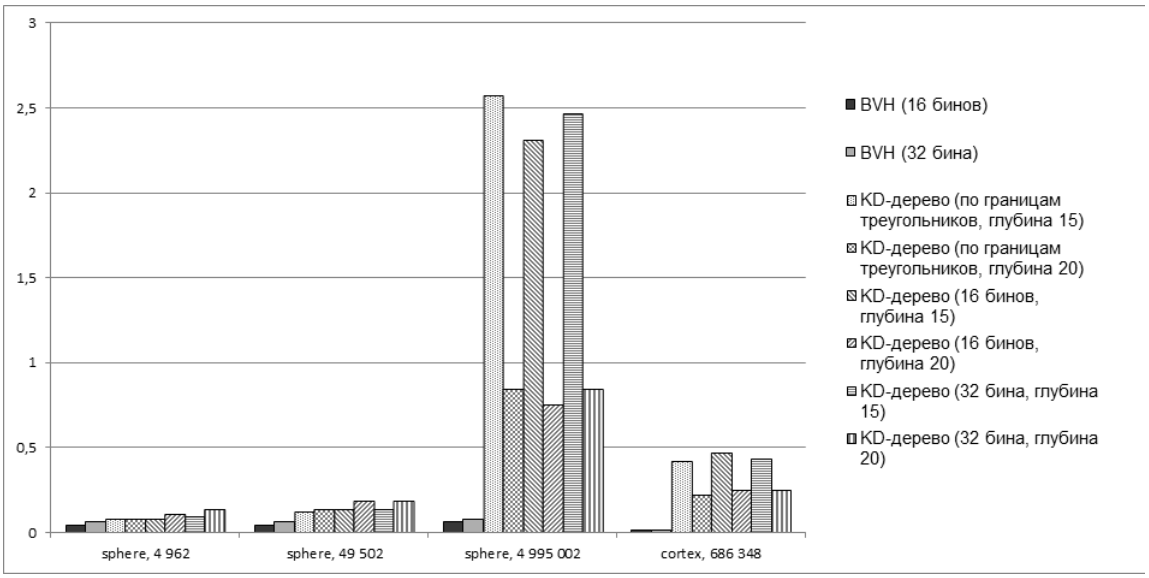


Рис. 2. Сравнение реализаций алгоритма поиска пересечений с учетом шага фотона

Эксперименты проводились на компьютере с центральным процессором Intel Core i5 2410M 2.30 ГГц, 4 ГБ оперативной памяти и операционной системой Microsoft Windows 7 x64. Результаты экспериментов показаны на рис. 2.

Результаты тестов показывают, что алгоритм на основе BVH-деревьев в большинстве случаев является лучшим в рамках описанных выше условий.

3. Эффективность и масштабируемость параллельной реализации

Как уже отмечалось выше, алгоритм моделирования распространения зондирующего излучения требует значительного времени для вычислений. Например, моделирование траекторий 4 млрд. фотонов при длине волны зондирования 830 нм и аппроксимации головного мозга сферической геометрией из 5000 треугольников занимает примерно 672 часа (или 21 день) на одном компьютере в один поток. А значит, присутствует необходимость в параллельном алгоритме моделирования.

В рамках данной работы были реализованы версии параллельного алгоритма для систем с общей и распределенной памятью и проведено измерение времени их работы (рис. 3). Алгоритм для систем с общей памятью тестировался на компьютере с процессором Intel Xeon X5670 2.93 ГГц (6 ядер) и 24 ГБ оперативной памяти. Для тестирования алгоритма с распределенной памятью использовался кластер из 16 узлов, на каж-

дом узле установлено 2 центральных процессора Intel Xeon L5630 2.13 ГГц (4 ядра) и 24 ГБ оперативной памяти. В обоих случаях использовалась операционная система Windows Server HPC Edition.

Результаты проведенных экспериментов показывают, что обе версии параллельного алгоритма обладают масштабируемостью, близкой к линейной, однако эффективность версии для систем с распределенной памятью ниже, что объясняется необходимостью обмена данными по сети между узлами кластера. Эффективность алгоритма для систем с общей памятью составляет около 100%, если брать в расчет только реальные ядра CPU (без учета Hyper-Threading). Для кластерной версии алгоритма этот показатель составляет 45%.

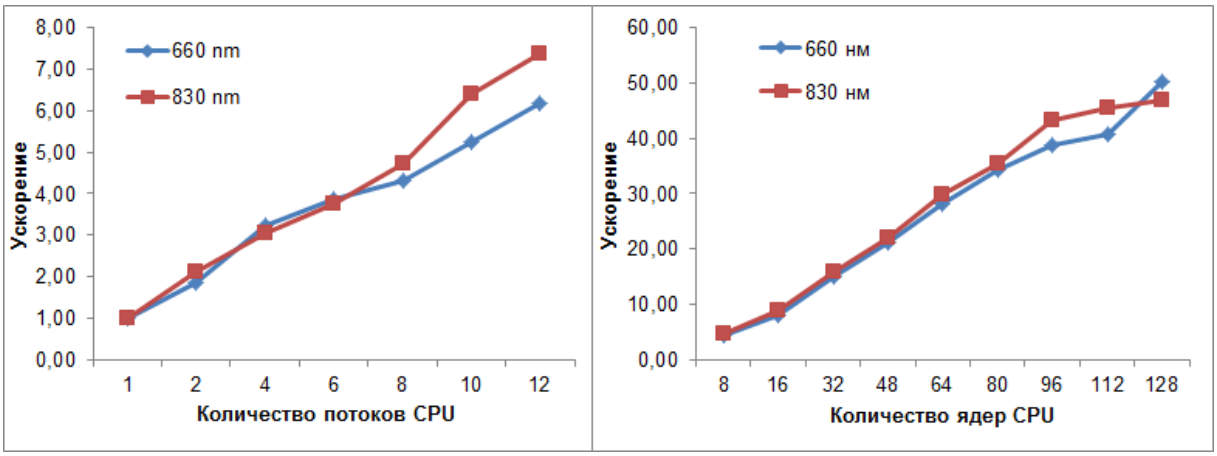


Рис. 3. Ускорение параллельных алгоритмов моделирования распространения излучения для систем с общей (слева) и распределенной (справа) памятью

Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России» (проект № 02.740.11.0839) и гранта Президента РФ МК-1652.2012.2 при организационной поддержке лаборатории информационных технологий ННГУ.

Литература

1. Wang L.H., Jacques S.L., Zheng L.Q. MCML – Monte Carlo modeling of light transport in multilayered tissues // Comput. Meth. Programs Biomed. 1995. V. 47. No. 2. P. 131–146.
2. Boas D.A., Culver J.P., Stott J.J., Dunn A.K. Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head // Opt. Express. 2002. V. 10, No. 3. P. 159–170.
3. Tian P.F., Devor A., Sakadzic S., Dale A.M., Boas D.A. Monte Carlo simulation of the spatial resolution and depth sensitivity of two-dimensional optical imaging of the brain // J. Biomed. Opt. 2011. V. 16, No. 1. P. 13.
4. Möller T., Trumbore B. Fast, minimum storage ray-triangle intersection // J. Graphics Tools. 1997. Vol. 2. P. 21–28.
5. Wald I. On fast Construction of SAH-based Bounding Volume Hierarchies // In Proceedings of the Eurographics Symposium on Interactive Ray Tracing. 2007. P. 33–40.
6. Боголепов Д.К., Сопин Д.П., Турлапов В.Е., Ульянов Д.Я. Построение SAH BVH деревьев для трассировки лучей на GPU в реальном времени // Высокопроизводительные параллельные вычисления на кластерных системах: Доклад XI Всероссийской конференции, Нижний Новгород, Россия, 2-3 нояб. 2011. – С. 301–306.

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ГЛОБАЛЬНОГО ПОИСКА С ОБОБЩЁННОЙ КВАДРАТИЧНОЙ ХАРАКТЕРИСТИКОЙ ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ ГИДРОДИНАМИКИ

В.А. Гришагин, Д.Н. Добряев

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Принятие оптимальных решений, т.е. наилучших в рамках некоторых критериев, является неотъемлемой частью процессов, окружающих человека. Быстрое развитие вычислительных средств, в частности возможности распараллеливания, предоставляют новые возможности для решения оптимизационных проблем, что позволяет существенно увеличить размерность и сложность решаемых задач [1, 2]. Но наряду с использованием новых высокопроизводительных технологий очень важное место занимает методика поиска оптимума. А это, в свою очередь, требует создания новых алгоритмов, адаптированных к специфике решаемой задачи. В настоящей статье предлагается алгоритм глобального поиска, основанный на характеристической структуре решающего правила, в котором используются обобщённые характеристики квадратичного типа, и его параллельная реализация. В силу широкой области применения методов оптимизации актуальным является вопрос использования данного подхода при решении практических задач во всевозможных областях с целью предоставления более удобного и производительного инструментария для изучения тех или иных процессов, в частности возможности увеличения спектра и размерности решаемых прикладных задач. Поэтому основное внимание уделяется использованию изложенных методов в гидродинамике и решению прикладных задач на примере нахождения потокораспределения в гидросистемах. Приведены соответствующие результаты экспериментов.

Обобщённая квадратичная характеристика и сходимость метода

Алгоритм с обобщённой характеристикой квадратичного типа определяется следующим решающим правилом [6]:

$$R(i) = \alpha m(x_i - x_{i-1}) + \beta \frac{(z_i - z_{i-1})^2}{m(x_i - x_{i-1})} - \gamma(z_i + z_{i-1}) + \delta|z_i - z_{i-1}|, \tag{1}$$

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \xi \frac{z_t - z_{t-1}}{m}, \tag{2}$$

где $\alpha > 0, \beta, \gamma, \delta, \xi \geq 0$ – параметры. Величина m вычисляется:

$m = \begin{cases} rM, M > 0 \\ 1, M = 0 \end{cases}$, где $M = \max\limits_{1 \leq i \leq \tau} \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}}$, а $r > 1$ – параметр метода.

Метод сходится, если параметры удовлетворяют следующим условиям [6]:

$$2\sqrt{\alpha\beta} + \delta > \gamma, \beta > \alpha, \xi < \frac{r}{2} \text{ или } 2\sqrt{\alpha\beta} + \delta \geq \gamma, \beta \leq \alpha, \xi < \frac{r}{2}. \tag{3}$$

Если функция $z(x)$ удовлетворяет условию Липшица, то предельная точка является глобальным минимумом, если выполнено $m > \frac{\gamma L}{\alpha}, \xi < \frac{r}{2}$. (4)

Параллельный алгоритм с обобщённой характеристикой

Распараллеливание решающего правила алгоритма с обобщённой характеристикой заключается в выборе нескольких интервалов (по количеству процессоров) с максимальными характеристиками и независимом вычислении в них значений целевой функции [2]. Пусть имеется $p > 1$ вычислительных узлов.

Шаг 1 (инициализация). Первые $p > 1$ испытаний алгоритма проводятся в произвольных точках $x^1, x^2, \ldots, x^p$ отрезка $[a,b]$. Вычисляются значения $z_i = f(x_i), 1 < i < p$.

Шаг 2 (начало итерации – формирование системы интервалов). Координаты испытаний $x^1, x^2, \ldots, x^k$, $k = k(n)$, перенумеровываются в порядке возрастания:

$$a = x_0 < x_1 < \ldots < x_{\tau-1} < x_\tau = b, \quad (5)$$

где $\tau = k(n) - 1$.

Шаг 3 (вычисление характеристик интервалов). Каждому интервалу $(x_{i-1}, x_i), 1 \le i \le \tau$, ставится в соответствие число (характеристика)

$$R(i) = \alpha m(x_i - x_{i-1}) + \beta \frac{(z_i - z_{i-1})^2}{m(x_i - x_{i-1})} - \gamma(z_i + z_{i-1}) + \delta|z_i - z_{i-1}|.$$

Шаг 4 (выбор интервалов с p максимальными характеристиками). Среди характеристик $R(i)$ выбираются p максимальных и соответствующие им интервалы.

Шаг 5 (формирование координат новых испытаний). В выбранных интервалах формируются точки испытаний $x^{k+s} = \frac{x_{t_s} + x_{t_s-1}}{2} - \xi \frac{z_{t_s} - z_{t_s-1}}{m}$ для $1 \le s \le p$.

Шаг 6 (проверка условия остановки). Если выполняется условие остановки

$$\min_{1 \le s \le p}(x_{t_s} - x_{t_s-1}) \le \varepsilon, \quad (6)$$

то вычисления заканчиваются. В противном случае осуществляется вычисление значений функции в точках $x^{k+1}, \ldots, x^{k+p}$ – каждое испытание на своём процессоре.

В системе ***PSoDI*** [3] реализована возможность использования параллельного алгоритма глобального поиска с обобщённой характеристикой квадратичного типа для решения многомерных многоэкстремальных задач оптимизации в рамках многошаговой схемы на многоядерных архитектурах, где параллельно решаются одномерные подзадачи по последней (n-й) переменной. Также есть возможность распараллеливания по первой переменной, когда параллельно решаются подзадачи размерности n–1.

Примеры использования предложенного подхода распараллеливания алгоритма с обобщённой характеристикой при решении прикладных задач стационарного потокораспределения в гидросистемах

Спектр применения методов оптимизации очень широк. Одной из задач, решение которой можно свести к поиску экстремума функции, является задача нахождения стационарного потокораспределения в гидросистемах [4]. Существенным преимуществом использования методики является тот факт, что минимумы совпадают с устойчивыми состояниями равновесия [7], что исключает проблему определения устойчивости.

Покажем примеры решения задачи нахождения равновесных режимов системы циркуляции теплоносителя (СЦТ) ядерной энергетической установки, являющейся

частным видом гидросистемы. В силу большой размерности и многоэкстремальности подобных задач целесообразно применять параллельные методы поиска оптимума.

На рис. 1 представлена расчетная модель типовой системы циркуляции.

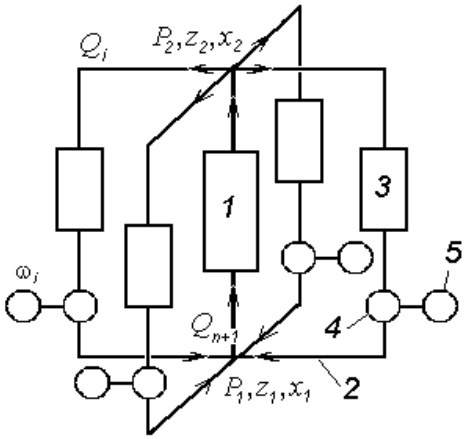


Рис. 1

Теплоноситель, нагревающийся в активной зоне реактора 1, поступает по трубопроводам в параллельно работающие теплообменные петли 2 и после охлаждения в теплообменниках 3 снова подается в активную зону. Течение теплоносителя обеспечивается работой циркуляционных насосов 4, снабженных электродвигателями 5. Стрелками показано направление течения теплоносителя.

Состояния равновесия данной системы являются стационарными точками соответствующей функции Ляпунова [5]. Для рассматриваемой СЦТ задача нахождения устойчивых состояний равновесия в шестимерном фазовом пространстве сводится к поиску минимумов этой функции.

Минимизируемая функция для данной схемы имеет вид [4]

$$f(x) = \sum_{i=1}^{4} S_i(x_i) + S_5(x) \text{ , здесь}$$

$$S_i(x_i) = \begin{cases} \dfrac{a_{1i}x_i^3}{3} - \dfrac{b_i\omega_i x_i}{2} - c_i\omega_i^2 x_i, \ x_i \ge 0; \\ -\dfrac{a_{2i}x_i^3}{3} - \dfrac{b_i\omega_i x_i}{2} - c_i\omega_i^2 x_i, \ x_i < 0; \end{cases}, \ i = \overline{1,4}, \quad S_5(x) = \begin{cases} \dfrac{d_1}{3}\left(\sum_{j=1}^{4} x_j\right)^3, \ \sum_{j=1}^{4} x_j \ge 0; \\ -\dfrac{d_2}{3}\left(\sum_{j=1}^{4} x_j\right)^3, \ \sum_{j=1}^{4} x_j < 0; \end{cases}$$

где x_i – расход теплоносителя в i-й петле, $i = \overline{1,4}$; функции $S_i(x_i)$, $i = \overline{1,4}$, представляют собой интегралы от гидравлических характеристик участков с теплообменниками, $S_5(x)$ – интеграл от гидравлической характеристики участка активной зоны; a_{1i}, a_{2i}, b_i, c_i, d_1, d_2 – положительные коэффициенты, величины которых зависят от гидравлических сопротивлений участков и гидравлических характеристик насосов; ω_i – угловая скорость вращения ГЦН в i-й петле.

Расчеты проводились при следующих заданных параметрах:

$\omega^0_1 = \omega^0_2 = \omega^0_3 = \omega^0_4 = 1{,}0$; $a_{1i} = 2$; $a_{2i} = -4$; $b_i = 3$; $c_i = 6$; $d_1 = -d_2 = 0{,}4$.

Поиск минимумов 6-мерной функции $f(x)$ был проведен в системе ***PSoDI*** с использованием параллельного алгоритма с обобщённой характеристикой в рамках многошаговой схемы редукции размерности на восьми ядрах (2 четырёхъядерных процессора *Intel(R) Xeon(R) CPU E5504 @ 2.0GHz.*) Были найдены следующие устойчивые состояния, совпадающие с экстремумами:

$f_{\min 1,2,3,4}$ =–19.14; x_1 = –0.72, x_2 = 1.52, x_3 = 1.52, x_4 = 1.52 (остальные 3 точки – перестановка элементов вектора X);

$f_{\min 5}$ = –18.85; x_1 = 1.04, x_2 = 1.04, x_3 = 1.04, x_4 = 1.04.

На следующем рисунке представлена расчетная схема СЦТ, соответствующая установке ВВЭР-440 и содержащая 6 параллельных подключенных к реактору петель циркуляции.

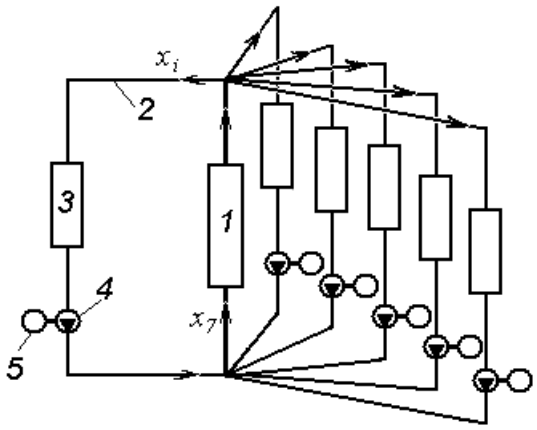


Рис. 2

Нарушение нормальной работы СЦТ недопустимо с точки зрения безопасности реактора.

Минимизируемая функция для данной схемы имеет вид [7]

$$f(x)=\sum_{i=1}^{6} S_i(x_i)+S_7(x)\text{ , здесь}$$

$$S_i(x_i)=\begin{cases}\dfrac{a_{1i}x_i^3}{3}-\dfrac{b_i\omega_i x_i}{2}-c_i\omega_i^2 x_i,\ x_i\ge 0;\\ -\dfrac{a_{2i}x_i^3}{3}-\dfrac{b_i\omega_i x_i}{2}-c_i\omega_i^2 x_i,\ x_i<0;\end{cases},\ i=\overline{1,6},\quad S_7(x)=\begin{cases}\dfrac{d_1}{3}\left(\sum\limits_{j=1}^{6}x_j\right)^3,\ \sum\limits_{j=1}^{6}x_j\ge 0;\\ -\dfrac{d_2}{3}\left(\sum\limits_{j=1}^{6}x_j\right)^3,\ \sum\limits_{j=1}^{6}x_j<0.\end{cases}$$

Расчеты проводились при следующих заданных параметрах:

$$\omega_1=\omega_2=\omega_3=\omega_4=\omega_5=\omega_6=1; a_{1i}=1.4;$$

$$a_{2i}=4; b_i=4; c_i=6; d_1=d_2=0.03; i=\overline{1,6}.$$

Были найдены следующие устойчивые состояния:

$$f_{\min 1-6}=-84.30;\ x_1=-0.90; x_2=2.93; x_3=2.93; x_4=2.93; x_5=2.93; x_6=2.93$$

(остальные 5 минимумов получаются перестановкой ветора X);

$$f_{\min 7}=-87.58.\ x_1=2.55; x_2=2.55; x_3=2.55; x_4=2.55; x_5=2.55; x_6=2.55.$$

Глобальный минимум этой функции соответствует равновесному режиму циркуляции. Наличие других режимов работы, соответствующих остальным минимумам функции $f(x)$, недопустимо с точки зрения безопасности работы, и задача конструктора состоит в выборе гидравлических характеристик, при которых состояние равновесия минимально и устойчиво «в целом». Проверка полученных решений, основанных на результатах работ [4], показала их правильность.

Заключение

В настоящей работе предложен параллельный алгоритм глобального поиска, основанный на характеристической структуре решающего правила, в котором используются обобщённые характеристики квадратичного типа, и показаны примеры решений при-

кладных задач о нахождении равновесных режимов работы систем циркуляции теплоносителя ядерных энергетических установок.

Работа выполнена при финансовой поддержке РФФИ (грант № 11-01-00682-а), Совета по грантам Президента Российской Федерации (грант № НШ-1960.2012.9), Министерства образования и науки Российской Федерации (государственное соглашение о предоставлении гранта № 14.B37.21.0878).

Литература

1. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация: Учебное пособие. Нижний Новгород: Изд-во ННГУ, 2007. – 489 с.
2. Sergeyev Ya.D., Grishagin V.A. Parallel asynchronous global search and the nested optimization scheme // Journal of Computational Analysis & Applications. 3(2), 2001. P. 123–145.
3. Добряев Д.Н., Гришагин В.А. Программная система конструирования и исследования характеристических методов многоэкстремальной оптимизации // Материалы конференции «Технологии Microsoft в теории и практике программирования», Н.Новгород, 3-4 апреля, 2007. С. 68-71.
4. Смирнов Л.В., Данилова Н.В. Основы прикладной аналитической гидромеханики напорного течения несжимаемой жидкости: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2009. – 65 с.
5. Смирнов Л.В., Гришагин В.А., Добряев Д.Н., Данилова Н.В. Применение прикладной аналитической гидромеханики и методов принятия оптимальных решений в задаче нахождения потокораспределения в гидросистемах // Вестник ННГУ. 2010, №2(1). С. 144–154.
6. Добряев Д.Н., Данилова Н.В. Исследование алгоритмов глобального поиска с обобщённой характеристикой квадратичного типа // Вестник ННГУ. 2011, №3(2). С. 25–32.
7. Данилова Н.В., Добряев Д.Н. Применение прикладной аналитической гидромеханики и методов принятия оптимальных решений в задаче нахождения устойчивых равновесных режимов гидросистем // Материалы X конференции ВСФПТПМ, 24-30 августа 2011 г., г. Н.Новгород.

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ЭВОЛЮЦИИ АКУСТИЧЕСКИХ ВОЛН В МЕДИКО-БИОЛОГИЧЕСКИХ СРЕДАХ

С.Н. Гурбатов, И.Ю. Демин, Н.В. Прончатов-Рубцов

Нижегородский госуниверситет им. Н.И. Лобачевского

Известно, что распространение интенсивных акустических волн в средах со слабой дисперсией описывается уравнением Бюргерса (УБ). Для акустических волн в мягких биологических тканях необходимо учитывать слабую дисперсию и нелинейное затухание. В этом случае аналитическое решение эволюционного УБ удается получить лишь для немногих частных случаев (например, уравнение Римана), поэтому необходимо использовать также численные методы.

В докладе рассмотрена численная схема решения УБ с использованием быстрого преобразования Фурье (БПФ) – спектральный метод. Это позволило обойти трудности вычислений как нелинейного оператора, так и линейного оператора, описывающего затухание и дисперсию среды. В качестве примера реализации схемы численного решения УБ выбрана операционная среда программирования LabVIEW. При численном моделировании на основе этого уравнения создана итерационная схема, которая позволила вычислить спектр и профиль волны на любых расстояниях от источника. Отметим, что данная схема решения УБ удобна для численного анализа эволюции как детерминированных, так и шумовых сигналов, а также может быть легко модифицирована для описания цилиндрических и сферических волн. Преимущество описанного метода численного моделирования перед обычной схемой решения уравнений параболического типа состоит в том, что он гораздо лучше подходит для расчета распространения в случае больших чисел Рейнольдса.

Проведено сравнение результатов численного моделирования с экспериментальными данными по распространению нелинейных акустических волн в мягких биологических тканях. Проведен анализ высших спектральных характеристик (биспектров) нелинейных акустических волн, и исследована возможность их использования в задачах диагностики мягких биологических тканей (акустическая томография).

Работа выполнена при поддержке гранта Правительства РФ № 11.G34.31.0066, гранта РФФИ № 11-02-00774-а и гранта поддержки ведущих научных школ НШ-333.2012.2.

ОБ АКТУАЛЬНОСТИ ИССЛЕДОВАНИЯ И РАЗВИТИЯ СОВРЕМЕННЫХ МОДЕЛЕЙ РАСПРЕДЕЛЕННОГО И ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

И.Г. Данилов

Южный федеральный университет, Ростов-на-Дону

Рассматриваются ряд проблем современных моделей распределённого и параллельного программирования и возможные направления путей их решения.

Введение

В настоящее время развитие суперкомпьютерной техники идёт в направлении увеличения мощности отдельных вычислительных узлов за счёт внедрения многоядерных процессорных технологий, наращивания количества вычислительных узлов в системе и совершенствования характеристик межузловых сетевых соединений. В связи с этим широко применяемые модели параллельного и распределённого программирования, которые используют *двусторонние коммуникации*, т.е. *обмен сообщениями*, могут оказаться неэффективными при программировании задач для систем, состоящих из десятков и сотен тысяч слабосвязанных ядер. Это объясняется тем, что:

1) для двустороннего обмена требуется участие обеих сторон: как отсылающей, так и принимающей, что приводит к усложнению программирования и возможным потерям производительности;

2) количество памяти, доступной вычислительному процессу, ограничено размером локальной памяти вычислительного узла, на котором выполняется этот процесс;

3) для моделей, использующих обмен сообщениями, характерно *фрагментированное представление о вычислениях* [1], что приводит к сложности распределения составных и комплексных структур данных по вычислительным процессам, необходимости написания кода по проверке граничных условий и возрастанию количества обменов сообщениями между вычислительными процессами;

4) при организации двустороннего обмена возникают накладные расходы, связанные с формированием, упорядочением и буферизацией сообщений, что оказывается узким местом при возрастании количества обменов сообщениями.

Другая популярная модель параллельного программирования, которая основана на взаимодействии через *общую разделяемую память*, предлагает более простой и продуктивный по сравнению с моделью передачи сообщений подход [2] к созданию параллельных приложений, так как программисту нет необходимости заботиться о локальности доступа к данным (*глобальное представление о вычислениях* [1]). Однако эта модель в свою очередь оказывается неэффективной при решении масштабируемых задач в силу сложности эффективной реализации (как аппаратной, так и программной, виртуальной) разделяемой памяти для больших вычислительных систем.

Модель разделённого глобального адресного пространства

Развитие и широкое распространение новых вычислительных архитектур и коммуникационных сетей в последние годы привело к активизации исследований в области совершенствования существующих моделей и алгоритмов параллельного и распреде-

лённого программирования с учетом новых технических возможностей. Одной из таких недавно появившихся моделей программирования является модель разделённого глобального адресного пространства (англ. *Partitioned Global Address Space*, PGAS), в которой используется принцип взаимодействия на основе односторонних коммуникаций, позволяющих одной части системы взаимодействовать с другой без активного её привлечения. Односторонние коммуникации эффективно реализуются в современных сетях для суперкомпьютеров с использованием протокола удалённого доступа в память (англ. *Remote Direct Memory Access*, RDMA).

Появление парадигмы PGAS – попытка разработать гибридную модель, объединяющую преимущества передачи сообщений и взаимодействия через разделяемую память. В реализациях PGAS, таких как расширение для языка программирования C – UPC [3] (англ. *Unified Parallel C*) или Fortran – Coarray Fortran (CAF), которые можно назвать *языками PGAS первого поколения*, распределение данных в вычислительной системе производится явно программистом с учётом локальности последующего доступа к ним. При этом инструкции доступа к данным генерируются компилятором неявно; программисту необходимо позаботиться только о синхронизации с целью получения непротиворечивых результатов. Данные языки используют низкоуровневые сетевые коммуникационные библиотеки PGAS, такие как GASNet [4] и ARMCI, которые предоставляют различные функции для односторонних коммуникаций, передачи сообщений, операций синхронизации и управления разделённым глобальным адресным пространством.

Основные проблемы современных моделей распределённого и параллельного программирования и возможные пути их решения

Широко распространённые реализации модели передачи сообщений (прежде всего MPI, от англ. *Message Passing Interface*) и модели PGAS (например, UPC) сравнимы по производительности [5]. Каждая реализация имеет свои отличительные достоинства и недостатки, но в то же время существует один общий для всех реализаций недостаток: плохая поддержка *неструктурированного параллелизма* – общей и наименее упорядоченной формы параллелизма [6]. Неструктурированный параллелизм предполагает возможность динамического создания (*динамический параллелизм*) вычислительных процессов, работающих по различным алгоритмам. В то же время как модель передачи сообщений, так и языки PGAS первого поколения реализуются на основе SPMD-подхода, в котором программа состоит из фиксированного набора вычислительных процессов, количество которых в течение её выполнения не изменяется (*статический параллелизм*), что приводит к плохой балансировке нагрузки вычислительной системы.

Потребность выполнения различных алгоритмов в программе в зависимости от некоторых условий приводит к необходимости использовать более общий подход по сравнению с SPMD, который называется MPMD и реализуется, например, в модели общей памяти на основе многопоточности, которая предусматривает специальные примитивы для динамического управления программными потоками. В свою очередь, в рассматриваемых моделях MPMD-программирование осуществляется с помощью определенных параметров вычислительных процессов системы (ранг, коммуникатор – для MPI, номер потока – для UPC) и соответствующего ветвления, что приводит к усложнению и плохой читаемости кода программы. Кроме того, остается нерешенным вопрос возможной разбалансировки нагрузки вычислительной системы. Для частичного решения данных проблем, которые возникают из-за отсутствия поддержки динамического параллелизма, применяют следующие методы.

1. *Гибридизацию* моделей программирования: передачу сообщений совмещают с многопоточными реализациями модели общей памяти (например, с OpenMP [7]) или с

моделью параллелизма задач (англ. *task parallelism*). Аналогично и для модели PGAS: например, для UPC существуют специальные расширения [8], использующие модель параллелизма задач. Стоит отметить, что при данном подходе вопрос разбалансировки нагрузки решается только в рамках одного вычислительного узла системы.

2. *Виртуализацию* процессов – подход, при котором вычислительные процессы модели программирования отображаются определённым образом на программные процессы или потоки операционной системы и могут *мигрировать* на любые доступные в системе вычислительные узлы. Для реализаций модели передачи сообщений данный подход поддерживается как на уровне операционной системы (MOSIX [9]), так и на уровне программных библиотек; можно отметить, что к модели PGAS виртуализация процессов трудно применима, т.к. для миграции процессов требуется сложная поддержка механизмов управления и доступа к глобальной разделяемой памяти со стороны компилятора и исполняющей среды.
3. *Расширение* спецификации, описывающей стандарт реализации модели программирования, с целью внесения новой функциональности для динамического порождения процессов/потоков. Например, такая функциональность появилась во второй версии спецификации для стандарта MPI, однако её использование существенно усложняет программирование. Спецификации для реализаций языков PGAS первого поколения не поддерживают конструкции динамического создания потоков.

Проблема отсутствия поддержки динамического параллелизма в языках PGAS первого поколения, наряду с рядом других проблем [1], привела к возникновению новых языков PGAS – *языков второго поколения*, разрабатываемых по инициативе американского агентства передовых оборонных исследовательских проектов (англ. *Defense Advanced Research Projects Agency*) в рамках программы по созданию высокопродуктивных компьютерных систем HPCS (англ. *High Productivity Computing Systems*). Наиболее известными среди них являются: Chapel [1] и X10. В новых языках HPCS параллелизм задач – для X10 или принцип многопоточности – для Chapel лежат в основе реализации. Кроме того, с помощью различных языковых конструкций в них поддерживаются и другие виды параллелизма, например *параллелизм по данным*.

В моделях параллельного и распределённого программирования, которые построены на понятии общей разделяемой памяти, память используется для взаимодействия вычислительных процессов друг с другом, что может привести к проблеме *гонки по данным*. Эта проблема особенно актуальна для неструктурированного параллелизма, когда доступ к общим данным в общем случае непредсказуем. Для избежания гонок требуется согласовать доступ вычислительных процессов к данным с помощью явной синхронизации, средства для которой предоставляются реализациями моделей программирования. Традиционно эти средства построены на различных *блокирующих методах* (например, семафорах) и позволяют решить *задачу о взаимном исключении процессов* (англ. *mutual exclusion*), в которой доступ к разделяемому ресурсу (памяти) в определённый промежуток времени ограничен строго одним процессом или же процессом определённого типа (например, процессом-читателем). В свою очередь, использование блокирующих методов синхронизации порождает ряд других проблем, таких как *взаимоблокировка*, что требует определённой квалификации и трудозатрат со стороны программиста. Для распределённых систем существует большое количество алгоритмов [10], которые решают задачу *распределённого взаимного исключения* (англ. *distributed mutual exclusion*) и используются для реализации примитивов распределённой синхронизации, предоставляемых в том числе и языками модели PGAS. Существующие алгоритмы распределённого взаимного исключения построены на основе передачи сообщений по сети и блокирующих принципов, ключевая характеристика кото-

рых – возможная задержка для входа в *критическую область* – возрастает в зависимости от числа процессов в системе и частоты обращения к разделяемым данным, что приводит к плохой масштабируемости параллельной программы в целом.

Необходимость решения проблем блокирующих методов синхронизации привела к возникновению исследований в области *неблокирующей синхронизации* (англ. *non-blocking synchronisation*). Предложены различные методы неблокирующей синхронизации и соответствующие программные/аппаратные абстракции [11], одной из которых является активно исследуемая в настоящее время *транзакционная память* (англ. *transactional memory*, TM). Транзакционная память базируется на методологии теории транзакционной обработки данных, разработанной в 70–80-х годах прошлого столетия такими исследователями, как K. P. Eswaran, C. H. Papadimitriou, P. A. Bernstein, H. T. Kung и др. Методы транзакционной обработки, применяемые в базах данных для поддержания данных в согласованном состоянии, были обобщены M. Herlihy и J. E. B. Moss, которые выдвинули идею [12] использования подобных методов для согласования доступа нескольких процессоров к разделяемой памяти. Предложенные Herlihy и Moss принципы аппаратной реализации TM (англ. *hardware TM*, HTM) были развиты N. Shavit и D. Touitou в работе [13], впервые описывающей принципы программной реализации TM (англ. *software TM*, STM).

Концепция транзакций тесно связана с задачей взаимного исключения. Как и блокирующие алгоритмы синхронизации, алгоритмы транзакционной памяти позволяют последовательно упорядочить – *сериализовать* – инструкции доступа к памяти параллельно выполняемых вычислительных процессов, тем самым обеспечив непротиворечивость результатов их работы. Основное отличие состоит в том, что транзакции – помеченные как атомарные (англ. *atomic section*) участки кода программы, в которых происходит обращение к общему ресурсу (памяти), – выполняются всеми процессами одновременно, в то время как в критической области в общем случае может находиться только один процесс. Если во время выполнения атомарных секций происходит конфликт по данным между двумя транзакциями, то одна из них откатывается на начало выполнения секции, а другая завершается и выходит из секции.

Большая часть зарубежных работ в данной области посвящена системам с общей памятью [14]; более того, ведущие производители процессорных устройств уже заявили поддержку механизмов транзакционной памяти в ряде моделей процессоров будущих поколений [15]. Для систем с распределённой памятью существует всего несколько рабочих программных реализаций [16]. В настоящее время разработчики языков HPCS исследуют возможности применения механизмов транзакционной памяти для программирования распределённых систем: например, в языке Chapel заложена специальная конструкция атомарной секции *atomic*, в основе которой модель глобальной транзакционной памяти [17].

Заключение

Можно отметить следующие актуальные направления исследований в области моделей распределённого и параллельного программирования:

1) отказ от модели взаимодействия на основе обмена сообщениями и переход к односторонним коммуникациям;
2) исследование и развитие неблокирующих методов синхронизации, таких как транзакционная память;
3) исследование и развитие моделей параллелизма задач и данных.

Литература

1. Chamberlain B., Callahan D., Zima H. Parallel Programmability and the Chapel Language // Int. J. High Perform. Comput. Appl. 2007. V. 21, no. 3. P. 291–312.

2. Hochstein L., Carver J., Shull F. et al. Parallel Programmer Productivity: A Case Study of Novice Parallel Programmers // Proceedings of the 2005 ACM/IEEE conference on Supercomputing. SC ’05. Washington, DC, USA: IEEE Computer Society, 2005. P. 35.
3. El-Ghazawi T., Carlson W., Sterling T., Yelick K. UPC: Distributed Shared Memory Programming. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, 2005.
4. Bonachea D. GASNet Specification, v1.1: Tech. rep. Berkeley, CA, USA: 2002. URL: http://techreports.lib.berkeley.edu/accessPages/CSD-02-1207 (дата обращения: 26.09.2012).
5. Mallon D. A., Taboada G. L., Teijeiro C. et al. Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures // Proceedings of the 16th European PVM/MPI Users’ Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Berlin, Heidelberg: Springer-Verlag, 2009. P. 174–184.
6. Sutter H., Larus J. Software and the Concurrency Revolution // Queue. 2005. – September. Vol. 3, no. 7. P. 54–62.
7. Smith L., Bull M. Development of mixed mode MPI / OpenMP applications // Sci. Program. 2001. – August. Vol. 9, no. 2,3. P. 83–98.
8. Dinan J., Krishnamoorthy S., Larkins D. B. et al. Scioto: A Framework for Global-View Task Parallelism // Proc. of the 2008 37th International Conf. on Parallel Processing. ICPP ’08. Washington, DC, USA: IEEE Computer Society, 2008. P. 586–593.
9. Barak A., La’adan O. The MOSIX multicomputer operating system for high performance cluster computing // Future Gener. Comput. Syst. 1998. – March. Vol. 13, no. 4-5. P. 361–372.
10. Attiya H., Welch J. Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition). John Wiley Interscience, 2004. – March. ISBN: 0-471-453242.
11. Fraser K. Practical lock-freedom: Tech. Rep. UCAM-CL-TR-579: University of Cambridge, Computer Laboratory, 2004. – February. – [http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-579.pdf].
12. Herlihy M., Moss J. E. B. Transactional memory: architectural support for lock-free data structures // SIGARCH Comput. Archit. News. 1993. – May. Vol. 21,no. 2. P. 289–300.
13. Shavit N., Touitou D. Software transactional memory // Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing. PODC ’95. New York, NY, USA: ACM, 1995. P. 204–213.
14. Grahn H. Transactional Memory // In: J. Parallel Distrib. Comput. - 2010. - Vol. 70 (10). - P. 993-1008.
15. Merritt R. IBM plants transactional memory in CPU // Intel Software Network – [http://www.eetimes.com/electronics-news/4218914/IBM-plants-transactional-memory-in-CPU].
16. Saad M. M., Ravindran B. Transactional Forwarding Algorithm: Technical Report // ECE Dept., Virginia Tech, January 2011.
17. Sridharan S., Vetter J. S., Kogge P. M. Scalable Software Transactional Memory for Global Address Space Architectures: Tech. Rep. FTGTR-2009-04: Future Technologies Group, Oak Ridge National Lab, 2009. – November.

О РЕШЕНИИ ЗАДАЧ МАГНИТОДИНАМИКИ И КОГЕРЕНТНЫХ ПРОЦЕССОВ В НАНОМАГНИТНЫХ СТРУКТУРАХ НА СУПЕРКОМПЬЮТЕРЕ

А.Г. Деменев, Е.К. Хеннер, Т.С. Белозерова, В.К. Хеннер, П.В. Харебов

Пермский государственный национальный исследовательский университет
E-mail: a-demenev@mail.ru

Сделан анализ потенциала распараллеливания исходных кодов программного обеспечения Spins и MagnetoDynamics, предназначенных для исследования динамики многочастичных систем наномагнитов. Создана параллельная версия программы MagnetoDynamics-F – моделирующего программного кода Белозеровой Т.С. на языке Fortran, использующая интерфейс прикладного программирования OpenMP. Получены оценки ускорения и эффективности реализованных алгоритмов на типичных задачах. Созданный параллельный код обеспечивает проведение исследований: возможности регулирования времени переключения магнитного момента наноструктуры; роли фактора геометрии нанокристалла в свойствах сверхизлучения с 1-, 2- и 3-мерными объектами; частных решений системы уравнений для описания магнитодинамики наноточки, индуктивно связанной с пассивным резонатором; зависимости нелинейного решения от начальной ориентации магнитного момента с целью нахождения конфигураций, в которых сверхизлучение и радиационное затухание максимальны.

Наномагнитная структура обычно рассматривается как система взаимодействующих наноразмерных частиц, обладающих магнитным моментом. Перспективным представляется возможное использование больших скоростей когерентных процессов таких наномагнитов в различного рода датчиках и переключателях, особенно в наноустройствах, где традиционные механизмы релаксации выражены очень слабо. Сверхизлучение является одним из наблюдаемых явлений в наномагнитах. Сверхизлучение происходит из-за когерентизации спиновых переходов, при которых эффективные спин-спиновые взаимодействия не уменьшаются с расстоянием. Обычно это реализуется помещением образца в пассивный резонатор. В результате из-за когерентности процессов релаксации шкала времени для таких процессов обратно пропорциональна числу спинов – необычный феномен для макроскопической физики, а излучаемая мощность пропорциональная не числу спинов, а их квадрату.

Представленная работа нацелена на развитие и применение вычислительных и информационных технологий в моделировании многомасштабной молекулярной динамики многочастичных систем наномагнитов. Моделируемые объекты – высокоспиновые наномолекулы, нанокластеры и молекулярные кристаллы. В физике магнитных явлений понятие «спин» по существу отождествляется с понятием «магнитный момент».

Основные трудности описания поведения многоспиновых систем коренятся в межчастичных взаимодействиях, связывающих все спины образца. Для систем, состоящих из большого числа частиц, диагонализация точного квантового гамильтониана невозможна за разумное время, т.к. вычислительная сложность решения квантомеханической задачи растет экспоненциально с числом частиц. При численном моделировании спины могут трактоваться как «классические»: движение магнитного момента каждой частицы описывается одним классическим вектором. Исследование реалистичных моделей спиновой динамики приводит к необходимости решения задач, вычислительная слож-

ность которых нелинейно растет с увеличением числа структурных элементов и времени наблюдения за системой.

К началу 2011 года авторский коллектив имел исходные программные коды, моделирующие магнитодинамику и когерентные процессы спиновой динамики, созданные в среде Borland C++ Builder Харебовым П.В. (программа Spins) и среде Borland Delphi Белозеровой Т.С. (программа MagnetoDynamics). Эти программные коды реализовывали только последовательные алгоритмы и компилировались только под MS Windows. Эти ограничения не позволяли эффективно использовать высокопроизводительные вычислительные системы в исследованиях магнитодинамики и когерентных процессов в наномагнитных структурах.

Распараллеливание алгоритмов и использование суперкомпьютеров позволяет значительно увеличить число структурных элементов и диапазон времен эволюции исследуемых систем, доступных для изучения. При этом приходится учитывать, что параллельные вычисления требуют специальных исследований на предмет обеспечения:

- корректности результатов, т.к. классическая теория сходимости неприменима;
- эффективности отображения вычислительных алгоритмов на современные параллельные компьютерные архитектуры.

При анализе потенциала распараллеливания исходных кодов программного обеспечения Spins и MagnetoDynamics были использованы методы анализа информационной структуры и асимптотического анализа сложности алгоритмов [1].

Были получены формулы для асимптотической оценки ускорения и эффективности многопоточного распараллеливания алгоритмов, реализованных в программах Spins и MagnetoDynamics, на типичных задачах:

- теоретические (по Амдалу);
- полуэмпирические (с учетом накладных расходов на поддержку многопоточности на мультиядерных процессорах).

Было показано, что с ростом числа моделируемых наночастиц вычислительная сложность алгоритмов, реализованных в Spins и в MagnetoDynamics, растет асимптотически кубично.

Было показано, что с ростом числа моделируемых наночастиц требования к оперативной памяти растут:

- асимптотически квадратично у алгоритма, реализованного в Spins;
- асимптотически линейно у алгоритма, реализованного в MagnetoDynamics.

Было показано, что если накладные расходы на многопоточное распараллеливание асимптотически растут так же, как требования к оперативной памяти, то с ростом числа моделируемых частиц возможен рост масштабируемости распараллеливания.

Анализировалась практика аналогичного переноса программного обеспечения из одной среды программирования в другую. Показано, что потенциал распараллеливания исходных кодов моделирующего программного обеспечения трудно реализовать на практике, т.к. среды и библиотеки программирования, использованные при разработке Spins и MagnetoDynamics, работают только в Windows, а большинство суперкомпьютеров использует операционную систему Linux. Перенос Spins на языке Borland C++ в кросс-платформенную среду разработки трудно реализовать из-за того, что используется не кросс-платформенная библиотека Microsoft .NET 4.0. Перенос MagnetoDynamics на языке Borland Delphi в кросс-платформенную среду разработки трудно реализовать из-за того, что язык Borland Delphi не имеет международного стандарта.

Поэтому была создана программа MagnetoDynamics-F – параллельная версия моделирующего программного кода Белозеровой Т.С. на языке Fortran. Для распараеливания использован интерфейс прикладного программирования OpenMP. Получена полу-

эмпирическая формула для асимптотической оценки ускорения и эффективности распараллеливания на мультиядерных процессорах.

В программе MagnetoDynamics-F были реализованы важные применения для исследований когерентных процессов[2]:

- возможности регулирования времени переключения магнитного момента наноструктуры, используя большую скорость когерентных процессов;
- роли фактора геометрии нанокристалла для свойств сверхизлучения с 1-, 2- и 3-мерными объектами;
- частных решений системы уравнений для описания магнитодинамики наноточки, индуктивно связанной с пассивным резонатором, для случаев слабой (стационарная прецессия) и сильной (инверсия намагниченности) неравновесности;
- зависимости нелинейного решения (сильная неравновесность) от начальной ориентации магнитного момента наноточки, чтобы найти конфигурации, в которых сверхизлучение и радиационное затухание максимальны.

При исследовании квазимолекулярных магнитных кристаллов использованы два подхода. В первом высокоспиновая молекула (атомный кластер) рассматривается как единая (бесструктурная) парамагнитная частица. В этом случае молекулярный (кластерный) кристалл можно считать суперпарамагнетиком с невысокой магнитной анизотропией; влияние последней на сверхизлучение до нас не исследовалось. Магнитодинамика с учетом когерентных эффектов может быть исследована методами, описанными выше. Кристалл должен быть сильно поляризован магнитным полем при очень низкой температуре. Электронная спиновая система может сохранить свою поляризацию в течение часов и может быть перенесена в пассивный резонатор. Обратная связь резонатора с системой взаимодействующих спинов пропорциональна производной по времени полного магнитного момента кристалла. Из-за когерентного движения все спины релаксируют за время, обратно пропорциональное числу спинов.

Во втором подходе квазимолекула представляется как спиновый нанокристалл с внутренними ориентационными степенями свободы. Сравнительно небольшое число (8-12) магнитных центров в квазимолекуле позволяет изучать ее поведение во внешних полях и спектр внутренних возбуждений путем прямого численного моделирования. Хотя уравнение ориентационного движения отдельной наноточки специфично именно для этого объекта, схема учета дипольного взаимодействия очень близка к расчету, планируемому для молекулярных спиновых систем.

Работа выполнена на базе Научно-образовательного центра «Параллельные и распределенные вычисления» (НОЦ ПиРВ) ПГНИУ с использованием уникального оборудования – программно-технического комплекса «Высокопроизводительный SMP-сервер» (приобретено по гранту РФФИ 10-01-05021) и суперкомпьютера «ПГУ-Тесла» (приобретен по проекту «Развитие центра коллективного пользования высокопроизводительными вычислительными ресурсами – НОЦ ПиРВ» Программы развития ПГНИУ). Работа была выполнена при поддержке грантов РФФИ 10-02-96023 - р_урал_а и 11-07-96007 - р_урал_а.

Литература

1. Деменев А.Г. Анализ параллельных вычислительных алгоритмов: Учеб.-метод. пособие / Перм. ун-т. – Пермь, 2007. 43 с.
2. Henner V.K., Raikher Yu. L., Kharebov P.V. // Phys. Rev. B. – 2011. – Vol. 84. – P. 144412-7.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ПОСТРОЕНИЯ БОЛЬШИХ КОММУНИКАЦИОННЫХ СЕТЕЙ СО СВОЙСТВАМИ “МАЛОГО МИРА”

А.П. Демичев[1,2], В.А. Ильин[1,2], А.П. Крюков[1,2], С.П. Поляков[2]

[1]Национальный исследовательский центр «Курчатовский институт», Москва
[2]Научно-исследовательский институт ядерной физики им. Д.В. Скобельцына, Москва

Предлагается подход к разработке коммуникационных сетей суперкомпьютеров следующего поколения. Рассмотрен ряд как известных в литературе, так и оригинальных алгоритмов построения сложных сетей со свойствами «малого мира», а именно с медленным (логарифмическим) ростом среднего расстояния между узлами с ростом их числа. При этом сети, построенные на основе этих алгоритмов, имеют базовую структуру регулярной решетки с дополнительными перемычками между узлами, которые и обеспечивают свойства «малого мира». Предложена методика сравнения эффективности алгоритмов различных типов.

Одной из важнейших составляющих суперкомпьютеров является коммуникационная сеть, которая в первую очередь определяет возможность увеличения числа вычислительных узлов, что необходимо для достижения желаемой производительности. Таким образом, одной из ключевых задач, которую предстоит решить на пути к построению суперкомпьютеров следующего поколения, является разработка коммуникационных сетей с хорошими свойствами масштабируемости и возможностью эффективно обслуживать огромное число вычислительных узлов [1].

Тремя основными аспектами проектирования коммуникационных сетей, которые в наибольшей степени определяют их функциональные свойства, являются: топология сети, метод управления потоками и алгоритм маршрутизации. В данной работе обсуждается, в основном, топология коммуникационной сети (в том смысле, в котором термин используется в теории сетей). Два других аспекта очень важны, но выходят за рамки текущего обсуждения. Выбор подходящей топологии жизненно важен для проектирования сети, поскольку маршрутизация и механизмы управления потоком в большой степени основаны на ее свойствах.

В работе рассматриваются только прямые сети, в которых каждый узел является терминальным, действующим и как источник, и как приемник для сообщений. Непрямые сети (содержащие узлы-рутеры, которые используются только для маршрутизации) имеют свои достоинства для ограниченного числа узлов, но плохо масштабируются. В идеальном случае коммуникационная сеть должна была бы быть полностью соединена (полный граф), чтобы позволить одновременную непосредственную связь между всеми парами узлов, достигая оптимальной пропускной способности и задержки. Этот подход также может быть применен к системам с небольшим числом узлов и не масштабируется на большие сети из-за слишком быстрого роста числа необходимых связей между узлами. Пропускная способность сети должна масштабироваться с ростом числа вычислительных узлов, что обеспечивается правильной комбинацией хорошего выбора топологии и алгоритмов маршрутизации.

В данной работе рассматриваются обобщения регулярных решеток с топологией n-мерных торов. В литературе, посвященной сетям, для такой топологии часто используется термин «k-ary n-cube» [2]. Известно, что такие сети при большом числе узлов име-

ют много преимуществ по сравнению с другими архитектурами, например гиперкубами высоких размерностей (см., например, [3]). Важным аргументом в пользу использования регулярных решеток является тот факт, что на такую структуру коммуникационной сети естественным образом отображаются параллельные вычислительные задания, связанные с численным моделированием *n*-мерных объектов. В частности, коммуникационные сети со структурой трехмерной решетки оптимальны для моделирования трехмерных реальных объектов, а именно такого типа задачи, как предполагается, будут составлять значительную долю задач, решаемых на суперкомпьютерах следующих поколений, в частности суперкомпьютерах экзафлопсного уровня.

Однако при огромном числе узлов, характерном для компьютеров следующих поколений, архитектура регулярных решеток с топологией *n*-мерных торов имеет и существенные недостатки. В частности, решетки невысокой размерности имеют весьма большую среднюю длину пути между узлами, а решетки высокой размерности, сравнимой с логарифмом числа узлов, трудно реализовать технически из-за большой длины физических коммуникационных каналов. С другой стороны, известно, что наилучшими структурами вычислительных систем по различным критериям функционирования, например производительности и надежности, при одинаковом числе вычислительных узлов и каналов связи, являются структуры с минимальным средним расстоянием между узлами (см., например, [4]). Поэтому обычные сети с простой структурой регулярных решеток окажутся недостаточно эффективными для решения более общего типа задач, не связанных с триангуляцией трехмерных объектов.

В связи с этим представляется перспективным использовать для построения коммуникативных сетей для компьютеров следующих поколений сети со свойствами «малого мира» [5], одним из важнейших свойств которых является малое среднее расстояние между узлами и малый диаметр сети. Более точное выражение свойства малой средней длины заключается в следующем: для регулярной *D*-мерной решетки среднее расстояние между узлами *d* растет как степень числа узлов: $d \sim N^{1/D}$, а для сети со свойствами «малого мира» существенно медленнее: $d \sim \ln N$.

В классическом варианте [5] сложные сети со свойствами «малого мира» получаются на промежуточной стадии процесса стохастической трансформации регулярных решеток в полностью случайные графы Эрдеша-Реньи (см., например, обзор [6] и ссылки в нем). При этом структура регулярной решетки нарушается, что, как отмечалось выше, нежелательно для коммуникационных сетей суперкомпьютеров. Поэтому в данной работе предлагается использовать ряд модификаций способа построения сетей с малой средней длиной пути между узлами, при которых сохраняется базовая решеточная структура, но к ней определенным образом добавляются дополнительные связи, называемые перемычками, которые и обеспечивают свойства «малого мира».

Длина пути (расстояние) между узлами понимается в сетевом смысле: как минимальное число ребер, по которым надо пройти, чтобы попасть из одного узла в другой. Соответственно среднее расстояние между несовпадающими узлами определяется как среднее по всем парам узлов данной сети. Однако для больших сетей определенная таким образом длина пути между узлами может оказаться неадекватной характеристикой, поскольку для нахождения кратчайших маршрутов необходимо знать глобальную структуру сети. Соответственно маршрутизация сообщений, использующая кратчайшие пути, может оказаться слишком сложной и неэффективной, так как связана с хранением и обработкой большого объема информации. Поэтому особую важность приобретают алгоритмы маршрутизации, основанные на локальной навигации [7]. В краткой формулировке задача навигации в сетях ставится следующим образом: узел «знает» географическое положение (другими словами, положение в базовой решетке) всех узлов, а также своих ближайших сетевых соседей с учетом дополнительных перемычек;

необходимо доставить сообщение в узел назначения по возможно кратчайшему пути, не используя глобальной информации обо всех перемычках в сети. В простейшем варианте эту задачу решает так называемый жадный алгоритм (greedy algorithm): текущий узел пересылает сообщение тому из своих соседей, который географически (то есть в смысле координат на решетке) ближе всего к цели (узлу назначения). Таким образом, в данной работе наряду с глобальным средним расстоянием между узлами рассматривается и средняя навигационная длина пути между узлами сети как важная характеристика, определяющая коммуникационные свойства сети. При этом для некоторых рассмотренных сетей оказалось необходимым рассмотреть обобщение обычного жадного алгоритма, при котором учитывается положение не только ближайших соседей текущего узла, но и соседей соседей. Другими словами, сообщение на каждом шаге пересылается в тот соседний узел, один из соседей которого ближе всего к узлу назначения в смысле решеточной метрики. Хотя при таком алгоритме объем вычислений на каждом шаге маршрутизации несколько увеличивается, но алгоритм остается локальным (не вычисляется весь путь до адресата, и объем не зависит от размеров системы). Поэтому этот алгоритм является хорошо масштабируемым и приемлем для сверхбольших коммуникационных сетей.

Основной целью работы является разработка оптимального алгоритма (или алгоритмов) построения сети с большим числом узлов, но малой глобальной и/или навигационной средней длиной пути между узлами. Общая идея состоит в добавлении к решеточной основе дополнительных перемычек по специальному алгоритму (или алгоритмам), чтобы оптимизировать соотношение «цены» и «качества» для получаемой таким образом сети. В качестве «цены»выступает удельная длина дополнительных перемычек C/L (общая длина перемычек C в единицах базовой решетки, деленная на число узлов сети L), а «качество» – это глобальная d или навигационная l средняя длина пути между узлами. Очевидно, что эти две величины взаимосвязаны: увеличивая цену C/L можно улучшить качество (уменьшить d или l), и, наоборот, улучшение качества зачастую связано с увеличением цены. Существует ряд подходов и методов для решения задач такой многокритериальной оптимизации (см., например, [8]). В работе используется один из простейших и наглядных методов, а именно метод взвешенных сумм (в более общем контексте такой подход называется скаляризацией многокритериальной оптимизации). Для этого определены и оптимизированы (а именно, минимизированы) следующие скалярные целевые функции $G_w = w\,d + (1–w)\,C/L$ и $G_w^{nav} = w\,l + (1–w)\,C/L$. Минимизация этих целевых функций означает, что подобраны оптимальные значения параметров алгоритмов с точки зрения качества (малой длины пути между узлами) и цены (малой длины перемычек). При этом параметр $0 \le w \le 1$ характеризует относительную значимость каждого из критериев (качество и цена).

Рассмотрен ряд как известных в литературе [9–14], так и предложенных авторами алгоритмов построения сложных сетей со свойствами «малого мира». Как уже упоминалось, оригинальный алгоритм получения сложной сети со свойствами «малого мира» [5] является стохастическим: на каждом шаге алгоритма ребра графа меняют свое положение с некоторой вероятностью. В результате многократного применения такого алгоритма возникает ансамбль графов с некоторым распределением их характеристик, в частности с некоторым распределением средней длины пути между узлами экземпляра графа. Для многих реальных сетей стохастический процесс их формирования оказывается внутренне присущим (так, это справедливо для сети Интернет; другие примеры см., например, в [6]). Однако проектирование коммуникационной сети суперкомпьютера находится под контролем разработчика, и поэтому стохастичность не является внутренне присущим элементом этого процесса. Поэтому важным вопросом является следующий: существует ли такой регулярный (детерминистский) алгоритм модификации

решетки с помощью перемычек, превращающей ее в сеть «малого мира», чтобы соотношение «цена»–«качество» полученной сети было лучше, чем при использовании стохастических алгоритмов. Исследованию этого вопроса и посвящена, в основном, эта работа. Показано, что наиболее эффективной для построения сверхбольших коммуникационных сетей структурой обладают сети, построенные на основе предложенного в данной работе детерминистского алгоритма.

Работа частично финансируется РФФИ, грант 12-07-00408-а.

Литература

1. Shainer G., Sparks B., Graham R. Toward Exascale computing, HPC Advisory Council – [http://www.hpcadvisorycouncil.com/pdf/Toward_Exascale_computing.pdf].
2. Report on Institute for Advanced Architectures and Algorithms Interconnection Networks Workshop 2008, Future Technologies Group Technical Report Series, Oak Ridge, Tennessee USA – [http://www.csm.ornl.gov/workshops/IAA-IC-Workshop-08].
3. Dally W.J., Towles B.P. Principles and Practices of Interconnection Networks. – Amsterdam: Elsevier Science, 2003. 550 p.
4. Dally W.J. Performance Analysis of k-ary n-cube Interconnection Networks // IEEE Transactions on Computers. 39 (1990) 775.
5. Kleinrock L. Communication Nets: Stochastic Message Flow and Design. New York: McGraw-Hill, 1964. 220 p.
6. Watts D.J., Strogatz D.H. Collective dynamics of small-world networks // Nature. 393 (1998) 440.
7. Albert R., Barabasi A.-L. Statistical mechanics of complex networks // Rev. Mod. Phys. 74 (2002) 47.
8. Kleinberg J.M. Navigation in the small world // Nature. 406 (2000) 845.
9. Steuer, R.E. Multiple Criteria Optimization: Theory, Computations, and Application.- New York: John Wiley and Sons, 1986. 330 p.
10. Zou Zhi-Yun et al. Regular Small-World Network // Chin. Phys. Lett. 26 (2009) 110502.
11. Boettcher S., Goncalves B., Azaret J. Geometry and Dynamics for Hierarchical Regular Networks // Journal of Physics A 41 (2008) 335003.
12. Boettcher S., Goncalves B., Guclu H., Hierarchical Regular Small-World Networks // J. Phys. A. 41 (2008) 252001.
13. Comellas F., Ozona J., Peters J. G. Deterministic small-world communication networks // Information Processing Letters. 76 (2000) 83.
14. Comellas F., Mitjana M., Peters J.G. Broadcasting in Small-World Communication Networks // In: Proc. 9th Int. Coll. on Structural Information and Communication Complexity (2002), eds. C. Kaklamanis and L. Kirousis. P. 73–85.
15. Moukarzel C.F., de Menezes M.A. Shortest paths on systems with power-law distributed long-range connections // Phys. Rev. E. 65 (2002) 056709.
16. Sen P., Chakrabarti B. Small-world phenomena and the statistics of linear polymer // J. Phys. A. 34 (2001) 7749.
17. Barthelemy M. Spatial Networks // Phys. Reports. 499 (2011) 1.

ИСПОЛЬЗОВАНИЕ МОДУЛЯРНОЙ АРИФМЕТИКИ ДЛЯ УСКОРЕНИЯ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ НАД ЧИСЛАМИ БОЛЬШОЙ РАЗРЯДНОСТИ

М.А. Дерябин, А.А. Зайцев

Северо-Кавказский федеральный университет, Ставрополь

Описывается один из методов ускорения вычислений над числами большой разрядности, основанный на применении системы остаточных классов. Рассмотрены современные методы и алгоритмы модулярной арифметики. Приведен пример использования модулярной арифметики для ускорения работы алгоритма RSA. Представлены результаты тестирования разработанных на основе исследования программ.

Введение

Совершенствование вычислительной техники влечет за собой необходимость работы с большими объемами данных, с числами большой разрядности. Это приводит к замедлению операций, лежащих в основе многих алгоритмов. Возникает необходимость поиска путей ускорения преобразований. Часто для этого используется декомпозиция и распараллеливание последовательных алгоритмов выполнения операций. Но в некоторых случаях стандартные способы распараллеливания вычислений малоэффективны. Одним из перспективных путей решения задачи сокращения времени обработки данных является применение различных форм параллельной обработки данных на основе числовых систем с параллельной структурой. Современным подходом к созданию отказоустойчивых высокопроизводительных средств обработки данных является использование системы остаточных классов (СОК).

1. Основы модулярной арифметики. Модульное возведение в степень

Система остаточных классов (Residue number system) является непозиционной системой представления чисел [1]. Пусть задана некоторая система взаимно-простых модулей $\{p_1, p_2, \ldots, p_n\}$. Число A в СОК по данным модулям представляется в виде кортежа чисел $(a_1, a_2, \ldots, a_n)$, где $a_i = A(\bmod p_i)$ для $i = 1,2,\ldots,n$. В соответствии с Китайской теоремой об остатках, такое представление числа A является единственным, если $0 \le A < P = p_1 p_2 \ldots p_n$, где P называется диапазоном СОК. При этом операции $C = A + B(\bmod P)$ и $D = A \cdot B(\bmod P)$ для чисел $A = (a_1, a_2, \ldots, a_n)$ и $B = (b_1, b_2, \ldots, b_n)$, представленных в СОК, определяются следующим образом:

$$c_i = a_i + b_i(\bmod p_i),\ d_i = a_i \cdot b_i(\bmod p_i),\ i = 1,2,...,n.$$

Описанное представление эффективно использовать при выполнении операций умножения и сложения, так как числа a_i и b_i имеют гораздо меньшее число разрядов, чем исходные числа A и B. При этом, так как в таком представлении нет межразрядных связей, эти операции можно выполнять параллельно по разным модулям. Таким образом, модулярная арифметика позволяет проводить декомпозицию системы большого динамического диапазона на ряд параллельных независимых каналов меньшей разрядности. Использование такого подхода увеличивает эффективность вычислений.

Одной из наиболее часто используемых операций при работе с большими целыми числами является модульное возведение в степень. Алгоритмы, реализующие данную операцию, используются в различных областях. Модульное возведение в степень представляет собой последовательное повторение операций умножения. Так как операция умножения выполняется по каждому модулю параллельно, то возведение в степень по составному модулю P можно представить как n независимых возведений в степень по модулям $p_1, p_2, \ldots, p_n$.

Рассмотрим один из алгоритмов быстрого модульного возведения в степень. Пусть требуется вычислить $z = a^s (\mathrm{mod}\, p)$. Число s может быть записано как

$$s = \sum_{i=0}^{h-1} s_i 2^i \qquad (1)$$

Положим старший разряд $s_{h-1} = 1$ по определению. Числа s_i могут принимать значения 0 и 1. Число h является длиной бинарного представления числа s. Из выражения (1) следует следующая формула для возведения в степень:

$$z = a^{\left(\sum_{i=0}^{h-1} s_i 2^i\right)} (\mathrm{mod}\, p) = \prod_{i=0}^{h-1} \left(a^{2^i}\right)^{s_i} (\mathrm{mod}\, p). \qquad (2)$$

Формула (2) лежит в основе алгоритма быстрого модульного возведения в степень [2].

Еще один способ ускорения операции возведения в степень в СОК заключается в сокращении показателя степени. Такая возможность следует из Малой теоремы Ферма. При этом требуется, чтобы все модули системы являлись простыми числами. В Малой теореме Ферма говорится, что для любого числа a, не делящегося на простое число, верно равенство $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$.

Интересным для нас является следствие из данной теоремы: если целое число a не делится на p и если $n \equiv m (\mathrm{mod}(p-1))$, то $a^n \equiv a^m (\mathrm{mod}\, p)$. В следствии говорится, что при работе по простому модулю p показатели могут быть снижены до $\mathrm{mod}(p-1)$.

Принимая во внимание вышеизложенные факты, выражение $Z = A^B (\mathrm{mod}\, P)$, где $P = p_1 p_2 \cdot \ldots \cdot p_n$ и p_i – простые числа $(i = 1, 2, \ldots, n)$, можно заменить вычислениями в СОК по модулям $p_1, p_2, \ldots, p_n$: $Z = (a_1^{B_1} (\mathrm{mod}\, p_1), a_2^{B_2} (\mathrm{mod}\, p_2), \ldots, a_n^{B_n} (\mathrm{mod}\, p_n))$, где $B_i = B(\mathrm{mod}(p_i - 1))$.

2. Перевод чисел из СОК в позиционное представление и обратно

Особенностью адаптации алгоритмов для СОК является необходимость перевода данных из позиционной системы в СОК и обратно. Прямые преобразования из позиционной системы и обратно в случае обработки больших целых чисел являются непараллельными, что увеличивает сложность реализации преобразуемого алгоритма. В [3] предлагается эффективный метод преобразования двоичного числа в СОК на основе разбиения исходного двоичного числа на отдельные форматы, для которых отводится заранее известное количество двоичных разрядов. Любое двоичное число может быть записано в виде

$$X = \sum_{j=0}^{M} \left(\sum_{i=0}^{B-1} x_{jB+i} 2^i \right) 2^{jB},$$

где B – количество разрядов выбранного формата; M – степень формата; x_i – коэффициент 0 или 1; $j = 0, B, 2B, \ldots, MB$ – позиция формата; i – позиция разряда в формате.

Основываясь на этом выражении, можно записать формулу для вычисления остатка по модулю p:

$$|X|_p = \left|\sum_{j=0}^{M}\left(\sum_{i=0}^{B-1} x_{jB+i}2^i\right)2^{jB}\right|_p = \left|\sum_{j=0}^{M}\left(\left|\sum_{i=0}^{B-1} x_{jB+i}2^i\right|_p \cdot \left|2^{jB}\right|_p\right)\right|_p .$$

При выполнении операций следует учитывать, что $\omega_i = \left|2^{jB}\right|_p$ есть заранее вычисленные константы. Остаток от деления для каждого формата можно находить независимо относительно остальных форматов. После чего вычисленные остатки по каждому из форматов складываются по модулю p. При таком подходе к нахождению остатков операции проводятся над числами гораздо меньшей разрядности.

Обратное преобразование числа из модулярного представления в двоичную форму базируется на классической теореме из теории чисел, которая называется Китайской теоремой об остатках (КТО). На основании известного представления чисел в СОК $(\alpha_1, \alpha_2, ..., \alpha_n)$ КТО делает возможным определение числа в позиционную систему счисления (ПСС) $|x|_p$, если наибольший общий делитель любой пары модулей равен 1.

Китайская теорема об остатках имеет вид

$$|x|_p = \left|\sum_{i=1}^{n} P_i \left|\frac{\alpha_i}{p_i}\right|_{p_i}\right|_P , \qquad (3)$$

где $P_i = \dfrac{P}{p_i}$, $P = \prod_{i=1}^{n} p_i$ для $(p_i, p_j) = 1$, $i \neq j$. Такая форма КТО предполагает лишь последовательные вычисления по модулю P, который является достаточно большим. Снизить разрядность модуля можно с помощью применения позиционной системы со смешанными основаниями. Под обобщенной позиционной системой счисления (ОПСС) понимается такая система, в которой целое число N представляется в виде

$$N = a_{n-1}\pi_{n-1}\pi_{n-2}....\pi_2\pi_1 + a_{n-2}\pi_{n-2}\pi_{n-3}...\pi_2\pi_1 + ... + a_2\pi_2\pi_1 + a_1\pi_1 + a_0 ,$$

где a_j – цифры $0,1,...\pi_{j-1}$ $(j = 1,2,...n)$. Для перехода от вычислений по модулю P к вычислению по модулям p_i предлагается метод восстановления чисел на основе совместного использования КТО и обобщенной позиционной системы счисления (ОПСС) [3].

Пусть задана система оснований $p_1, p_2, ..., p_n$ с диапазоном $P = p_1 p_2 ... p_n$ и ортогональными базисами $B_1, B_2, ..., B_n$, которые определяются как

$$B_i = \frac{m_i P}{p_i} \equiv 1 (\mathrm{mod}\, p_i), \;\; i = \overline{1,n},$$

где m_i – веса ортогональных базисов. Представим ортогональные базисы B_i в ОПСС, тогда

$$B_i = b_{i1} + b_{i2} p_1 + b_{i2} p_1 p_2 + ... + b_{in} p_1 p_2 ... p_n , \qquad (4)$$

где b_{ij} – коэффициенты ОПСС, $i,\ j = 1,2,...,n.$ Из (4) и (3) следует

$$X_{ОПСС} = a_1(b_{11}, b_{12}, ..., b_{1n}) + a_2(0, b_{22}, ..., b_{2n}) + ... + a_n(0,0,...,b_{nn}).$$

Так как $B_i \bmod p_i = 0,\ \ \forall j > i,$ то перед первым значащим разрядом будет $i-1$ нулей. Для удобства вычислений базисы можно представить в виде матрицы. Тогда $X_{ОПСС}$ можно вычислить следующим образом

$\tau_i = \sum_{j=1}^{n} \alpha_i b_{ij}$ для $i = 1,2,...,n$, $a_1 = \tau_1 (\bmod p_1)$, $a_i = \left(\tau_1 + \left[\frac{\tau_{i-1}}{p_{i-1}} \right] \right) (\bmod p_i)$ для $i = 2,3,...,n$, где a_i – коэффициенты ОПСС числа x; α_i – вычеты числа x по $\bmod p_i$; b_{ij} – ортогональные базисы, представленные в ОПСС. Получение позиционного представления числа осуществляется по формуле

$$X = a_n p_1 p_2 \cdot ... \cdot p_{n-1} + a_{n-1} p_1 p_2 \cdot ... \cdot p_{n-2} + ... + a_2 p_1 + a_1 = a_1 + \sum_{k=1}^{n-1} q_k a_{k+1},$$

где $q_k = \prod_{i=1}^{k} p_i$ для $k = 1,2,...,n-1$ являются константами.

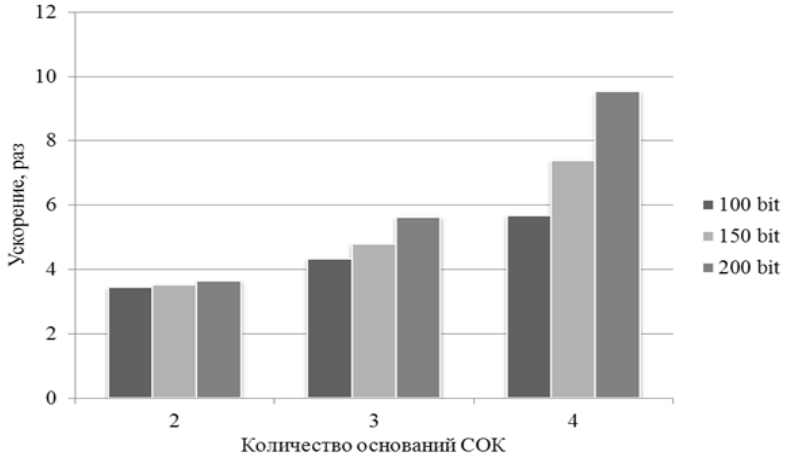


Рис. 1. Среднее ускорение тестовых программ, использующих СОК, относительно последовательных программ

На основе рассмотренных выше алгоритмов было разработано две программы. Первая реализует рассмотренный выше алгоритм модульного возведения в степень над числами различной разрядности. Вторая реализует тот же алгоритм, но с применением СОК. На рис. 1 приведена сравнительная диаграмма ускорения, получаемого при использовании СОК на основе тестовых данных разработанных программ для системы оснований, состоящей из двух, трех и четырех модулей по 100, 150 и 200 бит каждый. Из диаграммы видно, что при увеличении числа задействованных оснований и их разрядности ускорение резко возрастает. Причем оно превосходит количество используемых параллельных потоков.

3. Ускорение алгоритма расшифрования в схеме RSA

Модульные операции с большими целыми числами требуются во многих алгоритмах, классическим примером является схема шифрования RSA. Основной отличительной чертой RSA является тот факт, что модуль, используемый в схеме шифрования RSA, представляет собой произведение двух простых чисел. Это позволяет использовать двухмодульную СОК для ускорения операций с закрытым ключом.

Для простых P и Q любое сообщение $M < N = PQ$ единственным образом представляется парой $[M_P, M_Q]$, где $M_P = M (\bmod P)$ и $M_Q = M (\bmod Q)$. Таким образом можно получить M по вычислениям с M_P, M_Q и с их последующей «сборкой», а не обычным возведением $M = C^D (\bmod N)$. С помощью следствия Малой теоремы Ферма размер показателя может быть уменьшен. Со значениями $C_P = C (\bmod P)$ и

$C_Q = C(\bmod Q)$, а также $D_P = D(\bmod(P-1))$ и $D_Q = D(\bmod(Q-1))$ мы получаем следующие выражения для M_P и M_Q: $M_P = C_P^{D_P}(\bmod P)$ и $M_Q = C_Q^{D_Q}(\bmod Q)$.

Если предположить, что показатели $D_P = D(\bmod(P-1))$ и $D_Q = D(\bmod(Q-1))$ были предварительно вычислены, RSA-расшифрование, основанное на использовании СОК, может иметь место в соответствии со следующими шагами:

Вычислить $C_P = C(\bmod P)$ и $C_Q = C(\bmod Q)$.

Возвести в степень $M_P = C_P^{D_P}(\bmod P)$ и $M_Q = C_Q^{D_Q}(\bmod Q)$.

Перевести полученные результаты из СОК в ПСС.

Два возведения в степень (шаг 2) можно вычислить независимо друг от друга и параллельно. На рисунке 2 представлены результаты тестов предложенного алгоритма и его последовательной версии. При этом достигается ускорение, в среднем равное четырем.

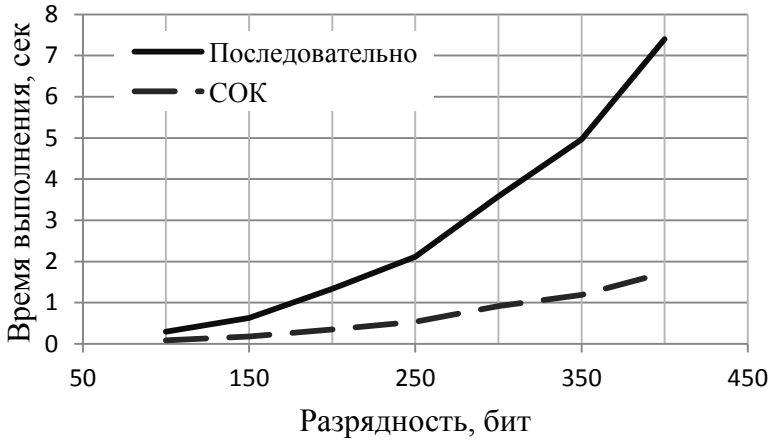


Рис. 2. Скорость выполнения обычного алгоритма RSA и RSA с применением СОК

Вывод

Современные достижения модулярной арифметики делают ее крайне эффективным средством для ускорения алгоритмов, использующих целые числа большой разрядности. При увеличении числа модулей ускорение выполнения операций возрастает, что ставит этот способ представления чисел в ряд наиболее перспективных методов в данной области.

Литература

1. Omondi A., Premkumar B. Residue number systems. Theory and Implementation. – London: Imperial College Press, 2007.
2. Schneier Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. – New York: John Wiley & Sons, 1995. – 662 p.
3. Червяков Н.И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка и применение. №10, 2006. С. 24–36.

ИНТЕГРАЦИЯ СЕРВИСА ИНТЕРАКТИВНОЙ ВИЗУАЛИЗАЦИИ ДЛЯ РАСПРЕДЕЛЕННЫХ ВИРТУАЛЬНЫХ ИСПЫТАТЕЛЬНЫХ СТЕНДОВ С СИСТЕМОЙ CAEBEANS

Д.А. Диков, Г.И. Радченко

Южно-Уральский госуниверситет, Челябинск

Для решения задач инженерного моделирования и анализа целесообразно использование удаленных высокопроизводительных вычислительных систем. Однако объем данных, полученных в результате моделирования, может достигать сотен гигабайт, что порождает проблему передачи решений конечному пользователю. Для исключения копирования результатов на ПК нами предлагается механизм удаленной интерактивной визуализации решений задач инженерного моделирования. Решение имеет формат «облачного» веб-сервиса, интегрированного с системой CAEBeans и поддерживающего исполнение множества виртуальных машин, выполняющих визуализацию решений CAE-задач. Использование технологии noVNC позволяет обеспечить доступ к сессиям визуализации посредством веб-браузера.

Введение

На сегодняшний день наблюдается тенденция к объединению ресурсов нескольких высокопроизводительных суперкомпьютерных систем в вычислительные комплексы, в рамках которых осуществляется решение задач инженерного моделирования и анализа. Результатами вычислений являются файлы решений, объем которых может достигать нескольких сотен гигабайт. Визуализация такого объема данных на пользовательском ПК затруднена, а в ряде случаев невозможна в силу недостаточности аппаратных ресурсов. Решением данной проблемы является перемещение процесса визуализации с ПК на высокопроизводительные удаленные вычислительные фермы, располагающие вместительным и надежным хранилищем данных.

Сервис Oxalya VisuPortal [9] предоставляет пользователю удаленный доступ к ресурсам облачной системы визуализации. Для передачи результатов визуализации клиенту используется технология VNC. Веб-портал, посредством которого осуществляется доступ к сервису, использует для отображения удаленного рабочего стола Java-апплет, выполняющий функции VNC-клиента. Система удаленной визуализации [3] использует сервис-посредник для взаимодействия с модулем визуализации, представленным приложением, выполняющим построение графической сцены. Изображение, передаваемое клиенту, захватывается из окна приложения. В качестве клиента используется веб-браузер на компьютере пользователя.

В рамках систем интерактивной визуализации для снижения объема передаваемых клиентской стороне данных рядом исследователей применяется технология VNC (Virtual Network Computing) [1, 2, 9]. Подход, основывающийся на применении noVNC-решений [6], использующих преимущества технологии HTML5 для организации тонкого клиента на основе веб-браузера, не требует установки VNC-клиента на пользовательское устройство.

Предлагаемый подход может быть реализован на основе концепции «облачных вычислений» [4] в формате «облачного» веб-сервиса, предоставляющего пользователю механизм взаимодействия с распределенными виртуальными испытательными стендами [8], реализуемыми на основе системы CAEBeans. На сегодняшний день целый ряд

широко используемых CAE-систем, таких как ANSYS, DEFORM, FlowVision, были внедрены в систему CAEBeans в виде грид-сервисов [7]. В основе технологии CAEBeans лежит обеспечение сервис-ориентированного предоставления программных ресурсов базовых компонентов CAE-систем и формирование иерархий проблемно-ориентированных оболочек, инкапсулирующих процедуру постановки и решения определенного класса задач.

В данной работе нами предлагается решение в формате облачного сервиса интерактивной визуализации решений CAE-задач. Данный сервис интегрирован с системой CAEBeans и имеет тонкий клиент, доступный для отображения в любом веб-браузере, поддерживающем стандарт HTML5.

Архитектура сервиса интерактивной визуализации

Цель сервиса интерактивной визуализации – обеспечить пользователю доступ к результатам визуализации решения CAE-задачи в рамках распределенного виртуального испытательного стенда.

В задачи сервиса входит:

1. выполнение визуализации решения задачи инженерного моделирования;
2. передача результатов визуализации пользователю;
3. предоставление пользователю возможности интерактивного взаимодействия с результатами визуализации.

В процессе проектирования архитектуры сервиса интерактивной визуализации была выделена сущность «сессия визуализации», инкапсулирующая процесс визуализации решения задачи инженерного моделирования и обеспечивающая предоставление результатов визуализации.

Сервис интерактивной визуализации поддерживает одновременное исполнение множества сессий визуализации, каждая из них представлена виртуальной машиной, в рамках которой установлен CAE-пакет, осуществляющий процесс визуализации загруженного решения CAE-задачи. Для запуска сессии визуализации сервису интерактивной визуализации требуется загрузить файлы решения CAE-задачи из хранилища данных, в роли которого выступает CAEBeans Server (рис. 1).

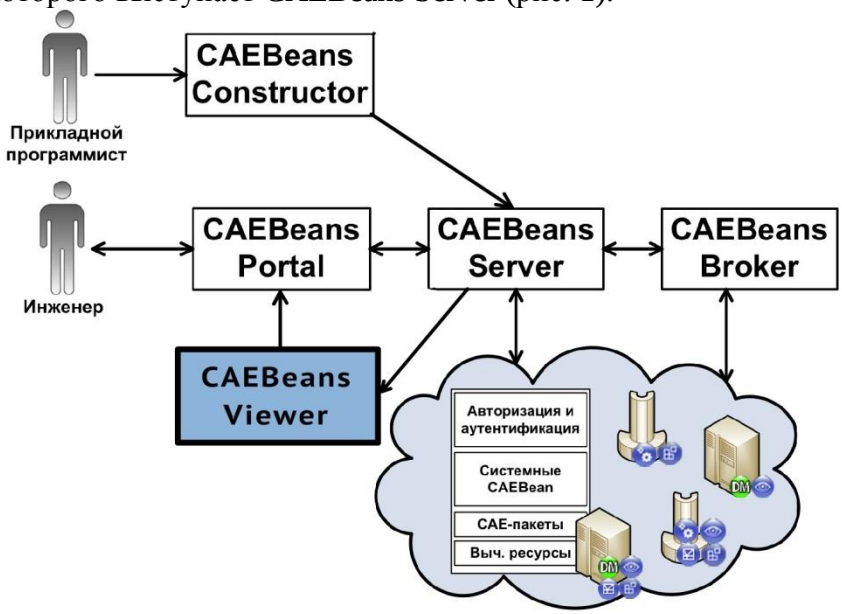


Рис. 1. Сервис CAEBeans Viewer в составе системы CAEBeans

В качестве клиента сервиса интерактивной визуализации выступает веб-браузер, установленный на пользовательском устройстве (рис. 2). Клиент использует для соеди-

нения с сессией визуализации протокол WebSocket, предоставляющий возможность установления полнодуплексной связи поверх TCP-соединения [3]. Это позволяет избежать необходимости установки VNC-клиента на пользовательское устройство.

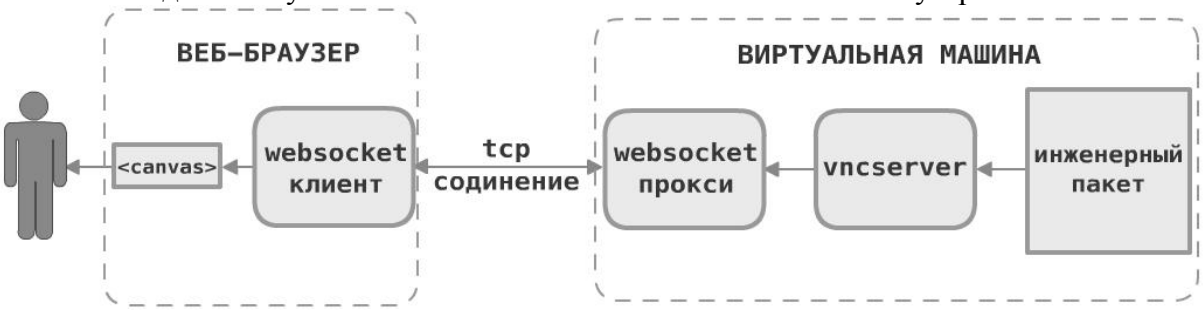


Рис. 2. Архитектура noVNC-соединения

Подготовка сессии визуализации инициируется по запросу от клиента и состоит из следующих этапов:

1. создание виртуальной машины и установка требуемой операционной системы;
2. загрузка файлов решения CAE-задачи;
3. установка инженерного пакета;
4. запуск визуализации решения;
5. подготовка noVNC-подключения;
6. добавление информации о созданной и запущенной сессии визуализации в базу данных.

CAE-пакет, устанавливаемый на виртуальной машине, подбирается в соответствии с форматом файлов решения задачи инженерного моделирования, построенных в рамках системы CAEBeans и хранящихся на сервере CAEBeans Server. Архитектура сервиса удаленной визуализации не накладывает ограничений на выбор инженерного пакета и предоставляет возможность развертывания любого требуемого ПО в рамках виртуальных машин на сервере.

Реализация сервиса интерактивной визуализации

Сервис интерактивной визуализации – это множество виртуальных машин под управлением гипервизора Xen, одна из которых содержит веб-службу, посредством которой предоставляется функционал сервиса, а остальные поддерживают сессии визуализации решений CAE-задач. Веб-служба, размещенная на сервере, представляет собой приложение, реализованное на основе инфраструктуры Ruby On Rails.

Интерфейс сервиса интерактивной визуализации обеспечивает возможность запрашивать и передавать данные о пользователе или сессии визуализации посредством POST- и GET-запросов. Для начала работы с сервисом клиенту необходимо направить GET-запрос, содержащий данные для аутентификации пользователя. При условии успешного прохождения процесса аутентификации, пользователь получает доступ к основному функционалу, предоставляемому сервисом.

Когда пользователь впервые запрашивает интерактивную визуализацию, в базе данных создается запись о новой, неактивной сессии визуализации. Сервер выполняет подготовку и настройку виртуальной машины, после чего состояние сессии визуализации меняется на «активное», что позволяет пользователю осуществить подключение и просматривать результаты визуализации. При поступлении запроса на подключение к сессии визуализации веб-служба передает клиенту хранимую в базе данных информацию для подключения к необходимой виртуальной машине. Используя полученные данные, клиент устанавливает подключение. Принимаемое клиентом сессии визуализа-

ции изображение преобразовывается в формат, приемлемый для отображения на веб-странице при помощи Canvas 2D API.

Для подключения клиент выполняет асинхронный запрос, содержащий идентификатор требуемой сессии визуализации. Ожидаемый ответ сервера содержит пароль для VNC-соединения и IP-адрес виртуальной машины. Эти данные используются клиентом для вызова функции connectVNC, устанавливающей noVNC-подключение к сессии визуализации.

Тестирование сервиса интерактивной визуализации

Для испытаний предложенного подхода сервис интерактивной визуализации был развернут на ресурсах суперкомпьютерного центра ЮУрГУ(НИУ). Для тестирования был использован веб-браузер Google Chrome (версия 19.0). В рамках подготовки демонстрационной сессии визуализации была создана виртуальная машина под управлением ОС Fedora 16. Для визуализации был использован файл геометрии в формате .k, построенный в инженерном пакете ANSYS, а также пре- и постпроцессор LS-PrePost 3.2. Внешний вид запущенного клиента сервиса CAEBeans Viewer, отображающего решение CAE-задачи, приведен на рис. 3.

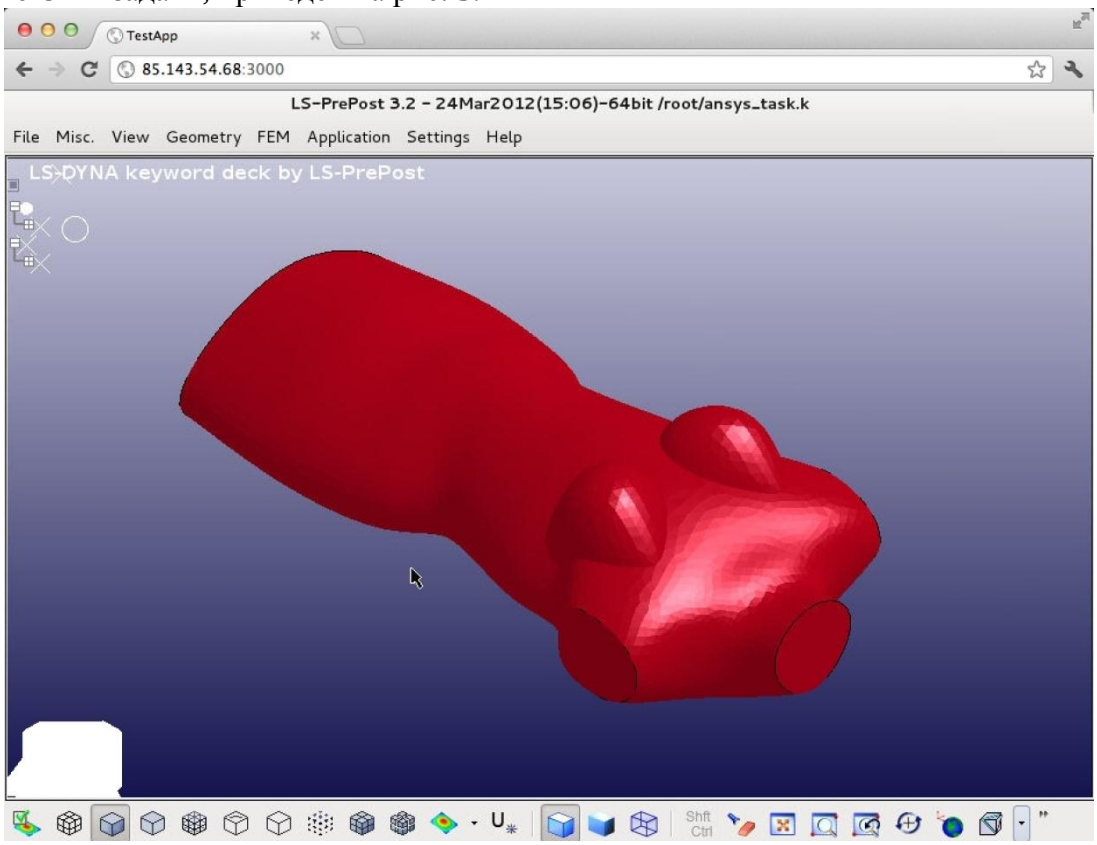


Рис. 3. Процесс работы сервиса

В результате подключения к сессии визуализации клиентом было отображено изображение удаленного рабочего стола – запущенная программа LS-PrePost и отображенная в окне «viewport» геометрия.

Заключение

В данной работе предложен новый подход к проблеме интерактивной визуализации решений задач инженерного моделирования, предполагающий использование noVNC-решений для организации тонкого клиента. Была разработана архитектура сервиса интерактивной визуализации CAEBeans Viewer, и реализован прототип сервиса, позволя-

ющий поддерживать пользовательские сессии визуализации решений CAE-задач. Дальнейшее развитие представленной работы предполагает разработку полнофункциональной версии сервиса с последующим созданием промышленного прототипа сервиса и внедрением его в опытную эксплуатацию.

Проект выполнен при поддержке Совета по грантам Президента Российской Федерации (проект № МК-1987.2011.9), Российского фонда фундаментальных исследований (проект № 11-07-00478) и Фонда содействия развитию малых форм предприятий в научно-технической сфере (проект № 10980р/16895).

Литература

1. Margalef F.P., Vall M. R., Garcia A.I., et al. Usage of thin clients on STB for secure interactive applications – [http://nem-summit.eu/wp-content/ plugins/alcyonis-event-agenda//files/Usage_of_thin_clients_on_STB_for_secure_inter-active_applications.pdf].
2. Shizuki B., Nakasu M., Tanaka J. VNC-based Access to Remote Computers from Cellular Phones // CSN ’02: Proceedings of the IASTED International Conference on Communication Systems and Networks, 2002. P. 74-79.
3. The Web Sockets API – [http://www.w3.org/TR/2009/WD-websockets-20091222].
4. Vaquero L.M. et al. A break in the clouds: towards a cloud definition // ACM SIGCOMM Computer Communication Review, 2009. Vol. 39. P. 50-55.
5. Бахтерев М.О., Васев П.А., Казанцев А.Ю., Манаков Д.В. Система удаленной визуализации для инженерных и суперкомпьютерных вычислений // Вестник ЮУрГУ. 2009. №17(150). С. 4–11.
6. Диков Д.А. Подход к проблеме удаленной визуализации сложных 3D-моделей на базе noVNC-решений // Параллельные вычислительные технологии (ПаВТ’2012): Труды Международной научной конференции (Новосибирск, 26 – 30 марта 2012 г.). Челябинск: Издательский центр ЮУрГУ, 2012. С. 444–448.
7. Долганина Н.Ю., Сапожников С.Б., Маричева А.А. Моделирование ударных процессов в тканевых бронежилетах и теле человека на вычислительном кластере «СКИФ Урал» // Вычислительные методы и программирование: Новые вычислительные технологии. Т. 11. 2010. С. 117-126.
8. Радченко Г.И., Соколинский Л.Б. Технология построения виртуальных испытательных стендов в распределенных вычислительных средах // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. № 54. 2008.
9. Сайт проекта Oxalya – [http://www.oxalya.com].

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ В РЕШЕНИИ КРАЕВОЙ ЗАДАЧИ НА ПРИМЕРЕ ВХОДА АППАРАТА В АТМОСФЕРУ

В.В. Дикусар, Н.Н. Оленёв

Вычислительный центр им. А.А. Дородницына РАН, Москва

При решении задач оптимального управления применение принципа максимума редуцирует исходную постановку к краевой задаче для системы обыкновенных дифференциальных уравнений. При этом возникает необходимость численно определять матрицу Якоби. Предложена схема параллельного вычисления указанной матрицы на каждом шаге расчета траектории. Применение параллельной схемы иллюстрируется практической задачей о выборе угла атаки аппарата, тормозящего в атмосфере при полете на максимальную и минимальную дальность с учетом ограничений на величину полной перегрузки.

Введение

Применение принципа максимума сводит задачу оптимального управления к краевой задаче для системы обыкновенных дифференциальных уравнений [1]. Трудоемкая процедура численного расчета матрицы Якоби в краевой задаче естественно распараллеливается в силу независимости частных производных. Параллельный алгоритм применяется для решения задачи оптимального управления, возникающей при расчете входа аппарата в атмосферу.

1. Уравнения движения. Постановка задачи

Рассматривается задача о выборе управления подъемной силой спускаемого аппарата в атмосфере при полете на максимальную и минимальную дальность с учетом ограничений на величину полной перегрузки. Решение указанной задачи позволяет определить маневренные возможности аппарата [1].

Выражение для полной перегрузки имеет вид:

$$\varphi(x,u) = n = (c_x^2 + c_y^2)^{1/2} qS/G \le N, \quad q = \rho V^2/2, \tag{1}$$

где q – скоростной напор; ρ – плотность атмосферы; V – скорость; c_x – коэффициент сопротивления; c_y – коэффициент подъемной силы; S – характерная площадь аппарата, G – вес.

Из (1) видно, что $\varphi(x,u)$ содержит явно управление c_y и рассматриваемое ограничение принадлежит к классу смешанных.

Аэродинамические силы, действующие на аппарат, характеризуются полярой вида

$$c_x = a + bc_y^2, \tag{2}$$

где a – коэффициент сопротивления при нулевом угле атаки, b – параметр поляры.

На величину управляющей функции $c_y(t)$ наложены ограничения

$$c_1 \le c_y(t) \le c_2. \tag{3}$$

Уравнения плоского движения аппарата в атмосфере имеют вид

$$\dot{V} = -c_x q \frac{S}{m} - g\sin\theta,\ t \in [t_0, t_1],$$

$$\dot{\theta} = c_y q/mV + (V/R - \frac{g}{V})\cos\theta, \tag{4}$$

$$\dot{h} = V\sin\theta,\ \dot{L} = V\cos\theta.$$

Здесь g – ускорение силы тяжести; h – высота; L – дальность полета; θ – местный угол наклона траектории; t – время; m – масса аппарата; R – радиус планеты.

Будем считать атмосферу изотермической

$$\rho = \rho_o \exp(-\beta h), \tag{5}$$

где ρ_o – плотность атмосферы на поверхности планеты; β – показатель экспоненты в формуле для плотности.

Для системы (4) заданы начальные условия

$$V(t_0) = V_0,\quad \theta(t_0) = \theta_0,\quad h(t_0) = h_0,\quad L(t_0) = 0. \tag{6}$$

Граничные условия определяются конечной высотой полета

$$h(t_1) = h_1, \tag{7}$$

причем $V_1(t_1)$, $\theta(t_1)$, t_1 в конце полета не фиксированы.

Требуется найти траекторию, удовлетворяющую связям (1)-(7), на которой $L(t_1)$ достигает экстремального значения.

2. Принцип максимума (регулярный случай)

Исследование задачи в регулярном случае проводится с использованием классического принципа максимума Понтрягина Л.С. [1]. Траектория называется регулярной, если во всех состояниях системы, встречающихся на оптимальной траектории, управление эффективно. Для рассматриваемой задачи требование регулярности эквивалентно требованию $\partial n/\partial c_y \neq 0$ при $n = N$.

Обозначая сопряженные переменные к θ, h, V, L для системы (4) через p_1, p_2, p_3, p_4, запишем функцию Понтрягина

$$H = p_1\left[c_y \rho VS/2m + \left(\frac{V}{R} - g/V\right)\cos\theta\right] + p_2 V\sin\theta -$$

$$- p_3\left(\frac{c_x \rho V^2 S}{2m} + g\sin\theta\right) + p_4 V\cos\theta. \tag{8}$$

Так как система (4) автономна и на время спуска никаких ограничений не накладывается, то $H \equiv 0$ на всем интервале движения.

В соответствии с [1] система сопряженных уравнений должна иметь вид

$$p_i = -\partial H/dx_i + \lambda(t)\partial n/\partial x_i,\quad x = (\theta, h, V, L) = \{x_i\},\quad i = \overline{1,4}.$$

Выпишем теперь уравнения для p_i в явном виде

$$\dot{p}_1 = -\frac{\partial H}{\partial \theta}, \dot{p}_2 = -\frac{\partial H}{\partial h}, \dot{p}_3 = -\frac{\partial H}{\partial V}, \dot{p}_4 = 0, \tag{9}$$

где $\lambda(t)$ – множитель Лагранжа, причем $\lambda(t)(n - N) = 0$,

$$\lambda(t) = g(p_1/2 - bp_3 c_y V)\sqrt{c_x^2 + c_y^2}/V \cdot c_y(1 + 2bc_x). \tag{10}$$

Оптимальное управление определяется из условия минимума (максимума) функции Понтрягина (8) при $p_4(t_1) = -1$

$$H \to \min_{c_y} \text{ при } L(t_1) \to \max, H \to \max_{c_y} \text{ при } L(t_1) \to \min . \quad (11)$$

Из условий $\dot{p}_4 = 0$ (9) и $p_4(t_1) = -1$ следует, что $p_4(t) \equiv -1$ на всей оптимальной траектории. Поскольку $V(t_1)$ и $\theta(t_1)$ в конце полета не фиксированы, то

$$p_1(t_1) = 0, \quad p_3(t_1) = 0. \quad (12)$$

Если задать $p_1(t_0)$ и $p_3(t_0)$, то из условия $H \equiv 0$ можно определить $p_2(t_0)$, и число контролируемых в конце траектории функций $p_1(t_1)$ и $p_3(t_1)$ (12) совпадает с числом параметров, задаваемых в начальной точке.

Таким образом, поставленная задача сводится к двухточечной краевой задаче для системы обыкновенных дифференциальных уравнений.

Рассмотрим методические особенности решения задачи с учетом ограничения $n \le N$ (1). При движении в открытой области $n < N$ имеем согласно условию дополняющей нежесткости $\lambda(t)(n - N) = 0$: $\lambda(t) = 0$; при этом значение n соответствует значению c_y, определенному согласно принципу максимума (минимума)

$$c_y = c_1, \quad c_y = c_2, \quad c_y = c_y^*, \quad c_y^* = p_1/2bp_3V, \quad c_1 \le c_y^* \le c_2.$$

Значение c_y^* вычисляется из уравнения $\partial H/\partial c_y = 0$. Для полученных значений c_y вычисляем величины

$$H_1 = H(c_1), \quad H_2 = H(c_2), \quad H_3 = H(c_y^*), \quad H_0 = \max\{H_1, H_2, H_3\}. \quad (13)$$

Максимальное значение функции Понтрягина (13) определяет оптимальное управление

$$c_y^0 = \arg H_0. \quad (14)$$

В некоторый момент времени функция $n(t)$ станет больше N и тогда равенством $n(t) = N$ определится время выхода на ограничение. При движении по ограничению $\varphi(x,u) = N$ (1) управление $c_y(t)$ определяется из условия связи

$$c_y^2(t) = \{-(1+2ab) + [(1+2ab)^2 + 4(Nmg/sq - a^2)b^2]^{1/2}\}/2b^2 = r(t). \quad (15)$$

Отсюда следует: $c_y = \sqrt{r(t)}$, $c_y = -\sqrt{r(t)}$. Полученные выражения дают возможность вычислить

$$H_4 = H(\sqrt{r(t)}), \quad H_5 = H(-\sqrt{r(t)}), \quad H_n = \max\{H_4, H_5\}. \quad (16)$$

Соотношения (16) определяют необходимый знак для $c_y^0(t)$.

Из выражения для $c_y^2(t)$ (15) следует, что $|c_y(t)|$ является непрерывной функцией в силу непрерывности $q(t)$.

Момент схода с ограничения $n(t) = N$ устанавливается двумя теоремами.

Теорема 1. В задаче на $L(t_1) \to \min$ сход с ограничения $n(t) = N$ происходит в момент $c_y(t) = c_1$.

Теорема 2. В задаче на $L(t_1) \to \max$ сход с ограничения $n(t) = N$ наступает в момент $c_y(t) = c_2$.

Доказательство теорем опускаем ввиду ограниченного объема статьи.

Если в процессе движения по ограничению $n(t) = N$ в какой-то точке $c_y = 0$ и $\dot{q} \ne 0$, то на такой траектории в последующий момент происходит нарушение ограничения. Это связано с тем, что локальное влияние на управление уже исчерпано; $\partial n/\partial c_y = 0$. При решении краевой задачи итеративными методами указанный факт ва-

жен прежде всего потому, что на некоторых пробных траекториях может происходить нарушение ограничения (1). Одновременно с этим возникают вычислительные трудности в построении итерационных методов расчета, поскольку из (10) следует, что $\lambda(t) \to \infty$ при $|c_y| \to 0$.

Краевая задача решалась с использованием параллельного варианта метода Ньютона. Матрица Якоби вычислялась с применением параллельного алгоритма на суперкомпьютере МСЦ РАН.

Работа выполнена при поддержке РФФИ (проекты №10-08-00624, №11-07-00201).

Литература

1. Афанасьев А.П., Дикусар В.В., Милютин А.А., Чуканов С.В. Необходимое условие экстремума. М., 1990. 320 с.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ГРАДИЕНТНОГО БУСТИНГА ДЕРЕВЬЕВ РЕШЕНИЙ

П.Н. Дружков

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача обучения с учителем, приводится краткое описание параллельного алгоритма обучения дерева решений, а также алгоритма градиентного бустинга деревьев решений, применимых для случая распределенного хранения данных, предназначенных, в первую очередь, для решения больших задач машинного обучения. Приводятся результаты вычислительных экспериментов с помощью созданной программной реализации рассматриваемых алгоритмов.

Введение

Обучение с учителем, или обучение по прецедентам, заключается в восстановлении неизвестной общей зависимости между набором входных переменных и некоторым целевым (выходным) показателем на основании конечного набора пар вида «вход–выход». Чем больше прецедентов доступно на этапе обучения, тем большим объемом информации о восстанавливаемой зависимости располагает алгоритм, и, следовательно, может построить более точную модель зависимости. Таким образом, сама возможность применения алгоритма машинного обучения в случае наличия большой обучающей выборки может повысить качество решения задачи. Данная тенденция поддерживается и с другой стороны. В последнее время появляются все более и более объемные наборы данных, связанные с важными прикладными задачами: от извлечения информации из поисковых Интернет-запросов и Интернет-рекламы [1] до интеллектуального анализа изображений и видео [2]. В связи с этим возникает необходимость модификации существующих алгоритмов или разработки новых, ориентированных именно на обработку больших наборов данных. Данная работа посвящена модификации, позволяющей обрабатывать распределенно хранящиеся данные, одного из наиболее перспективных алгоритмов решения задачи обучения с учителем: градиентного бустинга деревьев решений [3].

1. Постановка задачи

Одной из задач, изучаемой в машинном обучении, является *задача обучения с учителем*. В рамках этой задачи дано некоторое множество *объектов* (пространство признаков) X. Каждому объекту $x \in X$ поставлена в соответствие величина y, называемая *выходом*, *ответом* или *целевой переменной* и принадлежащая множеству допустимых ответов Y. Упорядоченная пара «объект–ответ» (x,y), где $x \in X$, $y \in Y$, называется *прецедентом*. Требуется восстановить зависимость между входом и выходом, основываясь на данных о конечном наборе прецедентов, называемом *обучающей выборкой*: $\{(x_i, y_i) \mid x_i \in X,\ y_i \in Y,\ i = 1, ..., n\}$. Другими словами, задача состоит в построении функции f из некоторого множества K, которая, получив на вход x, предсказала бы значение ответа y как можно точнее. В случае конечного Y говорят о *задаче классификации*, если $Y = \mathbf{R}$ – *задаче восстановления регрессии* [4]. Процесс нахождения f называется *обучением* (*тренировкой*, *настройкой*) модели, процесс определения выхода по некоторому входу с помощью уже построенной модели – *предсказанием*.

2. Описание алгоритма

Одними из наиболее популярных и перспективных на данный момент подходов в машинном обучении являются алгоритмы, основанные на построении деревьев решений и их ансамблей. Дерево решений строит разбиение пространства признаков на непересекающиеся области с помощью рекурсивной процедуры [5]. Фактически каждому узлу дерева соответствует некоторая область пространства признаков и правило, по которому осуществляется ее разделение на две, приписанные к дочерним вершинам. Вид правила разбиения зависит от типа участвующей в нем переменной: для количественного признака s это неравенство вида $x^{(s)} < v^{(s)}$, где $x^{(s)}$ – s-я координата вектора входных переменных x, а $v^{(s)}$ – некоторое пороговое значение; для номинального признака правило имеет вид $x^{(s)} \in P$, где P – подмножество множества возможных значений переменной $x^{(s)}$. Для обучения данной модели используется жадная стратегия максимального уменьшения функции, описывающей неоднородность данных $I(D)$, где D есть область, соответствующая некоторому узлу. В качестве $I(D)$ могут применяться различные функции: ошибка классификации, индекс Джини и т.д.

Вход: $D = \{(x_i, y_i), i = \overline{1,n}\}$ – обучающая выборка.

Выход: Обученное дерево решений T.

T = **Обучить_дерево_решений**(D)

1. Если не выполнен критерий остановки:
   a. Найти оптимальное правило разбиения D на DL и DR, максимизирующее I(D)-(I(D_L)+I(D_R)).
   b. Выполнить разбиение DL.
   c. Выполнить разбиение DR.
2. Иначе
   a. Найти значение, приписанное листу дерева.

Следует отметить, что поиск оптимального разбиения по количественной переменной производится для всех $v^{(s)} \in V^{(s)} = \left\{\frac{\left(\hat{x}_j^{(s)}+\hat{x}_{j+1}^{(s)}\right)}{2} : \hat{x}_j^{(s)} \le \hat{x}_{j+1}^{(s)}, j = \overline{1,n-1}\right\}$, где $\hat{x}^{(s)}$ – упорядоченные значения s-й переменной. В данной работе рассматривается алгоритм построения регрессионного дерева и квадратичная функция $I(D) = \sum_{i:x_i \in \mathrm{D}} (y_i - \bar{y})^2$. В данном случае выражение для уменьшения функции неоднородности упрощается до $\frac{\left(\sum_{i:x_i \in D_L} y_i\right)^2}{|D_L|} + \frac{\left(\sum_{i:x_i \in D_R} y_i\right)^2}{|D_R|}$, а каждому листу приписывается значение, равное среднему арифметическому значений целевой переменной попавших в него прецедентов.

Пусть обучающая выборка разделена на несколько частей $R_1, R_2, \dots, R_T$ и каждая часть обрабатывается отдельным процессом независимо от остальных, в то время как для синхронизации их работы служит один управляющий процесс. Рабочий процесс для каждой переменной s и значения $v^{(s)}$ вычисляет пару $\alpha_t^{(s)} = \left(\sum_{i:x_i \in D \cap R_t} 1, \sum_{i:x_i \in D \cap R_t} y_i\right)$, которая передается главному процессу для выбора глобально-оптимального разбиения в данной вершине дерева. Следует отметить, что поскольку каждому отдельно взятому процессу не доступна вся выборка, вычислить все элементы множества $V^{(s)}$ для количественной переменной s не представляется возможным. В связи с этим предлагается перед процессом обучения вычислить для каждой количественной переменной элементы множества $V^{(s)}$ как квантили уровней $\frac{i}{q}$, $i = \overline{1, q-1}$, для относительно небольшого фиксированного числа q. Вычисление значений в листьях дерева, а именно вычисление среднего арифметического, легко выполняется в случае распределенного хранения данных.

Как уже отмечалось ранее, практический интерес представляет не только обучение одиночных деревьев решений, но и построение моделей в виде их ансамблей. Так, бустинг-алгоритмы последовательно строят базовые модели (в настоящей работе – деревья решений) таким образом, что каждая следующая корректирует ошибки предыдущих, тем самым улучшая качество модели. Одним из наиболее общих методов этого направления является алгоритм градиентного бустинга деревьев решений, который использует аналогию с методом градиентного спуска для построения ансамбля деревьев решений, минимизирующего ошибку на обучающей выборке, заданную дифференцируемой неотрицательной функцией потерь (штрафа) $L(y, y')$. Данный алгоритм представляет собой итерационный процесс, на каждом шаге которого строится одиночное дерево решений, наилучшим образом предсказывающее не исходные значения целевой переменной, а антиградиент функции потерь $-\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=h_{m-1}}$, зависящий от истинных значений выходного признака и предсказаний модели, построенной к текущей итерации. Следует отметить, что каждое дерево решений обучается с использованием квадратичного штрафа, что не требует сильной модификации алгоритма его построения, за исключением вычисления значений, приписанных листьям уже построенного дерева. Листовые константы бустинг-дерева находятся с учетом функции потерь, используемой в алгоритме градиентного бустинга, из соображений минимизации потерь в листе: $\arg\min_{\rho} \sum_{i:\, x_i \in \text{leaf}} L(y_i, \rho)$. Следует отметить, что, в отличие от отыскания градиента, вычисление значений в листовых вершинах задача гораздо более сложная, в связи с чем зачастую применяются приближенные решения.

Таким образом, применение алгоритма градиентного бустинга деревьев решений, в случае когда данные расположены распределенно, требует, помимо алгоритма обучения одиночных деревьев решений, также модифицированных методов пересчета значений в листовых вершинах деревьев и вычисления антиградиента на каждой итерации. В общем случае обе эти процедуры зависят от конкретного вида функции потерь.

3. Программная реализация

В соответствии с описанными выше схемами были созданы параллельные программные реализации как алгоритма обучения одиночного дерева решений, так и алгоритма градиентного бустинга деревьев решений. Для последнего были реализованы следующие функции потерь [3] для задач восстановления регрессии: абсолютная, квадратичная, функция Хьюбера – и для классификации: логистическая и кросс-энропия. Как уже отмечалось выше, алгоритм вычисления антиградиента и листовых констант зависит от используемой функции потерь, и даже для реализованного набора целиком осуществить эти операции независимо над каждой частью данных не всегда представляется возможным. В таких случаях приходится пересылать необходимую информацию на управляющий процесс, где целиком производится вычисление необходимых значений. В остальных случаях б*о*льшая часть работы осуществляется параллельно рабочими процессами, после чего результирующее значение вычисляется управляющим процессом.

Для обеспечения параллелизма используется технология MPI.

4. Вычислительный эксперимент

Для оценки эффективности описанной программной реализации был проведен ряд вычислительных экспериментов. Для экспериментов использовался набор данных Million Song Dataset, взятый с репозитория UCI [6], и искусственно сгенерированный набор данных. Все эксперименты проводились на вычислительном кластере МГУ «Ломоносов». Модель градиентного бустинга деревьев решений обучалась с применением

функции потерь Хьюбера, строился ансамбль из 500 деревьев, на каждой итерации использовалось 70% от всей обучающей выборки.

Million Song Dataset представляет собой набор данных для предсказания года создания музыкальной композиции по ее характеристикам. Набор данных состоит из 515345 прецедентов, каждый из которых содержит 90 предикативных числовых переменных и числовую целевую переменную. Для каждой переменной было вычислено по 9 статистических квантилей значений в обучающей выборке. Несмотря на то, что данные предразбиты на обучающую и тестовую части, для оценки эффективности программной реализации параллельного алгоритма обучения модели градиентного бустинга обе выборки использовались для обучения, что позволило увеличить объем обрабатываемых данных. График масштабируемости алгоритма приведен на рис. 1(a). Как видно из приведенных результатов, параллельный алгоритм обеспечивает ускорение, близкое к линейному, или даже выше. Сверхлинейное ускорение в данном случае можно объяснить нелинейным характером трудоемкости последовательного алгоритма, а также более эффективным использованием памяти процессами при уменьшении объема обрабатываемых ими данных.

Для генерации искусственной выборки использовалась функциональная зависимость целевого признака от 100 предикативных $f(x_1, \dots, x_{100}) = \sum_{i=1}^{15} \exp(-2x_i^2) + \sum_{i=16}^{100} x_i$. Также, для повышения размерности пространства, было добавлено 2400 фиктивных (независимых с целевой) переменных. Для формирования обучающей выборки было сгенерировано 50000 точек, независимо и равномерно распределенных в 2500-мерном гиперкубе. Так как все признаки являются числовыми, для использования параллельного алгоритма обучения модели градиентного бустинга деревьев решений было подсчитано по 9 статистических квантилей для каждой переменной. Время работы параллельного алгоритма на разном количестве процессов приведено на рис. 1(b). Также следует отметить, что рассматриваемая параллельная схема и последовательный алгоритм градиентного бустинга не являются полностью эквивалентными из-за сужения множества рассматриваемых разбиений по количественным переменным в деревьях решений. Для сравнения качества решения задачи вся выборка была разбита на 2 части в соотношении 70 к 30 и были обучены бустинг-модели с помощью последовательного [7, 8] и параллельного алгоритмов, после чего была вычислена средняя квадратичная ошибка на тестовой выборке. Для данной задачи ошибка сохранилась на уровне 6.4, а следовательно, можно говорить о сохранении качества решения задачи с помощью параллельной модификации алгоритма.

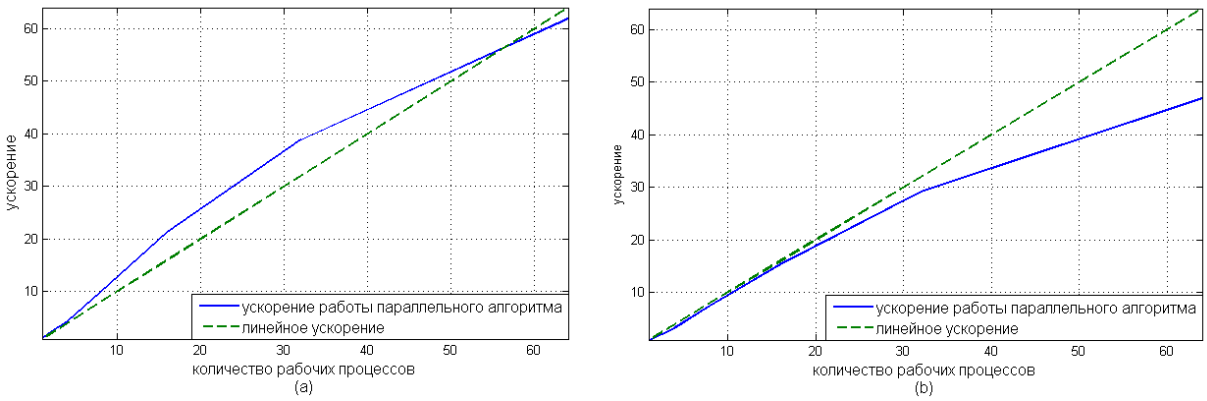


Рис. 1. Ускорение работы параллельного алгоритма обучения модели градиентного бустинга деревьев решений с ростом количества рабочих процессов (a) на данных Million Song Dataset, (b) на искусственных данных

Полученный результат свидетельствует о хорошей масштабируемости даже для относительно небольшой обучающей выборки, что позволяет использовать данный алгоритм не только для больших наборов данных, но также и для параллельного обучения модели градиентного бустинга. Разница в получаемых показателях ускорения для рассмотренных задач обусловлена не только размером выборки (количеством прецедентов), но и количеством предикативных переменных: случай, когда количество прецедентов значительно больше количества переменных, является наиболее подходящим для рассматриваемого алгоритма.

Работа выполнена в рамках программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы», государственный контракт № 11.519.11.4015.

Литература

1. Panda B., Herbach J.S., Basu S., Bayardo R.J. PLANET: Massively parallel learning of tree ensembles with MapReduce // Proceedings of the 35th International Conference on Very Large Data Base (VLDB). 2009. P. 1426–1437.
2. Dollar P., Wojek C., Schiele B., Perona P. Pedestrian Detection: A Benchmark // Computer Vision and Pattern Recognition (CVPR’09). 2009.
3. Friedman J. Greedy function approximation: the gradient boosting machine // Annals of Statistics. 2001. V. 29, N. 5. P. 1189–1232.
4. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. – Springer, 2008.
5. Breiman L., Friedman J.H., Olshen R.A., Stone C.J. Classification and Regression Trees. Wadsworth & Brooks, 1984.
6. UCI Machine Learning Repository – [http://archive.ics.uci.edu/ml].
7. Дружков П.Н., Золотых Н.Ю., Половинкин А.Н. Программная реализация алгоритма градиентного бустинга деревьев решений // Вестник Нижегородского государственного университета им. Н.И. Лобачевского. 2011. № 1. С. 193–200.
8. Druzhkov P. N., Eruhimov V. L., Kozinov E. A., Kustikova V. D., Meyerov I. B., Polovinkin A. N., Zolotykh N. Yu. On some new object detection features in OpenCV Library // Pattern Recognition and Image Analysis. 2011. V. 21, № 2. P. 377–379.

СПОСОБЫ ВЕКТОРИЗАЦИИ ВЫЧИСЛЕНИЙ ДЛЯ СИСТЕМ С АРИФМЕТИЧЕСКИМИ УСКОРИТЕЛЯМИ

Т.А. Жирнова, А.С. Рыбкин

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

В настоящее время актуальной является задача адаптации существующих или разработки новых комплексов программ и библиотек, эффективно работающих на системах с графическими процессорами – арифметическими ускорителями.

Существует класс задач, в которых требуется расчет различных функций на некотором массиве исходных данных. В докладе проведено исследование влияния условного перехода на длительность исполнения данного класса задач на примере элементарной функции, реализующей операцию умножения-сложения FMA. В случае расположения условного перехода внутри функции ядра доказано критическое влияние ветвлений внутри одного варпа на длительность счета. В случае расположения условного перехода снаружи функции ядра показана эффективность использования предварительной группировки исходных данных.

Подробно рассмотрены два способа векторизации вычислений для систем с арифметическими ускорителями. Первый способ представляет собой последовательный вызов всех функций на полном массиве исходных данных. Второй способ предполагает предварительную группировку исходных данных. Реализация способов выполнена на примере функций библиотеки УРС-ОФ. Сделаны выводы о целесообразности использования реализованных способов.

Реализованные способы векторизации вычислений подходят большому классу задач и могут использоваться для адаптации программ для систем с арифметическими ускорителями.

ВЫЧИСЛИТЕЛЬНАЯ РЕАЛИЗАЦИЯ ОБОБЩЕННОГО МЕТОДА DIRECT ДЛЯ ЗАДАЧИ С ФУНКЦИОНАЛЬНЫМИ ОГРАНИЧЕНИЯМИ

А.Ю. Жунин

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассмотрен вариант вычислительной реализации обобщения метода Direct [1] на задачи с липшицевыми функциональными ограничениями при неограниченном диапазоне значений констант Липшица. Данное обобщение предложено в работе [2], в которой приведено теоретическое описание и обоснование двух возможных подходов. Здесь реализован второй подход. В докладе будут приведены результаты вычислительных экспериментов.

Введение

Метод Direct, предложенный в работе [1], разработан для задач безусловной липшицевой многоэкстремальной оптимизации с неограниченным диапазоном изменения константы Липшица. Описание метода, а также некоторые его модификации можно также найти в [3]. Метод Direct не рассчитан на более широкий класс задач с функциональными ограничениями.

В работе [2] предложены два специальных подхода к распространению метода на эти более сложные задачи. Второй из этих подходов непосредственно обобщает концепции метода Direct на случай наличия функциональных ограничений. При этом используется предположение о том, что неизвестная константа L липшицевой минимизируемой функции f, где $L \in [0, \infty)$, связана с константой L_g липшицевой функции g обобщенного ограничения соотношением $L_g \in [0, \gamma L],\ \gamma > 0$.

Постановка задачи

Задача условной многоэкстремальной оптимизации рассматривается в следующей постановке:

$$f^* = f(x^*) = min f(x),\ \ x \in X, \tag{1}$$

$$X = \{x \in D = [a, b] \subset R^N:\ g_i(x) \le 0,\ (i = \overline{1, m})\}. \tag{2}$$

Набор функциональных ограничений сводится к одному обобщенному ограничению

$$g(x) \le 0, \tag{3}$$

где

$$g(x) = \max\ \{g_1(x), \dots, g_m(x)\}. \tag{4}$$

Предполагается, что функция f является липшицевой с константой L, значение L неизвестно и $L \in [0, \infty)$. Функция g липшицева с константой L_g, значение константы неизвестно, но взаимосвязано с L соотношением $L_g \in [0, \gamma L]$, где $\gamma > 0$.

Также как и метод Direct, рассматриваемое обобщение реализовано как компонентный метод, основанный на динамическом разбиении исходного множества-гиперинтервала $D = [a, b]$ на гиперинтервалы $D_i = [a_i, b_i]$ меньшего размера по центральной схеме деления на три. На каждой итерации совокупность гиперинтервалов $\{D_i\}$ полностью покрывает D не пересекаясь при этом по внутренним точкам. В центре

каждого гиперинтервала D_i вычисляются как значения целевой функции так и обобщенного ограничения, обозначаемые далее как f_i и g_i.

В [2] предложен специальный принцип выбора подмножества «лучших» (*потенциально оптимальных*) гиперинтервалов D_t, которые должны подвергаться очередному делению на три (по большему ребру) на текущей итерации. Такие гиперинтервалы выделяются из множества недоминируемых. *Недоминируемыми* в [2] названы такие гиперинтервалы D_t, что:

$$\exists\, L \in [0,\infty) : \; f_t - L\frac{d_t}{2} = \min\left\{f_i - L\frac{d_i}{2} : \exists\, L_g \in [0, L*\gamma] : g_i - L_g\frac{d_i}{2} \le 0\right\}, \qquad (5)$$

$$\exists\, L_g \in [0, L\gamma] : g_t - L_g\frac{d_t}{2} \le 0. \qquad (6)$$

Потенциально оптимальные это такие недоминируемые гиперинтервалы D_t, для которых выполнено дополнительное условие, препятствующее чрезмерному дроблению гиперинтервалов малого диаметра

$$f_t - L\frac{d_t}{2} \le f_{min} - \varepsilon\left|f_{min}\right|. \qquad (7)$$

Здесь под f_{min} понимается текущая найденная оценка условного глобально-оптимального значения целевой функции.

В [2] изложены теоретические основы реализации правила отбора (6)-(7) с помощью трех вспомогательных этапов. В следующих разделах описаны принципы их программной реализации, приводящие к выделению подмножества потенциально оптимальных гиперинтервалов.

Описание алгоритма

Алгоритм состоит из трех основных частей: обработка «слоев», создание общей таблицы недоминируемы в «слоях» гиперинтервалов, обработка таблицы с выделением потенциально оптимальных.

Выполнение метода начинается с того, что начальный гиперинтервал D делится на 3 равные части (по большему ребру). Это задает стартовый набор гиперинтервалов, образующих первый слой. *Слоем* будем называть подмножество гиперинтервалов с одинаковыми значениями d_i.

Далее описаны действия, выполняемые на некоторой промежуточной итерации.

Обработка слоя

Гиперинтервалы слоя с $d_i = c$ ассоциируются с набором точек (g_i, f_i). Те из них, у которых $g_i < 0$, будем называть *допустимыми*. Обработка допустимого подмножества гиперинтервалов заключается в отыскании среди них минимальной по значению целевой функции. Обозначим это минимальное значение через f_{min}^c.

Подмножество гиперинтервалов, соответствующих точкам с $g_i \ge 0$,будем называть *потенциально допустимыми*, множество их номеров обозначим через P.

Обработка множества потенциально допустимых точек заключается в отыскании в нем подмножества точек с номерами $i \in B$, каждая точка которого удовлетворяет следующему условию.

Если $\mathrm{i} \in \mathrm{B}$, то $\mathrm{f_i} \le \mathrm{f_{min}^c}$ и $\nexists (j \in P)\&(j \ne i)$: $(f_i < f_j)\&(g_i \le g_j)$.

Как видим, одним из условий обработки является то, что отобранные потенциально допустимые точки должны иметь значения целевой функции меньше, чем минимальное значение среди допустимых гиперинтервалов слоя. С допустимыми отобранными точками связывается некоторое «минимально допустимое» значение константы Липшица L_i, равное нулю, а с отобранными потенциально допустимыми – «минимально допустимое» значение константы Липшица, равное

$$L_i = \frac{2g_i}{d_i\gamma}. \qquad (8)$$

Отобранные допустимые и потенциально допустимые гиперинтервалы записываются в таблицу вместе со своим ключом L_i, являющимся «минимально допустимым» для них значением константы Липшица. Таблица упорядочивается по ключу. Значением в таблице по данному ключу является список гиперинтервалов с одинаковым значением ключа.

Такие таблицы строятся для каждого из существующих слоев.

Образование общей таблицы

Готовые таблицы слоев обрабатываются поочередно. Обход каждой из таблиц в отдельности начинается с последней строки. Запоминается значение ключа текущей строки L_k и значение ключа уже обработанной строки L_{k-1}. В случае когда обработка только началась (т.е. текущая строка – последняя в таблице), предполагается, что ранее обрабатывалась строка с бесконечным ключом $L_{k-1} = \infty$. По текущему ключу в общей таблице определяется строка, если она существует, в которую копируется список гиперинтервалов из строки обрабатываемой таблицы. Этот список также копируется в последующие строки общей таблицы с большими ключами до строки (не включительно) с ключом L_{k-1}, и если $L_{k-1} = \infty$, то копирование происходит до конца общей таблицы, включая последнюю строку.

Если в общей таблице не существует строки с ключом L_k, то такая строка создается. Затем происходит копирование списка гиперинтервалов по выше приведенному алгоритму для случая, когда строка существует, и копируется список из строки общей таблицы с ключом меньшим L_k в строку с ключом L_k.

Обработка общей таблицы

Структура построенной общей таблицы такова, что все интервалы из одной ее строки, в общем случае, относятся к разным слоям, и каждый из них является недоминируемым в своем слое для некоторого промежутка значений L. Причем в таблице для набора гиперинтервалов данной строки хранятся граничные значения L_{min} (ключ текущей строки в таблице) и L_{max} (ключ следующей строки таблицы), обладающие тем свойством, что вне этого промежутка хотя бы один из гиперинтервалов строки перестает быть недоминируемым в своем слое. Заметим, что когда обрабатывается последняя строка таблицы, L_{max} считается равной ∞.

Рассмотрим попарно гиперинтервалы из одной строки таблицы как точки на плоскости (f_e, g_e).

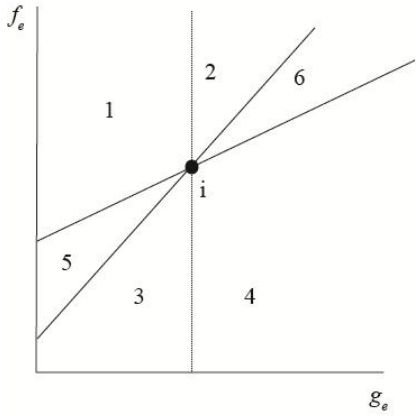


Рис. 1. Точка i с секторами

На рис. 1 показана схема секторов точки, выделяемых для принятия решения о недоминируемости точки из пары.

Если точка j попадает в сектора 1, 2 по отношению к точке i, то она уже не может быть недоминируемой в рассматриваемой группе, и точку j следует исключить из рассмотрения на данном этапе. Если j попала в сектора 3, 4, то исключить из рассмотрения на данном этапе следует точку i. В случае попадания j в сектор 5 вычисляется тангенс угла наклона прямой, проходящей через i и j, и он устанавливается в качестве нового значения L_{max} для j и в качестве нового значения L_{min} для i. При попадании в сектор 6 также вычисляется наклон аналогичной прямой, и тангенс угла наклона устанавливается как L_{max} для i и как L_{min} для j. Далее процесс продолжается до рассмотрения всех оставшихся пар точек в этой группе.

Дополнительная выбраковка оставшихся гиперинтервалов происходит при проверке условия (7), которое для них должно выполняться при соответствующем гиперинтервалу значении $L = L_{max}$.

При выполнении указанной процедуры для всех строк общей таблицы некоторые гиперинтервалы могут быть отобраны вторично. Повторное их включение в формируемый список потенциально оптимальных гиперинтервалов блокируется.

Все отобранные потенциально оптимальные гиперинтервалы разделяются на три равные части по своему большему ребру и в новых центрах проводятся дополнительные измерения функций задачи. Далее процесс обработки повторяется. Останов вычислений выполняется по исчерпанию отведенного вычислительного ресурса.

Во время доклада будут приведены результаты вычислительных экспериментов.

Работа выполнена при финансовой поддержке Минобрнауки России, государственное соглашение о предоставлении гранта № 14.B37.21.0878.

Литература

1. Jones D.R., Perttunen C.D., Stuckman B.E. Lipschitzian optimization without the Lipschitz constant // J. Optim. Theory and Appl. V. 79. № 1. 1993. P. 157–181.
2. Городецкий С.Ю. Обобщения метода Direct на задачи с функциональными ограничениями // Труды XII Всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах». 26-28 ноября 2012, Н.Новгород. – Н.Новгород: Изд-во ННГУ, 2012 (настоящий сборник).
3. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008. – 352 с.

РЕАЛИЗАЦИЯ РЕШЕНИЯ РАЗНОСТНОЙ СХЕМЫ РАСЩЕПЛЕНИЯ ДВУМЕРНОГО УРАВНЕНИЯ ТЕПЛОПРОВОДНОСТИ В КЛЕТОЧНОЙ МАШИНЕ

Г.В. Заручевская

Северный арктический федеральный университет, Архангельск

Господствующим способом распараллеливания задач до сих пор является крупноблочное. При этом задача разбивается на большие подзадачи (блоки), предназначенные для параллельного решения на небольшом числе процессоров. С ростом числа процессоров блоки измельчаются и вычисления в подавляющем большинстве случаев будут идти медленнее: параллелизм вырождается. Избежать вырождения можно только при условии, что обмены происходят и одновременно, и локально, т.е. физическое расстояние между взаимодействующими процессорами мало и не зависит от размера задачи. Задача должна быть разбита на множество небольших однотипных подзадач, которые будут исполняться параллельно на отдельных вычислительных машинах (ВМ). Данные максимально распределены по системе, а программы в каждой ВМ используют минимально возможные наборы данных. В общем случае число обменов данными между ВМ имеет тот же порядок, что и число вычислительных операций. Таким образом, мелкозернистость, или массовое распараллеливание, означает, что в каждом вычислительном процессе в каждый момент времени содержится минимальное число команд и данных. Такой подход к распараллеливанию алгоритмов носит название мелкозернистого локально-параллельного программирования (МЛПП).

Существуют три обязательных условия, при которых производительность МЛПП не снижается:

1. Локальность взаимодействий, когда обмен данными происходит только в пределах ограниченного физического и структурного радиуса, независимо от размеров задачи и системы.

2. Параллелизм взаимодействий, когда все возможные в данный момент обмены совершаются параллельно и одновременно с процессом счета.

3. Количество глобальных операций не должно влиять на оценку временной сложности задачи.

Опишем клеточную машину, предназначенную для МЛП-программирования.

Клеточная машина (К-машина) представляет собой *клеточное множество* – массив поименованных клеток, в каждой из которых находится универсальный вычислитель, известный в теории вычислений как РАМ-машина. Клетка может обмениваться информацией с соседями согласно шаблону соседства, задающему доступную для клетки окрестность. Клетка К-машины содержит память и коммутационное устройство для связи с соседями и управляющей машиной. *Шаблон соседства* для данной клетки – это список приращений её координат для получения координат *соседей*. Взаимодействовать должны только ближайшие соседи. *Условие локальности* означает, что радиус шаблона соседства – это константа, намного меньшая размера всего клеточного множества и не зависящая от этого размера. *Топология* К-машины задаётся шаблоном соседства. Одна из них – *тор*, которая получается, если мы склеим попарно и вертикальные, и горизонтальные границы прямоугольника. Тор позволяет масштабировать сетку другой, отличающейся от него размерности. Действительно, пусть сетка $a \times b$ вкладывается

в тор $C\times D$, $a>C$, $b>D$ (его обозначают $E_2\{C, D\}$). Каждой клетке тора (x, y) поставим в соответствие ячейку сетки $(x+Ck, y+Dn)$, $k, n \in N$. При таком распределении сеточных узлов в тороидальной структуре параллельный алгоритм соответствует МЛП-стилю.

Целью настоящей работы является вложение данных МЛПП-алгоритма решения сеточной задачи Коши для двумерного уравнения теплопроводности по схеме переменных направлений в тороидальную структуру синхронной клеточной машины с иллюстрацией межклеточных обменов.

Решение сеточной задачи для уравнения теплопроводности по схеме переменных направлений сводится к системе линейных уравнений вида:

$$\begin{cases} \tilde{u}_{m,n+1} - 2(1+\dfrac{h^2}{\tau})\tilde{u}_{m,n} + \tilde{u}_{m,n-1} = -u^p_{m+1,n} + 2(1-\dfrac{h^2}{\tau})u^p_{m,n} - u^p_{m-1,n} \\ u^{p+1}_{m+1,n} - 2(1+\dfrac{h^2}{\tau})u^{p+1}_{m,n} + u^{p+1}_{m-1,n} = -\tilde{u}_{m,n+1} + 2(1-\dfrac{h^2}{\tau})\tilde{u}_{m,n} - \tilde{u}_{m,n-1} - h^2 f^{p+1}_{mn} \end{cases}. \qquad (1)$$

Рассмотрим параллельный алгоритм решения задачи.

Этап 1. Хост-машина рассчитывает τ/h^2, $2(1+\tau/h^2)$ и $2(1-\tau/h^2)$, рассылает эти значения в каждую клетку тороидальной структуры; исходные данные u^0_{mn} и $h^2\cdot f^p_{mn}$ рассылаются в клетки (mod(m, C), mod(n, D)). Также машина вычисляет $\aleph_k$ и рассылает в клетки (mod(k, C); mod(n, D)) и (mod(m, C); mod(k, D)); m, n, k=1..(N-1).

Этап 2. Вычислим правую часть первого уравнения (1) в подобласти $\omega^{1,1}$. В каждом горизонтальном кольце системе сдвигом вправо передаются значения $u_{m,n}$ соседним клеткам, затем аналогичным сдвигом осуществляется передача этих данных влево. Далее каждая клетка с номером m рассчитывает

$$-u^p_{m+1,n} + 2(1-\frac{h^2}{\tau})u^p_{m,n} - u^p_{m-1,n} = F^p_{m,n} .$$

Подобным образом вычисляются значения $F^p_{m,n}$ в остальных площадках $\omega^{k,n}$. Заметим, что если рассчитываются значения в приграничных площадках $\omega^{1,s}$ или $\omega^{[(M-1)/C]+1,s}$, то клетки кольцевой структуры с номерами 1 и C друг с другом данными не обмениваются.

Этап 3. В каждом вертикальном кольце клеток, составляющих тор, рассчитываются $\tilde{u}_{mn}$, $m=\overline{1..C}$. Согласно методу прогонки рассчитываются: $\wp_1$ в клетках (mod(m, C), 1); $\wp_{N-1}$ в клетках (mod(m, C), mod(N-1, D)). Значение $\wp_1$ передается параллельным сдвигом вверх в каждой кольцевой структуре для расчета $\wp_2$ по формуле $\wp_k=(\wp_{k-1}-F_{m,k})\aleph_k$, k=2..N-2 в клетках (mod(k, C), mod(m, D)). Подобно расчету $\wp_{N-2}$ находятся остальные $\wp_k$, k=2..N-2. Затем вычисляются $\tilde{u}_{mN-1}=(\wp_{N-1} + \aleph_{N-1}\cdot\wp_{N-2})/(1-\aleph_{N-1}\cdot\aleph_{N-2})$ в клетках (mod(m, C), mod(N-1, D)) и сдвигом вниз пересылаются соседним клеткам для расчета $\tilde{u}_{mN-2}$ по формуле $\tilde{u}_{mn}, =\aleph_n\cdot\tilde{u}_{mn+1}+\wp_n$. Аналогично расчету $\tilde{u}_{mN-2}$ находятся остальные $\tilde{u}_{mn}$, n=(N-2)..2 .

Подобным образом находится следующая партия $\tilde{u}_{mn}$, $m=\overline{(C+1)..(2C)}$; затем при $m=\overline{(2C+1)..(3C)}$ и т.д., пока не будет рассчитана последняя партия при $m=\overline{\left[\dfrac{M-1}{C}\right] .. (M-1)}$.

Этап 4. Рассчитаем правую часть второго уравнения (1) в подобласти $\omega^{1,1}$. В каждом кольце, образованном клетками P_{mn} с одинаковыми номерами m, сдвигом вверх передаются значения $u_{m,n}$ соседним клеткам, затем аналогичным сдвигом осуществляется передача этих данных в противоположном направлении. Далее каждая клетка P_{mn} выполняет арифметические операции по формуле

$$-\tilde{u}_{m,n+1} + 2(1-\frac{h^2}{\tau})\tilde{u}_{m,n} - \tilde{u}_{m,n-1} - h^2 f_{mn}^{p+1} = \tilde{F}_n^{p+1}.$$

Подобным образом вычисляются значения $F_{m,n}^{p}$ в остальных площадках $\omega^{k,n}$. При расчете значений на приграничных площадках $\omega^{k,1}$ или $\omega^{k,\ [(N-1)/D]+1}$ клетки на кольцевых структурах со вторыми номерами 1, D друг с другом данными не обмениваются.

Этап 5. В каждом горизонтальном кольце клеток, составляющих клеточную матрицу, согласно методу прогонки рассчитываются u_{mn}. Сначала в клетках (1, mod(n, C)) вычисляются $\wp_1=(u_{0,n}^1-F_{1,n})\cdot\aleph_1$, а в клетках (mod(($M$–1), C), mod(n, C)) – $\wp_{M-1}=(u_{N,n}^1-F_{M-1,n})\cdot\aleph_{M-1}$. Значение $\wp_1$ передается параллельным сдвигом вправо в каждой кольцевой структуре для расчета $\wp_2$ по формуле $\wp_k=(\wp_{k-1}-F_{k,n})\aleph_k$, k=2..M-2 в клетках (mod(k, C), mod(n, D)), аналогично рассчитываются остальные $\wp_k$. Затем вычисляются $u_{M-1,n}=(\wp_{M-1}+\aleph_{M-1}\cdot\wp_{M-2})/(1-\aleph_{M-1}\cdot\aleph_{M-2})$ в клетках (mod(M–1, C), mod(n, D)), значение сохраняется и передается параллельным сдвигом влево для определения $u_{M-2,n}$ по формуле $u_{k,n}=\aleph_k\cdot u_{k+1,n}+\wp_k$. Аналогично, используя ту же формулу, находятся остальные u_{mn}.

Вычислительный процесс заканчивается, когда выполнен расчет последнего слоя u^T. Цикл повторяется T раз.

В рассмотренном алгоритме только второй и четвертый этапы выполняются в мелкозернистом локально-параллельном стиле. Третий и пятый этапы не являются мелкозернистыми. Легко видеть, что при такой организации параллельных вычислений большинство клеток на этапах 3 и 5 простаивают. Этот недостаток можно исправить, вычисляя прогоночные коэффициенты $\wp_k$, а затем и значения $\tilde{u}_{mn}$ (u_{mn}) «волной», т.е. каждая i-я вертикальная кольцевая структура на третьем этапе одновременно вычисляет значения $\tilde{u}_{i,n}$, $\tilde{u}_{i+C,n}$, $\tilde{u}_{i+2C,n}$, …, $\tilde{u}_{i+C(D-1),n}$, затем $\tilde{u}_{i+CD,n}$, $\tilde{u}_{i+C(D+1),n}$, $\tilde{u}_{i+C(D+2),n}$, …, $\tilde{u}_{i+C(2D-1),n}$ и т.д. Аналогично каждая j-я горизонтальная кольцевая структура на пятом этапе параллельно вычисляет значения u_{mj}, $u_{m,j+D}$, $u_{m,j+2D}$, … , $u_{m,j+D(C-1)}$, затем $u_{m,j+DC}$, $u_{m,j+D(C+1)}$, $u_{m,j+D(C+2)}$, … , $u_{m,j+D(2C-1)}$ и т.д. Такой «модернизированный» алгоритм на всех этапах (не связанных с глобальными операциями) отвечает требованиям локальности и мелкозернистости.

При $t_{\text{сдвиг}}\approx 0$ «модернизированный» вариант достигает максимального параллелизма, т.е. $E_{CD}^{/}\approx 1$.

В заключение отметим, что существуют другие алгоритмы распараллеливания методов решения разностных схем гиперболических уравнений, решаемых с помощью метода прогонки. Однако эти алгоритмы крупноблочные и к ним трудно применима технология мелкозернистого локально-параллельного программирования, и, как правило, они реализуются либо на кластерах, либо на планарных неразрезных процессорных матрицах. Предложенную идею можно использовать для распараллеливания других методов решения разностных схем гиперболических уравнений.

ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧИ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ СИСТЕМОЙ, ОПИСЫВАЕМОЙ ЛИНЕЙНЫМ ЭЛЛИПТИЧЕСКИМ УРАВНЕНИЕМ, ПРИ НАЛИЧИИ НЕЛОКАЛЬНЫХ ОГРАНИЧЕНИЙ НА СОСТОЯНИЕ СИСТЕМЫ

Д.Г. Залялов, А.В. Лапин

Казанский (Приволжский) федеральный университет

Рассматривается задача оптимального управления в правой части уравнения Лапласа при наличии поточечных ограничений на функцию управления и нелокальных ограничений на функцию состояния. Строится ее конечно-разностная аппроксимация, вводится сеточная функция Лагранжа, формулируется соответствующая седловая задача. Наряду с исходной седловой задачей рассматривается ее регуляризованный вариант. Приводятся выводы о существовании решений обеих седловых задач. Для решения регуляризованной седловой задачи применяется градиентный итерационный метод, для решения исходной задачи – предобусловленный метод Узавы. Устанавливается сходимость итерационных методов, описываются алгоритмы их численной реализации. Приводятся результаты числовых расчетов.

Введение

Данная работа является продолжением исследований в области построения, теоретического и численного анализа итерационных методов решения сеточных задач оптимального управления с ограничениями на функции управления и состояния ([1, 2]).

1. Постановка и аппроксимация задачи оптимального управления

Пусть задачей состояния является однородная задача Дирихле для уравнения Пуассона на единичном квадрате Ω с границей $\partial\Omega$:

$$\Delta y = u, x \in \Omega, \qquad y(x) = 0, x \in \partial\Omega, \tag{1}$$

где u – функция управления, решение y – состояние системы. Множества ограничений на функции управления и состояния зададим равенствами

$$U_{ad} = \{u \in L_2(\Omega): \left|\left|u(x)\right|\right| \le 1 \ \forall x \in \Omega\}, Y_{ad} = \left\{y \in H_0^1(\Omega): \int\limits_{\Omega} y(x)dx \le 1\right\},$$

а целевой функционал возьмем в виде

$$J(y,u) = \frac{1}{2}\int\limits_{\Omega}(y - y_d)^2 dx + \frac{r}{2}\int\limits_{\Omega} u^2 dx, r = const > 0, y_d \in L_2(\Omega).$$

Будем решать следующую задачу оптимального управления:

$$\min_{(y,u)\in K} J(y,u), \tag{2}$$

$$K = \{(y,u): y \in Y_{ad}, u \in U_{ad}, \text{выполнено уравнение (1)}\}.$$

Задача (2) имеет единственное решение (y,u).

Аппроксимируем задачу (2) конечно-разностной схемой на равномерной сетке $\vec{w} = \{x_{ij} = (ih, jh); 0 \le i,j \le n+1, (n+1)h = 1\}$. Конечно-разностные аппроксимации задачи состояния, множеств ограничений на сеточные функции управления u_h и состояния y_h и целевого функционала имеют вид:

$$\begin{cases} \dfrac{-y_{i-1j}+2y_{ij}-y_{i+1j}}{h^2}+\dfrac{-y_{ij-1}+2y_{ij}-y_{ij+1}}{h^2}=u_{ij}, 1\le i,j\le n, \\ y_{0j}=y_{j0}=y_{n+1j}=y_{jn+1}=0, \end{cases} \quad (3)$$

$$U_{ad}^h=\{u_h: |u_{ij}|\le 1, i,j=1,2,\dots,n\}, Y_{ad}^h=\left\{y_h: h^2\sum_{i,j=1}^{n} y_{ij}\le 1\right\},$$

$$J_h(y_h,u_h)=\frac{h^2}{2}\sum_{i,j=1}^{n}(y_{ij}-y_{d,ij})^2+\frac{rh^2}{2}\sum_{ij=1}^{n}u_{ij}^2.$$

В результате получаем конечномерную задачу оптимального управления

$$\text{найти } \min_{(y_h,u_h)\in K_h} J_h(y_h,u_h), \quad (4)$$

$$K_h=\{(y_h,u_h): y_h\in Y_{ad}^h\}, u_h\in U_{ad}^h, \text{выполнено уравнение (3)},$$

которая имеет единственное решение (y_h,u_h).

2. Конечномерные седловые задачи

Пусть множество внутренних узлов $x_{ij}, 1\le i,j\le n$, сетки ω каким-либо образом упорядочено. Поставим во взаимно однозначное соответствие сеточным функциям векторы из $|R^N, N=n^2$ их узловых параметров с координатами, соответствующими выбранному упорядочению узлов.

Система линейных уравнений (3) может быть записана в виде $Ly=u$ с симметричной и положительно определенной матрицей L – матрицей сеточного оператора Лапласа при нулевых граничных условиях Дирихле, множества ограничений принимают вид:

$$U_{ad}=\left\{u\in R^N: \sum_{i=1}^{N} h^2 y_i\le 1\ \forall i\right\},$$

а целевая функция после деления на $\frac{h}{2}$ равна $\frac{1}{2}\|y-y_d\|^2+\frac{r}{2}\|u\|^2$. Здесь и далее $||.||$ и $||.,.||$ – это евклидова норма и скалярное произведение в R^N. Обозначим через $\varphi(u)=$ $=I_{U_{ad}}(u)$ и $\theta(y)=I_{Y_{ad}}(y)$ индикаторные функции множеств U_{ad} и Y_{ad}. В итоге, сеточная задача оптимального управления (4) преобразуется к следующему виду:

$$\min_{Ly=u}\left\{J(y,u)=\frac{1}{2}\|y-y_d\|^2+\frac{r}{2}\|u\|^2+\varphi(u)+\theta(y)\right\}. \quad (5)$$

Определим функцию Лагранжа для задачи (5) равенством

$$\acute{L}(y,u)=\frac{1}{2}\|y-y_d\|^2+\frac{r}{2}\|u\|^2+\varphi(u)+\theta(y)-(Ly-u,\lambda). \quad (6)$$

Седловая точка функции Лагранжа является решением [3] системы, дающей условия оптимальности первого порядка:

$$\begin{pmatrix} E & 0 & -L \\ 0 & rE & E \\ -L & E & 0 \end{pmatrix}\begin{pmatrix} y \\ u \\ \lambda \end{pmatrix}+\begin{pmatrix} \partial\theta(y) \\ \partial\varphi(u) \\ 0 \end{pmatrix}\ni\begin{pmatrix} y_d \\ 0 \\ 0 \end{pmatrix}. \quad (7)$$

Задача (7) имеет решение (y,u,λ), при этом, пара (y,u) определяется однозначно и совпадает с решением задачи (5).

Рассмотрим наряду с задачей (5) регуляризованную задачу, аппроксимируя функцию $\theta(y)=I_{Y_{ad}}(y)$ дифференцируемой функцией $\theta_\varepsilon(y)=\frac{1}{2\varepsilon}((\sum_{i=1}^{N}h^2y_i-1)^+)^2$:

$$\min_{Ly=u}\left\{J(y,u)=\frac{1}{2}\left||y-y_d||^2+\frac{r}{2}\right||u||^2+\varphi(u)+\theta_\varepsilon(y)\right\}. \quad (8)$$

Соответствующая регуляризованная задача имеет вид:

$$\begin{pmatrix} E & 0 & -L \\ 0 & rE & E \\ -L & E & 0 \end{pmatrix}\begin{pmatrix} y_\varepsilon \\ u_\varepsilon \\ \lambda_\varepsilon \end{pmatrix} + \begin{pmatrix} \nabla\theta(y_\varepsilon) \\ \partial\varphi(u_\varepsilon) \\ 0 \end{pmatrix} \ni \begin{pmatrix} y_d \\ 0 \\ 0 \end{pmatrix}. \quad (9)$$

Задача (9) имеет единственное решение $(y_\varepsilon, u_\varepsilon, \lambda_\varepsilon)$, компоненты $(y_\varepsilon, u_\varepsilon)$ совпадают с решением (8).

3. Итерационные методы решения седловых задач

3.1. Градиентный метод решения регуляризованной задачи

Разрешив третье уравнение в системе (9) относительно y_ε и затем первое уравнение относительно λ_ε, получим включение для вектора u_ε (в дальнейшем опускаем индекс ε у вектора u_ε):

$$P_\varepsilon(u) + \partial\varphi(u) \ni L^{-1}y_d, P_\varepsilon = rE + L^{-2} + L^{-1}\nabla\theta_\varepsilon{}^\circ L^{-1}. \quad (10)$$

Применим для решения (10) одношаговый итерационный метод

$$\frac{u^{k+1} - u^k}{\tau} + P_\varepsilon u^k + \partial\varphi(u^{k+1}) \ni L^{-1}y_d. \quad (11)$$

Итерационный метод (11) совпадает с градиентным методом для задачи минимизации (8) и сходится при $0 < \tau < \frac{2\varepsilon}{\mu_{min}^{-2} + \varepsilon(r + \mu_{min}^2)}$,

где μ_{min} – минимальное собственное число сеточного оператора Лапласа.

Алгоритм реализации метода (11) состоит из следующих шагов:

1) для известного вектора управления u^k найти решение уравнения состояния $Ly^k = u^k$;

2) найти сопряженное состояние λ^k: $\lambda^k = L^{-1}\left(y^k - y_d + \nabla\theta_\varepsilon(y^k)\right)$;

3) найти новое приближение к вектору управления, решив включение

$$u^{k+1} + \tau\partial\varphi(u^{k+1}) \ni (1 + \tau r)u^k - \tau\lambda^k.$$

3.2. Предобусловленный метод Узавы для исходной задачи

Исключив векторы y и u в системе (7), получим уравнение для λ:

$$P(\lambda) \equiv L(E + \partial\theta)^{-1}(L\lambda + y_d) - (rE + \partial\varphi)^{-1}(-\lambda) = 0. \quad (12)$$

Применим для решения (12) итерационный метод

$$L^2\frac{\lambda^{k+1} - \lambda^k}{\tau} + P(\lambda^k) = 0, \quad (13)$$

являющийся предобусловленным методов Узавы, для отыскания седловой точки функции Лагранжа (6). Итерационный метод (13) сходится при условии $0 < \tau < \frac{2r}{r + \mu_{min}^{-2}}$.

При реализации этого метода выполняются следующие шаги:

1) для известного вектора λ^k найти y^k и u^k, решив включения

$$(E + \partial\theta)y^k \ni L\lambda^k + y_d \text{ и } (rE + \partial\varphi)u^k \ni -\lambda^k;$$

2) вычислить $p^k = L^{-1}u^k$;

3) решить уравнение $L\frac{\lambda^{k+1} - \lambda^k}{\tau} = -y^k + p^k$.

4. Результаты численных экспериментов

Вычислительные эксперименты были проведены для задачи с функцией наблюдения $y_d = 10 * (\sin x_1 + \sin x_2)$ и для разных весовых параметров r в целевой функции.

Критерием остановки для градиентного метода служило неравенство $||\partial_u^k||_{L_2}^2 < 10^{-5}$, а для метода Узавы – $||\partial_\lambda^k||_{L_2}^2 < 10^{-5}$. Здесь $||\partial_u^k||_{L_2}^2 < 10^{-5}$ – норма невязки во включении (10), а $||\partial_\lambda^k||_{L_2}^2 < 10^{-5}$ – норма невязки в уравнении (12).

Были реализованы два варианта алгоритмов, а именно:

а) решение на фиксированной сетке с нулевыми начальными приближениями («односеточный метод»);

б) решение на сетке в два раза более грубой с последующим использованием полученных результатов в качестве начальных приближений на исходной сетке («2-сеточный метод»).

В табл. 1 включены результаты вычислений градиентным методом с ε=0,02. При уменьшении ε число итераций и время вычислений растет очень быстро.

Таблица 1. Градиентный метод и метод Узавы

r=0.01	1 / *h*	**Односеточный метод** число итераций, время	**Двухсеточный метод** число итераций, время
Градиентный метод ε = 0.02	60 100 200	946, 00:00:31 950, 00:02:31 951, 00:22:28	1902, 00:00:26 1890, 00:01:49 1858, 00:13:48
Метод Узавы	60 100 200	540, 00:00:09 901, 00:00:34 1868, 00:04:42	815, 00:00:07 1339, 00:00:31 2748, 00:03:56

r=0.01	1 / *h*	**Односеточный метод** число итераций, время	**Двухсеточный метод** число итераций, время
Градиентный метод ε = 0.02	60 100 200	901, 00:00:30 901, 00:02:27 904, 00:20:46	1851, 00:00:27 1829, 00:01:49 1801, 00:13:15
Метод Узавы	60 100 200	955, 00:00:12 1615, 00:00:59 3411, 00:08:14	1415, 00:00:09 2369, 00:00:42 4973, 00:05:28

Работа выполнена при финансовой поддержке РФФИ, проект 10-01-00629.

Литература

1. Hinze M., Schiela A. Discretization of interior point methods for state constrained elliptic optimal control problems: Optimal error estimates and parameter adjustment // Comput. Optim. Appl. 2011. V. 48, No. 3. P. 581–600.
2. Laitinen E., Lapin A., Lapin S. On the iterative solution of finite-dimensional inclusions with applications to optimal control problems // Comp. Methods in Appl. Math. 2010. V. 10, No. 3. P. 283–301.
3. Лапин А.В., Хасанов М.Г. Решение задачи оптимального управления правой частью эллиптического уравнения при наличии ограничений на состояние // Ученые записки Казанского университета. 2010. Т. 152, Кн. 4. С. 56–67.
4. Lapin A. Preconditioned Uzawa type methods for finite-dimensional constrained saddle point problems // Lobachevskii J. Math. 2010. V. 31, No 4. P. 309–322.

РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ С ПРИМЕНЕНИЕМ МОДИФИЦИРОВАННОГО АЛГОРИТМА, ОСНОВАННОГО НА ДЕСКРИПТОРЕ КОНТЕКСТА ФОРМЫ

Р.К. Захаров

Самарский государственный аэрокосмический университет им. С.П. Королёва

Предлагается метод распознавания изображений, основанный на дескрипторе контекста формы. Предложенный алгоритм является улучшением классического метода, основанного на применении дескриптора контекста формы. Алгоритм находит на изображении объекты заданного подкласса (к примеру, колёса, двери, крышу для класса автомобилей) и принимает решение, являются ли эти объекты объектом класса автомобиль.

Введение

В данной работе стоит задача продемонстрировать алгоритм распознавания изображений, в качестве базы данных изображений была взята база данных The PASCAL Object Recognition Database Collection. На рис. 1 представлен пример тестового изображения.

Рис. 1. Пример тестового изображения из базы данных PASCAL

В работе представлен подход к поиску визуальных слов (BoW Bag of Words) на изображении, их распознаванию и принятию решения, является ли данный объект автомобилем.

Постановка задачи

Общая постановка задачи распознавания образов следующая. Предполагается, что имеется M изображений каждого из *K* объектов. Каждое изображение представляется вектором $x=[x_1,x_2,...,x_N]^T$ размерности *N*, где $x_1,x_2,...,x_N$ – признаки. Векторы, соответствующие изображениям одного объекта, составляют класс. Совокупность векторов признаков всех классов образует обучающую выборку. Решение задачи распознавания состоит в конструировании решающей функции $f:R^N \to \{0,1,...,K\}$, которая каждому вектору x ставит в соответствие некоторый класс. Для уменьшения числа неправильных классификаций вводится также класс с номером 0, соответствующий отказу в распознавании.

Качество распознавания зависит от выбора системы признаков. Наряду с выбором системы признаков большую роль играют также используемая при распознавании мера близости и построенное на ее основе решающее правило.

Для решения задачи распознавания и поиска объектов на изображении широко используется машина опорных векторов [4], а в качестве признаков наиболее популярными являются гистограммы ориентированных градиентов (HOG) [6], также часто используют метод Viola-Jones [3], который показывает хорошие результаты для отслеживания человеческих лиц в кадре, и различные методы, основанные на комбинировании алгоритмов.

В настоящей работе ставится задача провести исследования модифицированного алгоритма, основанного на дескрипторе контекста формы, для базы данных автомобилей. Метод, основанный на дескрипторе контекста формы, впервые ввели Serge Belongie и Jitendra Malik в работе [2] в 2000 году. Данный дескриптор широко используется в задачах распознавания цифр, букв, он показывал высокую надёжность распознавания на базе данных MNIST.

Новизна работы заключается в модификации классического алгоритма, основанного на дескрипторе контекста формы, к задаче распознавания изображения. Осуществляется поиск и распознавание не самого объекта, а визуальных слов, описывающих данный объект. Далее принимается решение, является ли эта совокупность визуальных слов распознаваемым объектом.

Описание модифицированного алгоритма, основанного на дескрипторе контекста формы изображения

Приведём описание алгоритма распознавания изображений, предложенного в работе, на рис. 2.

Рис. 2. Общая схема модифицированного алгоритма распознавания изображений

На рис. 3 представлена более подробная схема первого и второго этапов алгоритма, анализа найденных визуальных слов.

Рассмотрим подробнее каждый из этапов, изображённых на схеме.

1. Поиск визуальных слов на изображении. Реализуется через метод скользящего окна по изображению.

2. Анализ найденных визуальных слов. Осуществляется по схеме на рис. 3.
   a. Для каждого окна находятся ключевые точки на изображении. Алгоритм поиска ключевых точек произвольный: SIFT [5,7], SURF [8], FAST [9].
   b. Вычисляем дескриптор формы [1, 2] для каждой ключевой точки на изображении.
   c. Распознаём, является ли текущее окно визуальным словом. Для распознавания в работе используется машина опорных векторов SVM [4] для увеличения скорости принятия решения данный шаг алгоритма был распараллелен. Одно окно проверялось одновременно на сопоставление всем визуальным словам.
3. Процедура принятия решения, принадлежит ли объект распознаваемому классу, строится на основе количества найденных визуальных слов, их взаимного расположения и типов найденных визуальных слов.

Рис. 3. Схема поиска и распознавания визуальных слов на изображении

Заключение

В работе представлен подход к распознаванию изображений с использованием дескриптора контекста формы. Предложенный алгоритм является модификацией классического метода, основанного на дескрипторе контекста формы для изображения. Алгоритм является достаточно эффективным и надёжным методом для распознавания объектов на изображениях.

Литература

1. Belongie S., Malik J. Matching with Shape Contexts // IEEE Workshop on Contentbased Access of Image and Video Libraries (CBAIVL-2000). 2000.
2. Belongie S., Malik J., Puzicha J. Shape Matching and Object Recognition Using Shape Contexts // IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (24):509 521.DOI : 10.1109 / 34.993558. April 2002.

3. Viola, Jones. Rapid object detection using a boosted cascade of simple features // Computer Vision and Pattern Recognition. 2001.
4. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines John C. Platt Microsoft Research jplatt@microsoft.com Technical Report MSR-TR-98-14 April 21. 1998 © 1998 John Platt.
5. Lowe D.G. Object recognition from local scale-invariant features // Proceedings of the International Conference on Computer Vision. 1999. 2. P. 1150–1157. DOI:10.1109 / ICCV.1999.790410.
6. Dalal N., Triggs W. Histograms of Oriented Gradients for Human Detection // IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05. 2005. V. 1(3). P. 886-893.
7. Lowe D.G. Distinctive Image Features from Scale-Invariant Keypoints // International Journal of Computer Vision. 2004. 60, 2. P. 91-110.
8. Bay H., Ess A., Tuytelaars T., Van Gool L. SURF: Speeded Up Robust Features // Computer Vision and Image Understanding (CVIU). 2008. Vol. 110. No. 3. P. 346–359.
9. Trajkovic M., Hedley M. Fast corner detection // Image and Vision Computing. 1998. 16 (2). P. 75–87.

МОДЕЛИРОВАНИЕ ДИНАМИКИ ИММУННОГО ОТВЕТА В АНСАМБЛЯХ Т-КЛЕТОК С ПРИМЕНЕНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Д.Ю. Зорин, М.В. Иванченко

Нижегородский госуниверситет им. Н.И. Лобачевского
laonden@gmail.com, ivanchenko@rf.unn.ru

Исследован процесс формирования иммунного ответа в математической модели многокомпонентных ансамблей пулов эффекторных и наивных Т-клеток. Изучены различные стратегии иммунопротеасомной нарезки пептидов, изменения динамики численности инфицированных клеток в зависимости от параметров модели и выбранной стратегии. Применена технология параллельных вычислений на графических процессорах.

Введение

В настоящее время математическое моделирование динамики клеточных популяций приобретает все больший вес в теоретических и прикладных биомедицинских задачах. Это связано с тем, что постановка прямых экспериментов *in vitro* и *in vivo* зачастую крайне затруднительна, если вообще возможна. В полной мере это относится к исследованиям в области иммунологии, в частности, динамики иммунного ответа, где широко применяются эксперименты *in silico* [1].

Ранее мы исследовали сложную переходную динамику установления равновесия в отсутствие инфекции [2, 3]. Сейчас мы обратимся к переходной динамике иммунного ответа. Здесь особый интерес вызывает вопрос контекстно-зависимого производства пептидов и, соответственно, презентации антигенов, универсальным инструментом которых является протеасома – белковый комплекс. Существует несколько видов протеасом, среди которых: конституционная – доминирует в отсутствие инфекции, и иммунная – при появлении инфекции.

Цель данной работы – моделирование различных возможных способов иммунопротеасомной «нарезки» белков на пептиды, проверка состоятельности гипотезы об усилении эффективности иммунного ответа в рамках математического анализа динамики многокомпонетной популяции.

Математическая модель

Была сформулирована математическая модель, которая описывает конкурентную динамику наивных и эффекторных клеток в отсутствие и в присутствии инфекции и представляет собой систему обыкновенных дифференциальных уравнений [4, 5]:

$$\frac{dN_i}{dt} = H_i + (2\phi - 1)\rho_N \sum_{j=1}^{N_{pep}} K_{ij} F_j N_i - \delta_N N_i, \tag{1}$$

$$\frac{dE_i}{dt} = 2(1-\phi)\rho_N \sum_{j=1}^{N_{pep}} K_{ij} F_j N_i + \rho_E \sum_{j=1}^{N_{pep}} K_{ij} F_j E_i - \delta_E E_i, \tag{2}$$

$$F_j = \frac{M_T P_j}{1 + \sum_{i=1}^{N_{cline}} K_{ij}(N_i + E_i)}, \tag{3}$$

где N_i – количество клеток в i-м клонотипе пула наивных клеток; E_i – количество клеток в i-м клонотипе пула эффекторных клеток; F – количество свободного места на АПК; P – пептидная концентрация (доля пептида); K – матрица силы связей i-го клонотипа с j-м пептидом (часть которых может быть патогенна, а остальная – нейтральна); δ_N, δ_E – скорость вымирания наивных и эффекторных клеток соответственно; ρ_N, ρ_E – коэффициенты пролиферации; φ – доля обновления наивных клеток.

Чужеродный патогенный пептид представлен в модели только тогда, когда появляется инфекция и, соответственно, инфицированные клетки. При больших количествах инфицированных клеток доля патогенного пептида достигает максимальной величины: $P_v(t) = \frac{P_0 I(t)}{I(t)+\theta}$, здесь θ – чувствительность презентации патогенов. Предполагается наличие только одного патогенного пептида.

Динамика количества инфицированных клеток I описана стандартной моделью [4, 5]: $\frac{dI}{dt} = r(1-cI)I - D(\sum_{i=1}^{N_{clone}} K_{iP}E_i)I$, где r – коэффициент репликация патогена; c – объем патогена; D – скорость удаления инфицированных клеток.

Моделирование механизма иммунопротеасомной «нарезки»

Мы выдвинули гипотезу, что именно степенной, негауссов закон распределения пептидных концентраций объясняет эффективность перераспределения пептидных концентраций для большого числа клонов.

Мы проанализировали несколько возможных сценариев иммунопротеасомного регуляторного механизма для более реалистичного случая с количеством клонотипов и пептидов по 10^3. Поскольку моделирование подобных систем с большим количеством взаимодействующих элементов требует больших вычислительных ресурсов, мы использовали технологию параллельных вычислений на графических процессорах (CUDA).

Начальные значения численностей клонотипов эффекторных и наивных клеток, а так же начальные пептидные концентрации во всех реализациях модели задавались одинаковым набором данных. Появление инфекции моделировалось только при достижении системой гомеостаза.

Посредством численных экспериментов установлено, что аппроксимация случайного изменения концентраций пептидов, которые производят различные виды протеасом, позволяет получить увеличение численности эффекторных клеток, распознающий патогенный пептид и вместе с тем увеличение эффективности иммунного ответа (рис.1, 2). Впервые показано, что ключевым элементом этого механизма является степенное (негауссово) распределение пептидных концентраций и размеров клонов.

Дальнейшее исследование предполагает создание модели предсказания продуктов иммунопротеасомной нарезки и сплайсинга.

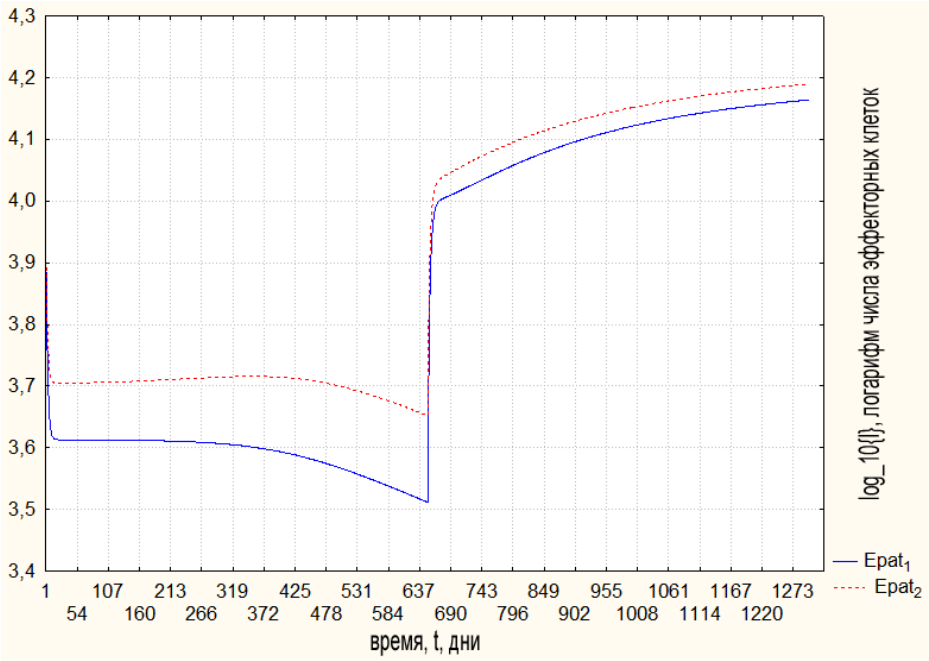


Рис. 1. Динамика численности эффекторных клеток, распознающих патогенный пептид. Время появления инфекции – 650. Оба случая имеют одинаковые начальные условия, кроме матрицы пептидных связей *K*

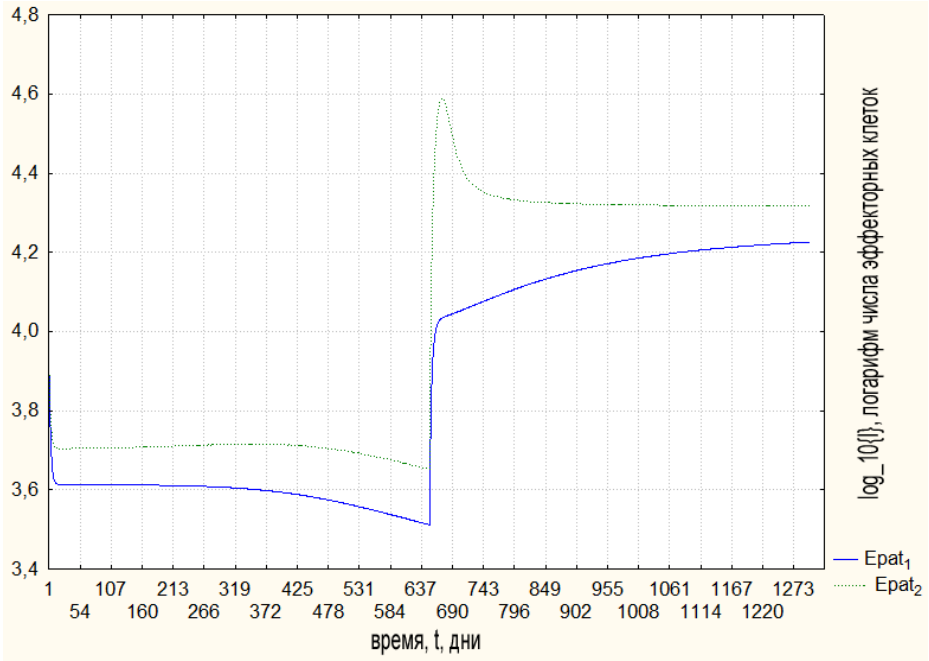


Рис. 2. Динамика численности эффекторных клеток, распознающих патогенный пептид. Время появления инфекции – 650. Во втором случае во время появления инфекции концентрации пептидов случайным образом изменялись

Литература

1. Perelson A., Weisbuch G. Immunology for physicists // Reviews of Modern Physics. 1997. V. 69. P. 1219.
2. Иванченко М.В. Конкуренция и селекция клонотипов в больших ансамблях иммунных Т-клеток // Вестник Нижегородского университета им. Н.И. Лобачевского. 2010. Т. 6. С. 38.

3. Ivanchenko M.V. Transient selection in multi-cellular immune networks // JETP Letters. 2011. V. 93. P. 35-40.
4. De Boer R., Perelson A. Competetive control of self-renewing T cell repertoire // International Immunology. 1997. V. 9. P. 779.
5. Or-Guil M. et al. Clonal expansion of cytotoxic T cell clones: the role of the immunoproteasome // Proceedings of the International Symposium on Mathematical and Computational Biology. 2005. P. 199.

ОБЛАЧНАЯ СРЕДА ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ ДЛЯ НАУЧНЫХ ВЫЧИСЛЕНИЙ И ОБРАЗОВАНИЯ

Р.И. Идрисов

Институт систем информатики им. А.П. Ершова СО РАН, Новосибирск

Доклад посвящен проекту браузерной среды программирования, предназначенной для функциональных языков программирования и использующей в качестве вычислителя рабочую станцию и удалённую систему нефиксированной структуры.

Введение

На сегодняшний день всё большую популярность набирают облачные сервисы, которые предоставляют различные возможности. Конечно, под словом «облачный» подчас скрываются обычные вещи, которые просто были названы по-новому. Согласно последней редакции российской Википедии (http://ru.wikipedia.org) на момент написания статьи, облачный сервис – это просто некоторый доступный ресурс в сети, который может быть использован без знания его внутренней структуры.

Статистика поисковых запросов в некотором смысле отражает интересы общества. Если проследить статистику google (http://www.google.com/insights/search/) по двум терминам «parallel» и «cloud», можно увидеть, что популярность «cloud» с 2004 года выросла вдвое и продолжает расти, а популярность «parallel», наоборот, неуклонно падает и с 2004 года изменилась вдвое. Рост популярности термина «cloud» обусловлен тем, что облачность представляет пользовательский интерфейс для использования какого-то ресурса. Сам термин «облачный» по одной из версий образовался из того, что Интернет изображался в виде облака, сокрытая от пользователя структура, находящаяся в Интернете, и является этим самым облаком. Концепция, конечно, была высказана уже давно, но, как и с функциональными языками программирования, которые были описаны задолго до их популярности, развитие технологий достигло уровня, дающеего возможность для массового применения облачности.

Продолжают набирать популярность облачные хостинги amazon, cloud9, которые предоставляют вычислительные ресурсы. В частности, они позволяют выполнять код на V8 (JavaScript от Mozilla), этот язык во многом совместим с браузерным JavaScript, что открывает возможность для создания переносимого кода. Одна и та же программа может быть исполнена как на персональной рабочей станции, так и на потенциально мощном облачном вычислителе. Притом здесь переносимость совсем не такая, как для Java. Программное обеспечение не требует дополнительной установки библиотек поддержки времени исполнения, а выполняется внутри браузера. Кроме того, это исполнение предполагается безопасным для пользователя. Растущее количество сервисов в сети Интернет, предоставляющих различные услуги, прямо или косвенно связанные с вычислениями, говорит о популярности такого подхода. Очень привлекательно, не имея компилятора, иметь возможность в любой момент зайти на соответствующую страницу в сети и выполнить интересующий код. Подобные сервисы варьируются от совершенно аскетичных, рассчитанных на исключительно короткие программы (http://codepad.org), до предоставляющих среду разработки с группировкой по проектам, подсветкой синтаксиса (http://c9.io – Cloud 9) и т.д.

Несмотря на большое количество таких ресурсов, всё же некоторая ниша остаётся незаполненной. В частности, нет таких, которые предоставляли бы возможность отладить и запустить научную задачу с массивным параллелизмом.

Следует отметить, что такой подход не является чем-то абсолютно новым, из-за большого количества разных параллельных систем разработчики стараются использовать переносимые варианты, которые могут быть исполнены в различных условиях, в тех случаях, когда, конечно, речь не идёт о специальном программном обеспечении для конкретного вычислителя.

Разрабатываемая система ставит перед собой две цели: научную и образовательную. Для научной цели более важна масштабируемость, а для образовательной – доступность. Кроме того, было бы неправильно ориентироваться только на один язык программирования, поскольку не существует единого мнения о наилучшем учебном языке. Для масштабируемости требуется универсальность описания параллелизма, это значит, что программа не должна быть адаптирована для структуры конкретной вычислительной системы. Согласно работам А.П. Ершова, это достигается, если язык программирования приближается к языку описания задач, а не к языку описания алгоритмов. В Институте систем информатики СО РАН мы продолжаем разработку потокового языка программирования Sisal [1, 2], эта работа ставит перед собой именно такие цели.

Литература

1. McGraw, J. R. et. al. Sisal: Streams and iterations in a single assignment language, Language Reference Manual Version 1.2 // Lawrence Livermore Nat. Lab. Manual M-146 (Rev.1). – Livermore, CA 1985.
2. Касьянов В.Н., Бирюкова Ю.В., Евстигнеев В.А. Функциональный язык SISAL 3.0 // Поддержка супервычислений и Интернет-ориентированные технологии. – Новосибирск, 2001. – С. 54-67.

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ АНАЛИЗА ЭНЕРГОЭФФЕКТИВНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.В. Калачев, А.С. Карсаков, И.Б. Мееров, Я.А. Напыльникова, А.Ю. Овсюхно

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время тема энергосбережения является достаточно актуальной. В связи с повсеместным применением информационных технологий энергетическая эффективность стала приоритетным направлением и в IT-сфере. Так, увеличение производительности процессоров привело к нелинейному росту энергопотребления в рабочих станциях и серверах. Производители мобильных устройств столкнулись с существенным падением времени автономной работы при расширении функциональности. Таким образом, необходимость внедрения энергосберегающих технологий продиктована не только желанием сэкономить ресурсы, но и невозможностью обеспечить приемлемое время автономной работы.

Полностью справиться с проблемой, используя только аппаратные решения (увеличение емкости батареи, оптимизация устройства процессора и др.), не представляется возможным, поэтому необходимо использовать и программные решения.

В работе предлагается новый программный инструмент для сбора и анализа данных об энергопотреблении в программах. Описывается подход к организации инструмента, приводятся примеры его использования на модельных задачах.

Текущее состояние предметной области

Метрики. Вопрос о введении согласованной системы метрик для оценки энергопотребления активно изучается в литературе. Для оценки энергопотребления могут использоваться классические физические характеристики: ватты, ватт*часы и джоули. В работе [1] предлагается ввести новую физическую величину MIPJ (millions-of-instructions-per-joule) миллион–операций–на–джоуль. В некоторых других источниках эту величину называют Gflop/s per Watt, что с точки зрения физики является эквивалентным. Чем больше эта величина, тем более энергоэффективной является вычислительная система. Долгое время эта метрика использовалась для сравнения энергоэффективности вычислительной техники. Аналогично Top-500 самых высокопроизводительных систем в мире (http://www.top500.org/) был создан список наиболее энергоэффективных систем – Green 500 (www.green500.org). В 2010 году была предложена альтернативная метрика [2] – FTTSE (*f(time to solution) * energy*), одновременно учитывающая время работы программы (*f(time to solution)*) и потраченную энергию (*energy*). Вопрос формирования и применения системы метрик для оценки энергопотребления представляет большой практический интерес.

Оптимизация энергопотребления. Оптимизация энергопотребления в мобильных устройствах может выполняться как на аппаратном, так и на программном уровне. Развитие вопроса протекало в двух основных направлениях: Dynamic Power Management (DPM) [3–6] и Dynamic Voltage Scaling (DVS) [6–8]. Суть первого направления заключается в том, чтобы операции, не требующие всей вычислительной мощи центрального процессора, выполнялись в более энергоэффективных состояниях центрального про-

цессора. Центральный процессор может переходить в состояния, при которых производительность понижена, что в свою очередь уменьшает энергопотребление. Основная сложность заключается в том, чтобы правильно предсказывать момент перехода в режим экономии энергии. Направление Dynamic Voltage Scalling предлагает другой подход, суть которого заключается в динамическом изменении питания, подаваемого на разные компоненты аппаратного обеспечения (такие как центральный процессор, оперативная память и др.) в зависимости от потребностей исполняемого в данный момент программного обеспечения.

В настоящее время указанные подходы достаточно хорошо изучены и активно развиваются с появлением новой вычислительной техники. Вместе с тем, все больший интерес вызывает оптимизация на уровне прикладных программ – написание энергоэффективного программного обеспечения [5, 9–13]. Выделены типовые подходы к оптимизации:

1. преобразование циклов (loop transformations);
2. преобразование структур данных (data structure transformations);
3. межпроцедурная оптимизация (inter-procedural transformations);
4. преобразование управляющих конструкций (operators and control structure transformations).

Оценка энергоэффективности программного обеспечения

Целью данной работы является разработка методов и программных средств анализа и оптимизации энергопотребления вычислительных устройств. Для ее достижения необходимо определить набор величин, основываясь на которых можно будет оценивать энергоэффективность. Нами были выбраны следующие метрики:

1. текущая потребляемая мощность (Вт);
2. процент времени, в течение которого процессор находится в том или ином C-state-состоянии;
3. количество переходов между разными C-state (переходов/с);
4. процент времени загруженности процессора;
5. процент времени бездействия (простоя) процессора.

Существует множество инструментов, которые позволяют определять данные метрики. Сравнительный анализ таких инструментов представлен в таблице.

	Intel(R) Power Checker	**Per fmon**	**Intel(R) Power Informer**	**Powertop**	**Joulemeter**
ОС	Windows	Windows	Windows	Linux	Windows
Потребляемая мощность	Да	Да	Да	Да	Да
Анализ C-State	Да	Да	Да	Да	Нет
Информация о центральном процессоре	Нет	Да	Да	Да	Нет
Отображение результатов в реальном времени	Нет	Да	Нет	Да	Да
Проведение серий экспериментов	Нет	Нет	Нет	Нет	Нет
Обновление данных (сек.)	1.0	1.0	неизвестно	15.0	1.0
Вывод результатов в файл	Да	Да	Да	Нет	Нет
Хранение результатов	Нет	Нет	Нет	Нет	Нет

Все инструменты обладают теми или иными недостатками. Более того, ни один из них не предоставляет разработчику полнофункциональный исследовательский комплекс, который был бы удобен при оптимизации программных продуктов. В связи с

этим возникла необходимость разработать систему, которая могла бы решить данную проблему. Были предъявлены следующие требования к системе:

Требование	Описание
Кроссплатформенность	Поддержка ОС: Windows XP, Windows 7 и Linux. Предусмотреть возможность портирования на другие архитектуры процессоров (ARM) и мобильные ОС (Android, iOS, Windows Phone).
Нагрузка	Система должна добавлять минимальную погрешность в проводимые эксперименты (не более 5%).
Автоматизация запуска экспериментов	Ручной эксперимент: приложение должно работать в интерактивном режиме с настройкой параметров эксперимента. Серийный эксперимент: приложение должно работать в автоматизированном режиме, используя конфигурационный файл. Система должна иметь открытую архитектуру и поддерживать расширение списка используемых параметров. Система должна поддерживать функцию калибровки. Основываясь на полученных параметрах, система должна следить за корректностью условий проведения экспериментов.
Формат хранения данных	Формат хранения должен быть расширяемым, самодостаточным, ориентированным на хранение параметров и результатов экспериментов, поддерживать быстрые операции чтения/записи в файл.
Источник данных	Исследовательский комплекс должен иметь возможность получать информацию об энергопотреблении из различных источников (измерительный прибор, API ОС, драйвер батареи и др.) Каждый источник данных должен быть взаимозаменяемым и не влиять на другую функциональность системы.
Пользовательский интерфейс	Взаимодействие с пользователем должно происходить с использованием графического пользовательского интерфейса. Должна поддерживаться следующая функциональность: задача параметров эксперимента (графическая форма или выбор конфигурационного файла на файловой системе), манипуляция с очередью экспериментов, запуск экспериментов, сбор результатов, представление результатов экспериментов (графический, конвертация в текстовый формат).
Представление данных	Пользователь должен иметь возможность загрузить результаты любого проведенного эксперимента. Результаты должны быть представлены в удобном и наглядном виде для пользователя с использованием графиков, таблиц, гистограмм и других вариантов представления данных.

Архитектура системы

Для минимизации накладных расходов при сборе метрик для разработки инструмента был использован язык программирования C++. Для обеспечения кроссплатформенности реализация пользовательского интерфейса была выполнена с помощью QT framework. Использование объектного подхода позволило предусмотреть возможность расширения системы как с точки зрения поддержки новых метрик, так и с точки зрения подключения новых измерителей (в текущей версии поддерживается Perfmon).

Высокоуровневая архитектура системы выглядит следующим образом (рис. 1).

Центральным управляющим модулем системы является *Controller*. Данная компонента предоставляет API, посредством которого другие компоненты системы могут взаимодействовать с внутренней машиной состояний и очередью экспериментов. Также, данная компонента имеет набор функций, с помощью которых другие компоненты могут получить доступ к модулю *Data Representation*, который отвечает за программное представление данных об эксперименте. Модуль *Experiment Executer* реализует всю логику, связанную с проведением экспериментов и сбором метрик. Для получения необработанных данных используются различные источники данных, реализации которых представлены в модуле *Data Source*. Все инструменты визуализации, а также компонента предварительной обработки данных находятся в модуле *Visualization*.

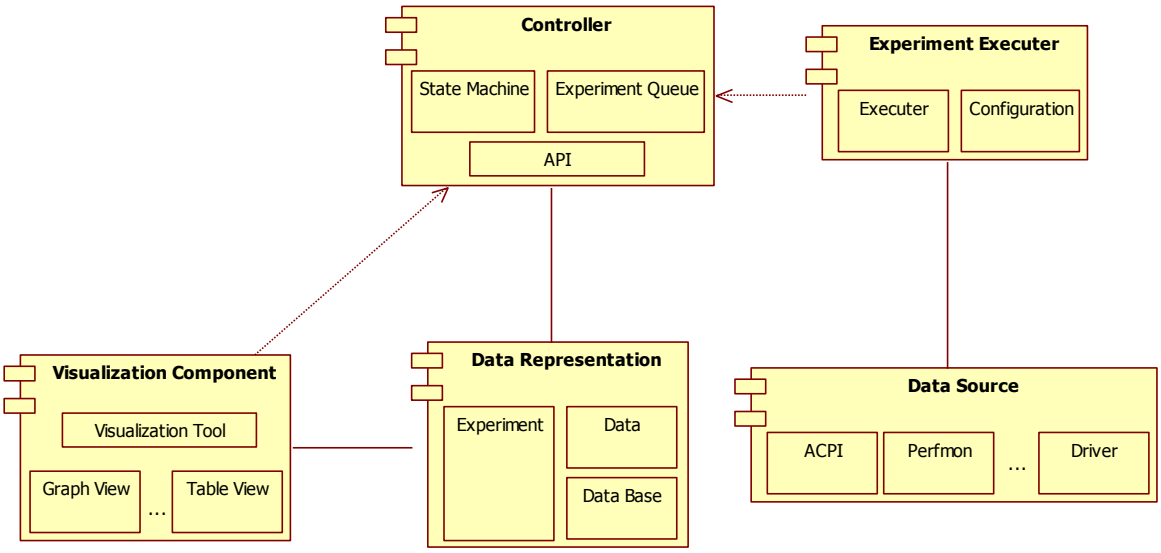


Рис. 1. Архитектура системы

Результаты проведенных экспериментов

Тестовая система

- Processor: Mobile QuadCore Intel Core i7 2630QM @ 2GHz (2.9 GHz Turbo Boost).
- Cache: L1 – 32 Kb, L2 – 256 Kb, L3 – 6Mb.
- RAM: 4Gb, DDR3 1333 MHz.
- Operating System: Windows 7 Ultimate (x64).

В первой серии экспериментов (рис. 2-4) производится сравнение потребляемой мощности при решении задачи матричного умножения в следующих конфигурациях: последовательная версия, версии с использованием SSE, OpenMP и Cilk+.

Заметим, что пиковое значение потребляемой мощности при работе параллельных версий больше, чем у последовательной версии. OpenMP и Cilk+ задействуют большие ресурсы, поэтому время работы приложений значительно меньше, что приводит к лучшим показателям энергопотребления, чем у последовательной версии. Реализация на SSE использует дополнительные возможности процессора, что позволяет решить задачу наиболее энергоэффективно.

На гистограммах видны следующие результаты: с ростом размеров матрицы увеличивается и разница в количестве энергии, потребляемой последовательной и параллельными версиями и версии с использованием SSE. Сравнивая версии на OpenMP и Cilk+ между собой, можно выделить преимущество второй версии при достаточно больших размерах задачи.

Вторая серия экспериментов (рис. 5) иллюстрирует влияние количества переходов между C-state-состояниями процессора на энергопотребление. Рассмотрим задачу, состоящую из нескольких независимых частей, которые можно запускать либо непрерывно друг за другом, либо с некоторыми паузами, что приводит к увеличению числа C-State-переходов.

В результате видим, что увеличение числа переходов между C-state-состояниями процессора приводит к росту энергопотребления.

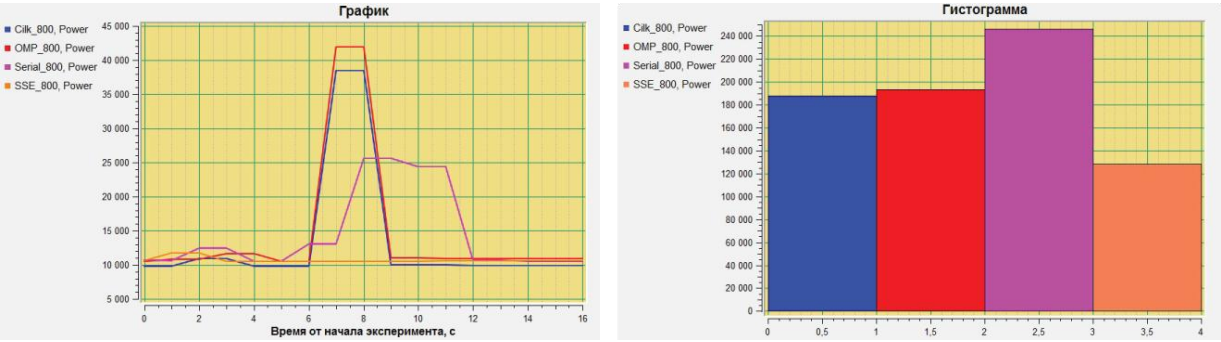


Рис. 2. Матричное умножение 600x600

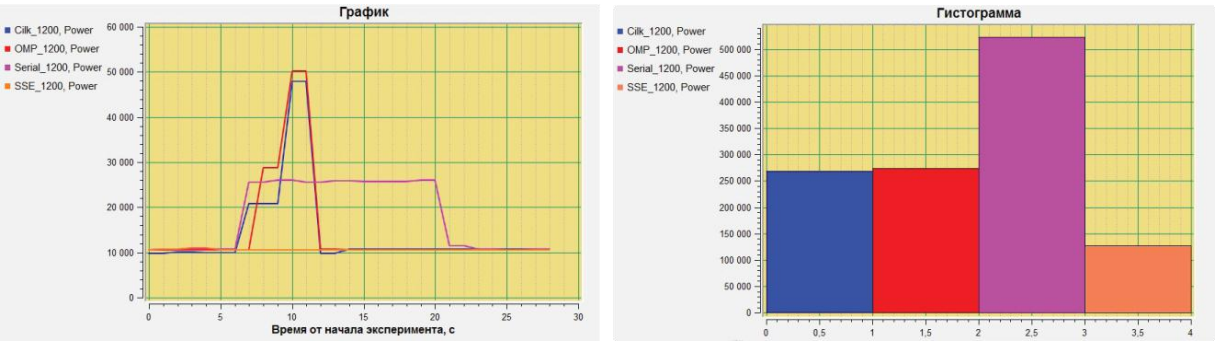


Рис. 3. Матричное умножение 1200x1200

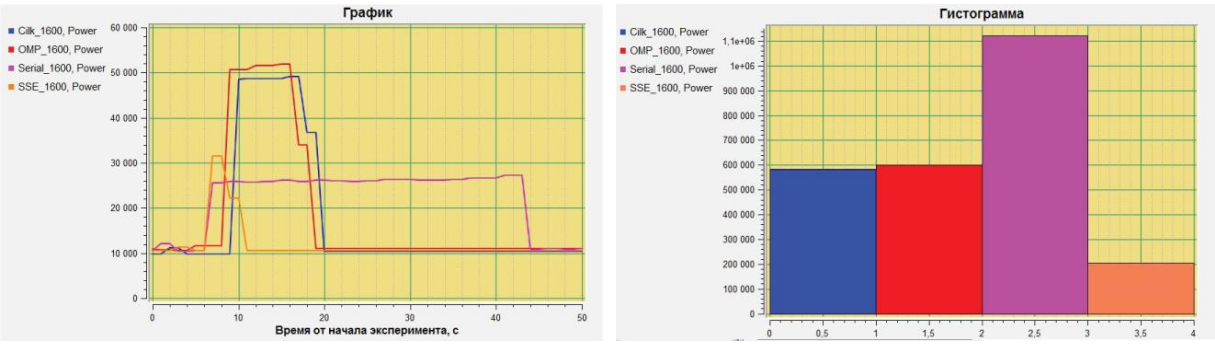


Рис. 4. Матричное умножение 1800x1800

Рис. 5. Матричное умножение. C-states

Заключение

В ходе работы представлен инструмент, предназначенный для помощи разработчику в оптимизации энергоэффективности программного обеспечения. Инструмент предоставляет широкий спектр возможностей для запуска экспериментов, сбора необходимых данных, хранения, обработки и визуализации результатов.

Работоспособность системы была продемонстрирована на серии из двух демонстрационных задач:

- Сравнение энергопотребления при решении задачи матричного умножения.
- Анализ зависимости энергопотребления от количества переходов между C-State-состояниями.

Система является расширяемой (новые метрики, источники данных, способы визуализации и др.). Расширение функциональности системы, а также апробация системы (включая выработку подходов к оптимизации энергопотребления) является темой дальнейшей исследований.

Работа выполнена в лаборатории Intel-ННГУ «Информационные технологии» (ITLab).

Литература

1. Weiser M., Welch B., Demers A., Shenker S., Scheduling for Reduced CPU Energy // Proceedings of Symposium on Operating Systems Design and Implementation, 1994.
2. Bekas C., Curioni A. A new energy aware performance metric // Computer Science – R&D. 2010. P. 187-195.
3. Benini L., De Micheli G. Dynamic power management: design techniques and CAD tools // Kluwer Academic Publisher. 1997.
4. Šimunic T., Benini L., Acquaviva A., Glynn P., De Micheli G. Dynamic Voltage Scaling and Power Management for Portable Systems // DAC 2001, Las Vegas, USA. 2001.
5. Šimunic T. Energy efficient system design and utilization // Doctoral Dissertation, Energy efficient system design and utilization. Stanford University Stanford, CA, USA, 2001.
6. Snowdon D., Ruocco S., Heiser G., Power Management and Dynamic Voltage Scaling: Myths and Facts // National ICT Australia and School of Computer Science and Engineering University of NSW. Sydney 2052, Australia, 2005.
7. Kappiah N., Freeh V.W., Lowenthal D.K. Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs // IEEE Computer Society. 2005.
8. Chen G., Malkowski K., Kandemir M.T., Raghavan P. Reducing power with performance constraints for parallel sparse applications // IPDPS. IEEE Computer Society, 2005.
9. Tiwari V., Malik S., Wolfe A. Power analysis of embedded software: A first step // IEEE Transactions on VLSI Systems, 1994.
10. Tiwari V., Malik S., Wolfe A., Lee M. Instruction level power analysis and optimization of softwar // Journal of VLSI Signal Processing Systems. 1996. P. 1-18.
11. Tomyiama H.H., Ishihara T., Inoue A., Yasuura H. Instruction scheduling for power reduction in processor-based system design // DATE, 1998.
12. Simunic T., Benini L., De Micheli G. Energy-Efficient Design of Battery-Powered Embedded Systems // International Symposium on Low Power Electronics and Design, 1999.
13. Brandoles C., Fornaciari W., Salice F., Sciuto D. The impact of source code transformations on soft-ware power and energy consumption // Journal of Circuits, Systems, and Computers. 2002. Vol. 11, No. 5. P. 477-502.

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ ПРИ ИССЛЕДОВАНИИ СВОДИМОСТИ ТРЕХ- И ЧЕТЫРЕХИНДЕКСНЫХ ЗАДАЧ

А.С. Катеров

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматриваются многоиндексные задачи линейного программирования транспортного типа. Один из подходов к решению таких задач – сведение их к задаче поиска потока в сети. В работе доказано, что (в рамках рассматриваемой концепции сводимости) условие 2-вложенности является необходимым и достаточным условием сводимости в трёх- и четырёхиндексном случае. Доказательство основано на переборной схеме, выполненной средствами высокопроизводительных вычислений.

Введение

Рассматриваются многоиндексные задачи линейного программирования транспортного типа. Задачи данного класса возникают при решении различных прикладных задач: распределения мощностей каналов передачи данных провайдерами сети Интернет, объемно-календарного планирования, транспортной задачи с промежуточными пунктами и т.д. [1]. Один из подходов к решению многоиндексных задач линейного программирования основан на исследовании возможности сведения задач к потоковым алгоритмам. В качестве потоковой задачи рассматривается задача поиска циркуляции минимальной стоимости [2]. В данной работе проводится исследование сводимости трёх- и четырёхиндексных задач. Разработана параллельная программная система, предназначенная для запуска на многопроцессорных вычислительных системах с распределённой памятью, позволяющая доказывать невозможность такого сведения для некоторых частных подклассов.

Многоиндексные транспортные задачи

При постановке многоиндексных задач линейного программирования транспортного типа воспользуемся формализацией, предложенной в [3]. Пусть $s \in N$ и $N(s) = \{1,2,...,s\}$. Каждому числу l поставим в соответствие параметр j_l, называемый индексом, который принимает значения из множества $J_l = \{1,2,...,n_l\}$, где $n_l \geq 2$, $l \in N(s)$. Пусть $f = \{k_1, k_2,..., k_t\} \subseteq N(s)$; $k_i < k_{i+1}$, $i = \overline{1, t-1}$. Набор значений индексов $F_f = (j_{k_1}, j_{k_2},..., j_{k_t})_f$ будем называть *t*-индексом. Множество всех *t*-индексов, соответствующих f, определим как $E_f = \{F_f \mid F \in J_{k_1} \times J_{k_2} \times ... \times J_{k_t}\}$. Там, где это не вызывает неоднозначности, будем опускать нижний индекс f в записи *t*-индексов. Если $F_{f'} \in E_{f'}$, $F_{f''} \in E_{f''}$, где $f', f'' \subseteq N(s)$ и $f' \cap f'' = \varnothing$, то через $F_{f'}F_{f''}$ обозначим такой набор, что $F_{f'}F_{f''} \in E_{f' \cup f''}$. Далее определим $\overline{f} = N(s) \setminus f$, тогда согласно введенному обозначению $F_{N(s)} = F_f F_{\overline{f}}$.

Каждому набору F_f поставим в соответствие действительное число z_{F_f}, $F_f \in E_f$. Данное отображение множества *t*-индексов E_f в множество действительных чисел

назовем (как и в [3]) t-индексной матрицей и обозначим через $\{z_{F_f}\}$. Рассмотрим s-индексную матрицу $\{z_{N(s)}\}$ и введем следующее обозначение:

$$\sum_{F_f \in E_f} z_{F_f F_{\overline{f}}} = \sum_{j_{k_1} \in J_{k_1}} \sum_{j_{k_2} \in J_{k_2}} \dots \sum_{j_{k_t} \in J_{k_t}} z_{F_f F_{\overline{f}}}, \; F_{\overline{f}} \in E_{\overline{f}}.$$

Введенное обозначение подсумм s-индексной матрицы будем использовать при формализации многоиндексных транспортных задач.

Пусть M – заданное множество, $M \subseteq 2^{N(s)}$; $\{a_{F_{\overline{f}}}\}$, $\{b_{F_{\overline{f}}}\}$ – заданные $|\overline{f}|$-индексные матрицы свободных коэффициентов, $0 \le a_{F_{\overline{f}}} \le b_{F_{\overline{f}}}$, $F_{\overline{f}} \in E_{\overline{f}}$, $f \in M$; $\{c_{F_{N(s)}}\}$ – заданная s-индексная матрица коэффициентов целевой функции; $\{x_{F_{N(s)}}\}$ – s-индексная матрица неизвестных. Тогда многоиндексная задача линейного программирования транспортного типа формализуется следующим образом:

$$a_{F_{\overline{f}}} \le \sum_{F_f \in E_f} x_{F_f F_{\overline{f}}} \le b_{F_{\overline{f}}}, F_{\overline{f}} \in E_{\overline{f}}, f \in M, \tag{1}$$

$$x_{F_{N(s)}} \ge 0, \; F_{N(s)} \in E_{N(s)}, \tag{2}$$

$$\sum_{F_{N(s)} \in E_{N(s)}} c_{F_{N(s)}} x_{F_{N(s)}} \to \min. \tag{3}$$

Далее класс всех многоиндексных задач вида (1)–(3) при фиксированном множестве M будем обозначать $W(M)$. Матрицу системы ограничений (1) обозначим через $Matr(M, n_1, ..., n_s)$, так как она однозначно определяется множеством M, а также количеством принимаемых значений для каждого индекса: $n_1, n_2, ..., n_s$.

В общем случае для решения таких задач могут быть использованы лишь универсальные методы решения задач линейного программирования [4]. Однако специфика поставленной задачи (линейные ограничения транспортного типа) позволила для частного класса рассматриваемых задач предложить более эффективные алгоритмы их решения, отличные от общих (достаточно трудоёмких) методов решения.

Исследование сводимости

В данной работе исследуется возможность сведения класса многоиндексных задач линейного программирования транспортного типа к классу задач поиска потока минимальной стоимости, который определяется следующим образом. Рассмотрим ориентированный граф $G = (V_G, A_G)$, $A_G \subseteq V_G^{\,2}$, здесь V_G и A_G – множество вершин и дуг графа G соответственно. Пусть l_{ij}, u_{ij} – пропускные способности дуги (i, j); e_{ij} – стоимость дуги (i, j); x_{ij} – неизвестная величина потока вдоль дуги (i, j), $(i, j) \in A_G$. Тогда через $v(G; l_{ij}, u_{ij}, e_{ij}, (i, j) \in A_G)$ обозначим следующую задачу поиска потока минимальной стоимости:

$$\sum_{(i,j) \in A_G} x_{ij} - \sum_{(j,i) \in A_G} x_{ji} = 0, \; i \in V_G,$$

$$l_{ij} \le x_{ij} \le u_{ij}, \; (i, j) \in A_G,$$

$$x_{ij} \ge 0, \; (i, j) \in A_G,$$

$$\sum_{(i,j) \in A_G} e_{ij} x_{ij} \to \min.$$

Обозначим через $Graph$ множество всех ориентированных графов. Класс задач поиска потока минимальной стоимости определим как

$$W_{Graph} = \{v(G, l_{ij}, u_{ij}, e_{ij}, (i,j) \in A_G) \mid l_{ij}, u_{ij} \in Z_+, e_{ij} \in Z, (i,j) \in A_G, G \in Graph\}.$$

Применяется концепция $t_1 \mid t_2 - equal \mid t_3 - edge$ сводимости, предложенная в [5].

Определение 1. Множество *M* называется *k*-вложенным, если существует разбиение множества *M* на *k* подмножеств $M_i = \{f_1^{(i)}, f_2^{(i)}, ..., f_{m_i}^{(i)}\}$, $i = \overline{1,k}$, что $f_j^{(i)} \subset f_{j+1}^{(i)},\ j = \overline{1, m_i - 1},\ i = \overline{1,k}$.

Теорема 1 [1]. Пусть $M \subseteq 2^{N(s)}$. Для того чтобы класс $W(M)$ являлся $L \mid L - equal \mid L - edge$ сводимым к классу W_{Graph}, достаточно, чтобы множество *M* было 2-вложенным.

Теорема 2 [6]. Пусть $M \subseteq 2^{N(s)}$. Для того чтобы класс $W(M)$ являлся $t_1 \mid t_2 - equal \mid t_3 - edge$ сводимым к классу W_{Graph}, необходимо, чтобы $Matr(M, n_1, ..., n_s)$ для любых $n_1, ..., n_s$ была абсолютно унимодулярна.

Теорема 3 [7]. Пусть $M \subseteq 2^{N(s)}$, $s \le 4$. Для того чтобы класс $W(M)$ являлся $t_1 \mid t_2 - equal \mid t_3 - edge$ сводимым к классу W_{Graph}, где $t_1(n), t_2(n), t_3(n) \ge n$, $n \in N$, необходимо и достаточно, чтобы множество *M* было 2-вложенным.

В рамках проведённых исследований была разработана параллельная программная система, которая по заданным множеству M, а также $n_1, ..., n_s$ строит $Matr(M, n_1, ..., n_s)$. Далее проверяется абсолютная унимодулярность построенной матрицы, таким образом, определятся выполнение необходимого условия сводимости теоремы 2. Проверка абсолютной унимодулярности в рассматриваемом случае основана на переборе всех миноров матрицы размером $n \times m$ с рангом больше 2, при этом число таких миноров равно $\sum_{i=3}^{\min(n,m)} C_n^i C_m^i$. Максимальный размер исследуемых матриц достигал 27×27, для них количество миноров, которые требовалось перебрать, равно 1.9×10^{15}. В связи с таким объёмом вычислений было принято решение о применении параллельности в разрабатываемой системе. Данная программная система запускалась на супер-ЭВМ МП-20 (РФЯЦ-ВНИИЭФ г. Саров).

Заключение

Исследован класс трёх- и четырёхиндексных задач линейного программирования транспортного типа. Рассмотрены вопросы сводимости многоиндексных задач к классу задач поиска потока в сети. В рамках исследуемой схемы сведения показано, что для класса многоиндексных задач с числом индексов, не превосходящим 4, условие 2-вложенности множества, определяющего систему ограничений, является одновременно необходимым и достаточным условием сводимости к классу задач поиска потока минимальной стоимости. Применение разработанной программной системы позволило отыскать множества *M*, для которых $Matr(M, n_1, ..., n_s)$ не является абсолютно унимодулярной, что и было использовано при доказательстве.

Работа выполнена при частичной финансовой поддержке Минобрнауки России, государственное соглашение о представлении гранта №14.B37.21.0878.

Литература

1. Афраймович Л.Г., Прилуцкий М.Х. Многоиндексные задачи распределения ресурсов в иерархических системах // Автоматика и телемеханика. 2006. № 6. С. 194-205.
2. Форд Л. , Фалкерсон Д. Потоки в сетях. М.: Мир, 1966.

3. Раскин Л.Г., Кириченко И.О. Многоиндексные задачи линейного программирования. М.: Радио и связь, 1982.
4. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985.
5. Афраймович Л.Г. Трехиндексные задачи линейного программирования с вложенной структурой // Автоматики и телемеханика. 2011. № 8. С. 109–120.
6. Афраймович Л.Г., Прилуцкий М.Х. Многопродуктовые потоки в древовидных сетях // Известия РАН. Теория и системы управления. 2008. № 2. С. 57-63.
7. Афраймович Л.Г., Катеров А.С. Трех- и четырехиндексные задачи с вложенной структурой // Математическое моделирование. Оптимальное управление. Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 3 (1). С. 163-169.

АНАЛИЗ МАСШТАБИРУЕМОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РЕШЕНИЯ КООПЕРАТИВНОЙ ИГРЫ

А.С. Кириллов

Оренбургский госуниверситет

В работе рассматриваются подходы к решению вычислительно-сложной задачи построения оптимальной коалиции и распределение выигрыша в кооперативной игре, заданной множеством биматричных игр для *n* игроков, с использованием параллельных вычислительных технологий.

Введение

Алгоритмы решения большинства игр имеют экспоненциальную временную сложность, для кооперативных игр необходимо осуществлять комбинаторный перебор всех возможных коалиций, а также применять симплекс-метод для решения серии задач линейного программирования [4, 6–9].

Понятно, что решение задач кооперативных игр – процесс трудоемкий и с увеличением числа игроков трудоемкость решения задачи только повышается [5].

Использование параллельных реализаций алгоритмов решения различных игр позволит, во-первых, увеличить размеры задачи (количество игроков, число их стратегий и т.д.), решаемой за то же время, и, во-вторых, значительно сократить время получения результатов, что обеспечивает большую интерактивность системе поддержки принятия решений при взаимодействии с пользователем, а также позволяет использовать решающий модуль в системах реального времени с критичным временем отклика, например, в планировщике задач для грид-системы [1].

В предыдущих работах [6, 7] были рассмотрены основные понятия теории кооперативных игр и способы их решения, такие, как алгоритм решения кооперативной игры, заданной биматричными играми.

Для оценки эффективности разработанных параллельных алгоритмов обычно рассматривают масштабируемость параллельного алгоритма на вычислительном кластере. При этом параллельный алгоритм называют масштабируемым, если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров.

1. Параллельный алгоритм решения кооперативной игры, заданной биматричными играми с использованием итеративного алгоритма Брауна-Робинсона

На основании вышесказанного нами разработаны параллельные алгоритмы решения кооперативной игры, заданной набором биматричных игр с использованием технологии MPI [6, 7, 9]. Алгоритмы основаны на решении задачи линейного программирования (симплекс метода) и на итеративном алгоритме Брауна–Робинсона.

Параллельный алгоритм поиска оптимальных стратегий с использованием линейного программирования и симплекс метода [6] позволяет получить точные решения стратегических игр как подзадач решения кооперативной игры. Но при этом для стратегических игр большого размера не всегда можно получить решения на ЭВМ, ввиду проблем нехватки памяти для размещения решаемой игры и накоплении ошибок.

Для преодоления этих проблем возможно использование итерационных методов решения матричных игр, в частности, мы рассмотрели метод Брауна–Робинсона [3, 1] и разработали параллельный алгоритм решения кооперативной игры, заданной набором биматричных игр, итерационным методом Брауна-Робинсона [10] с использованием технологии MPI.

Схема работы параллельного алгоритма представлена на рис. 1.

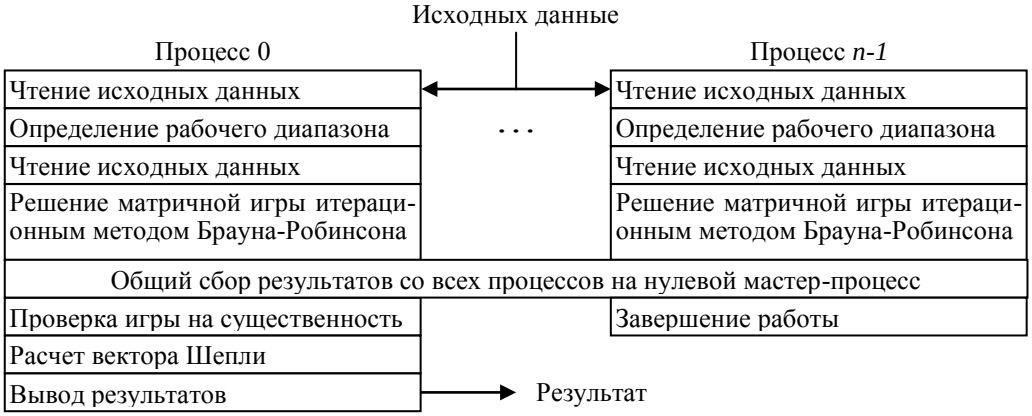


Рис. 1. Схема работы параллельного алгоритма решения кооперативной игры с использованием итерационного метода Брауна-Робинсона

2. Разработка и проведение вычислительного эксперимента

Тестирование разработанного параллельного алгоритма производилось на вычислительном кластере Оренбургского государственного университета.

Для запуска тестовых задач использовались четыре двухпроцессорных вычислительных узла на базе процессоров Intel Xeon 5440, на каждом узле использовались только два ядра. Такая методика тестирования была связана с тем, что решаемая задача кооперативных игр очень требовательна к объему оперативной памяти, и для каждого экземпляра задачи выделялось до 6 Гб памяти.

Тестирование разработанного алгоритма производилось последовательно для 4, 5, 6, 7 и 8 игроков, при этом каждый игрок имел 4, 5, 6, 7 и 8 стратегий. Для каждой задачи формировалось пять произвольных наборов исходных данных. Полученные наборы исходных данных затем решались последовательно на 1, 2, 4 и 8 процессорах, при этом замерялось время выполнения каждой задачи. Затем для каждой тройки значений количества игроков, количества стратегий и количества процессоров вычислялось среднее время работы параллельной программы на всех пяти различных наборах исходных данных. На основании полученных значений времени выполнения были построены графики времени работы и эффективности распараллеливания программ для 5, 6, 7, 8 игроков с 4, 5, 6, 7 и 8 стратегиями для варианта с использованием симплекс метода и итерационного метода Брауна–Робинсона. Графики представлены на рис. 2 – 7 соответственно.

3. Анализ результатов вычислительного эксперимента

На рис. 2 и 3 представлены графики, из которых видно, что при выполнении точного алгоритма решения кооперативной игры с использованием симплекс метода не достигается приемлемая эффективность работы параллельной программы, и также не достигается приемлемая масштабируемость алгоритма. Это прежде всего связано с тем, что решаемые подзадачи в виде матричных игр имеют различные количества стратегий для коалиции, т.е. имеют различные количества строк в матрицах, что сильно меняет

общий размер решаемой симплекс задачи, и, как следствие, появляется дисбаланс распределения подзадач между вычислительными узлами.

Кроме того, для семи игроков с количеством стратегий больше 5 не удалось получить хоть какого-либо решения, т.к. решаемая задача не уложилась в размер выделенной оперативной памяти вычислительного кластера. Аналогично обстоят дела и для игры с количеством игроков 8 и более при любом количестве стратегий. Это обстоятельство сильно ограничивает возможность практического использования данного алгоритма.

Однако в среднем эффективность разработанного параллельного алгоритма с использованием симплекс метода (отношение времени работы последовательного алгоритма к произведению времени работы параллельного алгоритма и количества процессов) составляет величину порядка 40%.

Обратная ситуация наблюдается при реализации параллельного алгоритма для метода Брауна-Робинсона. На графиках (рис. 4–7) видно, что с увеличением числа игроков и числа стратегий у них эффективность работы параллельного алгоритма решения кооперативной игры с использованием итерационного метода Брауна–Робинсона стремится к 100%.

Если сравнить графики на рисунках 2 и 4, то видно, что время выполнения параллельного алгоритма с использованием итерационного метода на несколько порядков меньше времени выполнения параллельного алгоритма с использованием симплекс метода на одних и тех же исходных данных.

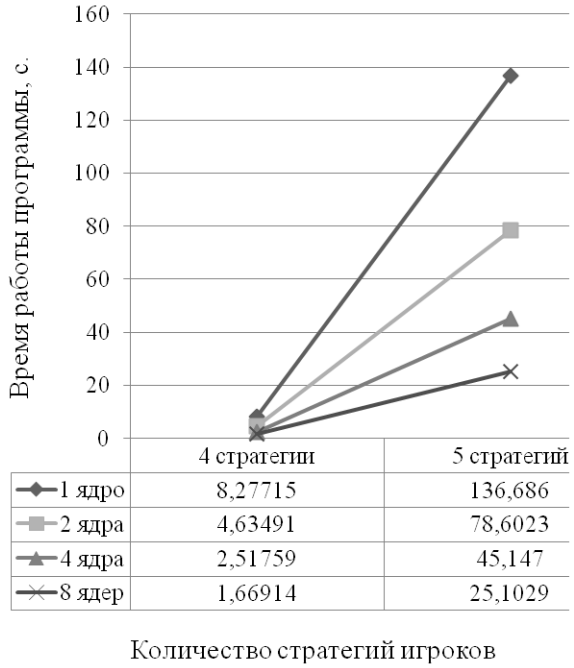


Рис. 2. График зависимости времени работы параллельной программы от количества процессоров и количества стратегий для семи игроков с использованием симплекс метода

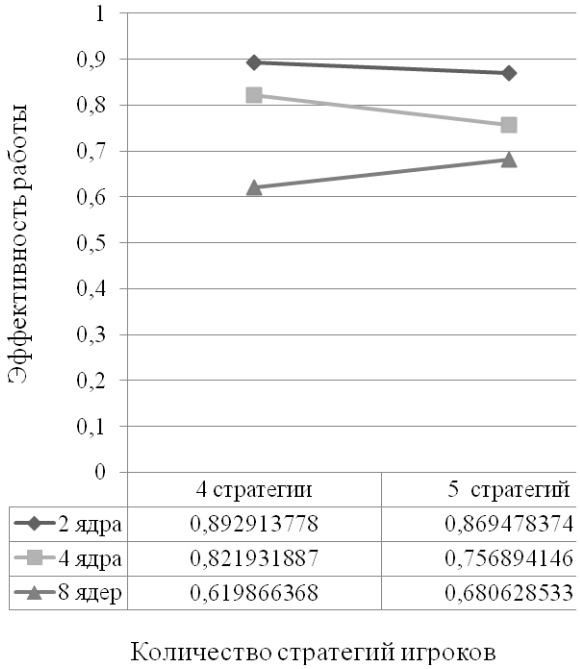
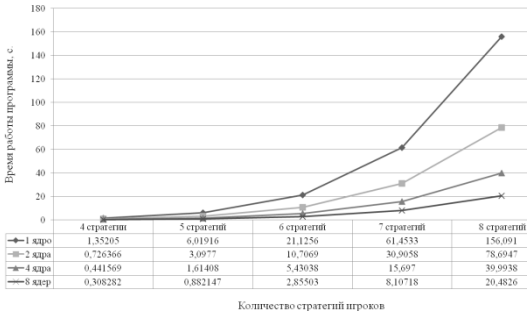
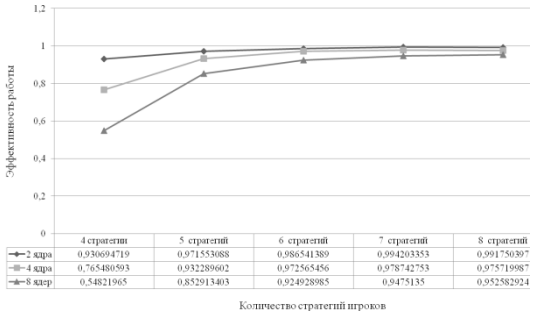


Рис. 3. График эффективности работы параллельной программы в зависимости от количества процессоров и количества стратегий для семи игроков с использованием симплекс метода



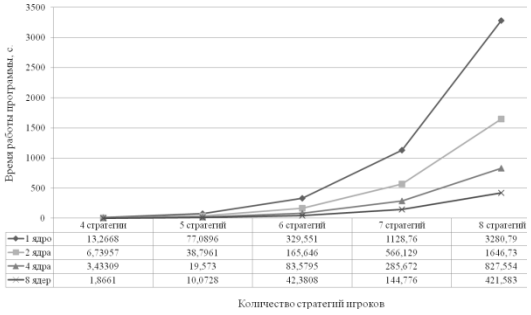
	4 стратегии	5 стратегий	6 стратегий	7 стратегий	8 стратегий
1 ядро	1,35205	6,01916	21,1256	61,4533	156,091
2 ядра	0,726366	3,0977	10,7069	30,9058	78,6947
4 ядра	0,441569	1,61408	5,43038	15,697	39,9938
8 ядер	0,308282	0,882147	2,85503	8,10718	20,4826

Рис. 4. График зависимости времени работы параллельной программы от количества процессоров и количества стратегий для семи игроков с использованием итерационного метода Брауна–Робинсона



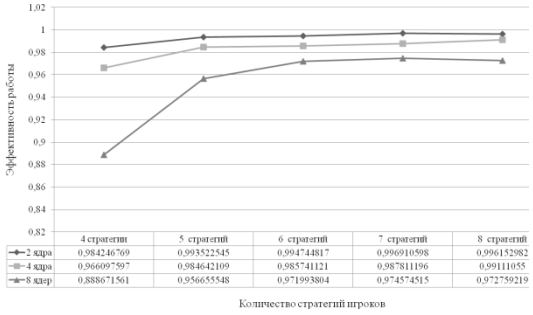
	4 стратегии	5 стратегий	6 стратегий	7 стратегий	8 стратегий
2 ядра	0,930694719	0,971553088	0,986541389	0,994203353	0,991750397
4 ядра	0,765480593	0,932289602	0,972565456	0,978742753	0,975719987
8 ядер	0,54821965	0,852913403	0,924928985	0,9475135	0,952582924

Рис. 5. График эффективности работы параллельной программы в зависимости от количества процессоров и количества стратегий для семи игроков с использованием итерационного метода Брауна–Робинсона



	4 стратегии	5 стратегий	6 стратегий	7 стратегий	8 стратегий
1 ядро	13,2668	77,0896	329,551	1128,76	3280,79
2 ядра	6,73957	38,7961	165,646	566,129	1646,73
4 ядра	3,43309	19,573	83,5795	285,672	827,554
8 ядер	1,8661	10,0728	42,3808	144,776	421,583

Рис. 6. График зависимости времени работы параллельной программы от количества процессоров и количества стратегий для восьми игроков с использованием итерационного метода Брауна–Робинсона



	4 стратегии	5 стратегий	6 стратегий	7 стратегий	8 стратегий
2 ядра	0,984246769	0,993522545	0,994744817	0,996910598	0,996152982
4 ядра	0,966097597	0,984642109	0,985741121	0,987811196	0,99111055
8 ядер	0,888671561	0,956655548	0,971993804	0,974574515	0,972759219

Рис. 7. График эффективности работы параллельной программы в зависимости от количества процессоров и количества стратегий для восьми игроков с использованием итерационного метода Брауна–Робинсона

4. Рекомендации по применению разработанных алгоритмов

На основании результатов вычислительного эксперимента можно предложить следующие варианты использования разработанных параллельных алгоритмов.

1. Размер исходных данных небольшой (не больше 7 игроков и количество стратегий не более 8), и при этом требуется точное решение кооперативной игры.

В этом случае нужно применить параллельный алгоритм решения кооперативной игры с использованием симплекс метода. В этом случае мы получим точное решение и приемлемое время, необходимое для решения задачи.

2. Размер исходных данных небольшой (не больше 7 игроков и количество стратегий не более 8), и при этом не требуется точное решение кооперативной игры.

В этом случае можно применить параллельный алгоритм решения кооперативной игры как с использованием симплекс метода, так и с использованием итеративного метода Брауна–Робинсона. В первом случае мы получим точное решение и приемлемое время, а во втором – решение, близкое к точному, и практически «мгновенное» решение задачи.

3. Размер исходных данных большой (больше 7 игроков), и при этом требуется точное решение кооперативной игры.

В этом случае вначале нужно применить параллельный алгоритм решения кооперативной игры с использованием итеративного метода Брауна–Робинсона. В результате мы получим решение, близкое к точному, и выигрышную коалицию. Затем, узнав вы-

игрышную коалицию, мы можем применить симплекс метод только для выигрышной коалиции для получения точного решения кооперативной игры.

4. Размер исходных данных большой (больше 7 игроков), и при этом не требуется точное решение кооперативной игры.

В этом случае вначале нужно применить параллельный алгоритм решения кооперативной игры с использованием итеративного метода Брауна–Робинсона. В результате мы получим решение, близкое к точному, и выигрышную коалицию.

Литература

1. Brown G.W. Iterative solution of games by fictitious play // Activity analysis of production and allocation (T. C. Koopmans, ed.). New York: Wiley. 1951. P. 374-376.
2. Carroll T.E., Grosu D. Formation of virtual organizations in grids: A game-theoretic approach // Economic Models and Algorithms for Distributed Systems, Autonomic Systems. 2010. Part I. P. 63–81.
3. Robinson J. The annals of mathematics, second series. 1951. Vol. 54, No. 2 (Sep.). P. 296-301.
4. Данциг Дж. Линейное программирование, его применения и обобщения. М.: Прогресс, 1966. 600 с.
5. Крушевский А.В. Теория игр. Киев.: Вища школа, 1977. 216 с.
6. Нестеренко М.Ю., Кириллов А.С. Разработка и анализ высокопроизводительных параллельных алгоритмов решения кооперативных игр // Вестник ЮУрГУ. Серия Математическое моделирование и программирование. 2011. Вып. 8, № 17 (234). С. 92-101.
7. Нестеренко М.Ю., Леонов Д.В., Болгова Е.В., Кириллов А.С. Разработка программного обеспечения для моделирования конкурентного рынка на кластерных системах // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Март–апрель 2008. № 54. Технологии высокопроизводительных вычислений и компьютерного моделирования. Санкт-Петербург, СПбГУ ИТМО, 2008. С. 156–161.
8. Оуэн Г. Теория игр. М.: Мир, 1971. 229 с.
9. Свидетельство о регистрации программы для ЭВМ № 2010614972. Решение кооперативных игр n-игроков с использованием технологии параллельного программирования MPI / М.Ю. Нестеренко, А.С. Кириллов. Зарегистрировано в Реестре программ для ЭВМ 29.07.2010 г.
10. Свидетельство о регистрации программы для ЭВМ № 2012612496. Решение кооперативных игр n-игроков с использованием итерационного алгоритма Брауна–Робинсона с помощью технологии параллельного программирования MPI / А.С. Кириллов. Зарегистрировано в Реестре программ для ЭВМ 07.03.2012 г.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ РАСЧЕТОМ СЛОЖНЫХ СТРОИТЕЛЬНЫХ ОБЪЕКТОВ НА БАЗЕ МЕТОДА РАЗДЕЛЕНИЯ ОБЪЕКТА НА ПРОЕКТНЫЕ ЕДИНИЦЫ

Д.И. Кислицын

Нижегородский государственный архитектурно-строительный университет

Описывается реализация вычислительной системы на базе локальной компьютерной сети проектной организации, позволяющей при незначительных финансовых затратах существенно ускорить процесс проектирования сложных строительных объектов. В основе разработанной системы лежит метод разделения объекта на проектные единицы. Подготовка данных (расчётные схемы, нагрузки, тепловые и другие воздействия) для проектных единиц выполняется проектировщиками параллельно на своих персональных компьютерах (ПК). Расчёт объекта как единой конструкции также выполняется на ПК проектировщиков под управлением специальной разработанной программы, обеспечивающей механическую совместимость проектных единиц. Применение данной системы позволяет распараллелить выполнение наиболее трудоёмких работ.

Метод разделения на проектные единицы впервые был разработан для упругих стержневых конструкций при силовых воздействиях и при разделении объекта на две проектные единицы [1, 2]. Кратко рассмотрим здесь данный метод.

Положим, что проектируемый объект B условно разбит на конечное число N проектных единиц (ПЕ) e_1, …, e_n,…, e_N поверхностями $R_{n,m}$ и $R'_{n,m}$, где n, m – номера условно-разделяемых элементов (рис. 1).

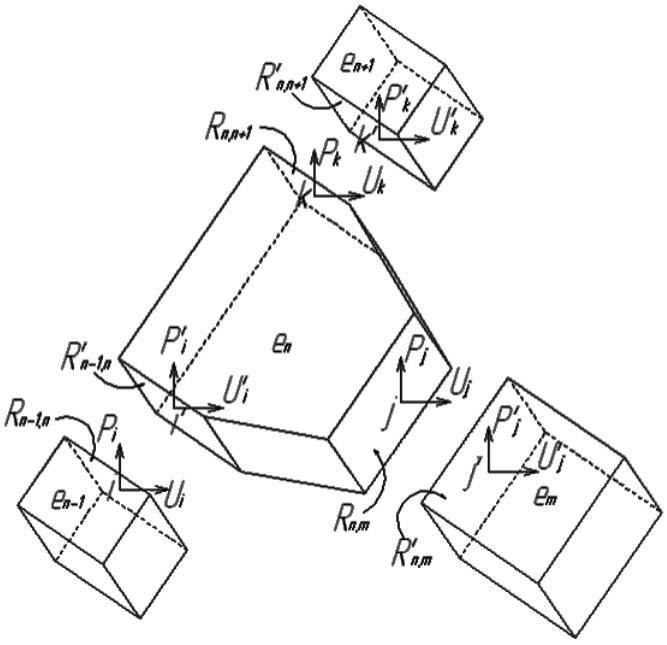


Рис. 1. Проектные единицы после условного разъединения

Условимся считать, что в каждом сечении взаимодействие условно разъединяемых элементов осуществляется через конечное множество $I_{n,m}$ = {…, j, …} «точек взаимодействия», где j – номер точки. Такими точками могут быть в расчётной схеме концы

условно перерезаемых стержней, узлы конечных элементов или суперэлементов и т.п. Ограничимся для простоты рассуждений только механическим взаимодействием элементов разделения.

Пусть $P^*(t) \equiv P^*$ – совокупность изменяющихся во времени t воздействий на объект B и P_n^* – часть P^*, приходящаяся на элемент e_n (n = 1, 2, …, N). Тогда в каждый момент времени взаимодействие каждой пары соседних элементов e_n и e_m будет характеризоваться совокупностью векторов обобщённых усилий $\boldsymbol{P}_j$ и обобщённых перемещений $\boldsymbol{U}_j$ ($j \in I_{n,m}$). Тогда условиями совместной работы разделённых элементов будут условия равновесия

$$\boldsymbol{P}_j + \boldsymbol{P'}_j = \boldsymbol{0}$$

и совместности

$$\boldsymbol{U}_j = \boldsymbol{U'}_j.$$

Вычислительная система для проектирования сложного строительного объекта может состоять из рабочих станций …, PC_{n-1}, PC_n, PC_{n+1}, PC_m, … и сервера, соединённых в локальную вычислительную сеть типа «звезда». Функции сервера может выполнять и одна из рабочих станций.

Опыт разработки автоматизированной системы управления расчётом сложных строительных объектов на базе метода разделения объекта на проектные единицы [3–5] показал, что практическая реализация системы сводится к решению следующих задач: разработка базы данных для централизованного хранения всех необходимых в процессе расчёта данных; разработка графического интерфейса пользователя; организация клиент-серверного взаимодействия между проектировщиками и сервером; формирование исходных данных на входном языке базового программного средства; управление базовым программным средством; графическое отображение структуры и параметров модели; формирование и решение систем уравнений; организация сбора и анализа результатов расчёта.

К настоящему времени в ННГАСУ разработана и проходит тестирование вторая версия системы, позволяющая проектным организациям при незначительных финансовых затратах существенно ускорить процесс проектирования сложных строительных объектов. Система была разработана в среде Microsoft Visual C# 3.5. В качестве СУБД в разработанном программном комплексе используется свободно распространяемая версия Microsoft SQL Server 2008 Express.

Структурная схема, иллюстрирующая работу разработанного программного средства, представлена на рис. 2. Проектировщики взаимодействуют с системой через графический интерфейс (блоки 1, 3, 5, 10). Подсистема графического отображения модели (блок 1а) позволяет проектировщикам визуально проверить структуру модели и взаимосвязи между проектными единицами. Клиент-серверное взаимодействие реализовано через блоки 0, 2, 4, 6, 7, 11, 15, 20, 21. Исходные данные на входном языке базового программного средства (в данной версии реализована поддержка ПК «Лира») формируются в блоках 8, 12, 17. За управление базовым программным средством отвечают блоки 13, 18. Формирование и решение системы уравнений реализованы в блоках 14, 16. Сбор и анализ результатов расчёта выполняется в блоках 19, 22.

Графический интерфейс позволяет работать в двух режимах: ГИП – главный инженер проекта и Проектировщик (рис. 3), обеспечивая тем самым взаимодействие ГИПа и проектировщиков с вычислительной системой в интерактивном режиме.

В режиме ГИП (рис. 2, блок 1) главный инженер проекта вводит:

- фамилию и свой личный идентификационный номер (ID);
- данные об объекте (наименование объекта, шифр проекта);
- количество проектных единиц (ПЕ);

- фамилии проектировщиков, которые будут рассчитывать «свои» ПЕ;
- ссылки на чертежи (рисунки) и иную проектную документацию для проектировщиков по каждой ПЕ.

Ввод данных о связях между ПЕ и их геометрических параметрах ГИП согласует с проектировщиками.

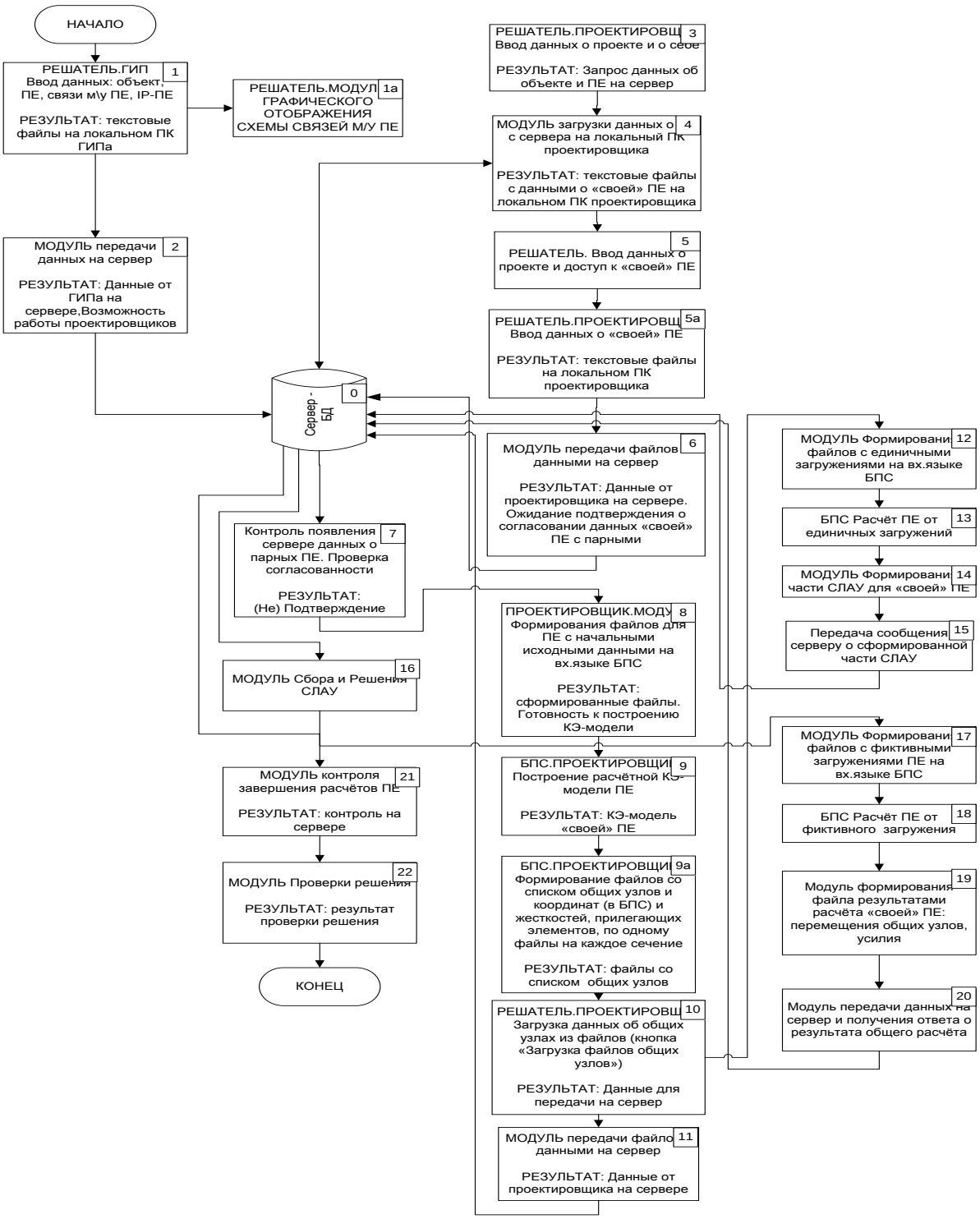


Рис. 2. Структурная схема разрабатываемой системы

Для связи с сервером в меню Настройки задается его IP-адрес. Все введённые данные ГИП может сохранить в файл или загрузить из файла соответствующими кнопками

(Сохранить, Загрузить, Очистить). После ввода всех необходимых данных ГИП кнопкой «Передать на Сервер» сохраняет данные в БД на сервере. Проект становится доступен для работы другим проектировщикам. Конструкторы работают в режиме «Проектировщик». Каждый проектировщик входит в систему под своей учётной записью и в соответствии с данными БД получает доступ только к «своему» проекту и «своей» ПЕ. Над одним проектом одновременно могут работать неограниченное число проектировщиков.

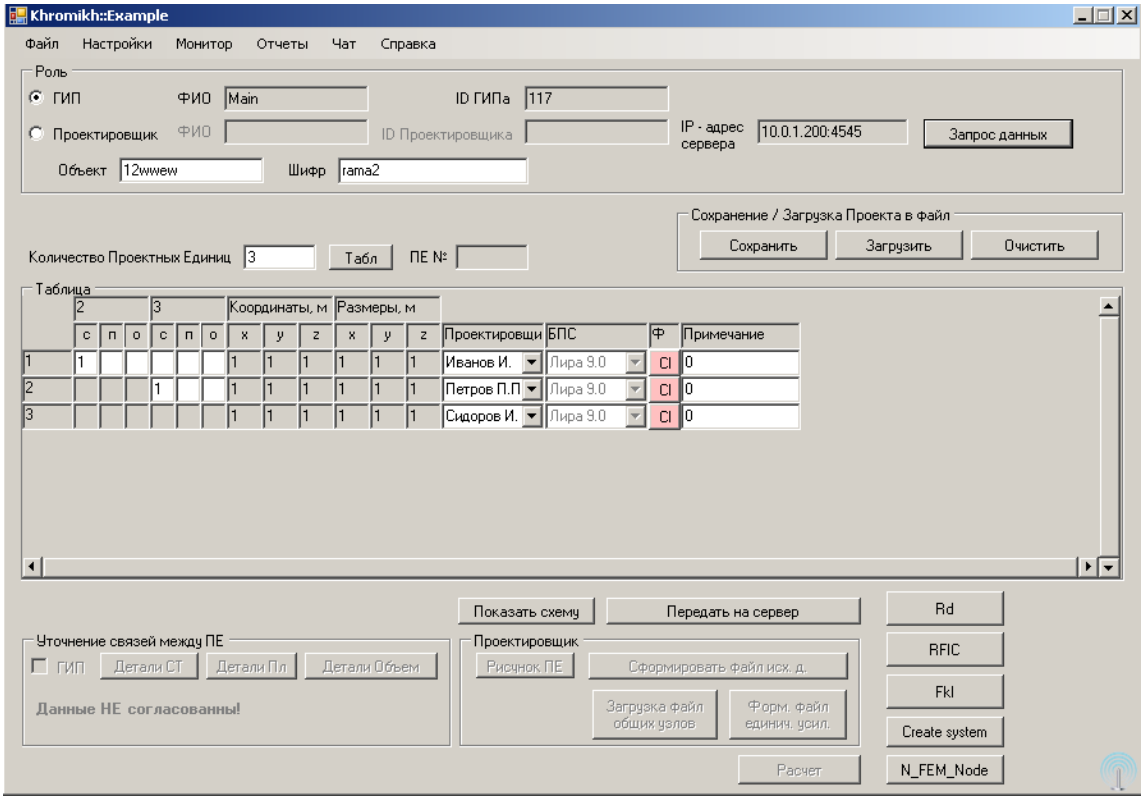


Рис. 3. Главная форма работы с программой

По нажатию кнопки «Запрос данных» в форму проектировщиков из БД подгружается введённая ГИПом информация по проекту (рис. 2, блок 3). Далее проектировщики в отдельной форме, открывающейся по кнопке «Детали СТ», указывают базовое программное средство (БПС) для «своей» ПЕ и координаты общих узлов (рис. 2, блок 5). После ввода данных, уточняющих взаимосвязи между ПЕ, необходимо согласовать эти данные кнопкой «Согласовать данные». После автоматического согласования координат общих узлов на сервере проектировщики строят в БПС модели «своих» ПЕ, а затем кнопкой «Загрузка файлов общих узлов» через соответствующую форму загружают сформированные проектировщиками в БПС файлы со списками номеров и координат общих узлов в системе БПС для передачи в БД на сервер (рис. 2, блок 10). Далее, на всех локальных компьютерах проектировщиков выполняется автоматический расчёт объекта по методу разделения на ПЕ с последующей автоматической проверкой полученных результатов расчётов. Результаты сохраняются в БД сервера, а также выводятся в виде сообщений проектировщикам и ГИПу. Файлы с моделями ПЕ и результатами расчётов сохраняются на локальных компьютерах проектировщиков. Таким образом, проектировщики могут продолжить работу с ПЕ, например, подготовить конструкторскую документацию.

Подсистема клиент-серверного взаимодействия построена по трехуровневой архитектуре в виде совокупности трех компонент: сервера базы данных, клиентского приложения и сервера приложений, отвечающего за выполнение логики приложения. Вы-

бор данной архитектуры во многом обусловлен тем, что в классической двухуровневой клиент-серверной архитектуре помимо сложности масштабирования, существует и необходимость установления соединения между базой данных и каждым работающим с системой пользователем. Преимуществом выбранной трехуровневой модели является то, что в ней сервер базы данных отвечает только за хранение данных и обработку запросов.

Подсистема формирования исходных данных на входном языке базового программного средства создаёт и модифицирует текстовые файлы на входном языке выбранного базового программного средства (в данной версии реализовано взаимодействие с ПК «Лира»). В блоке 8 выполняется формирование файла, содержащего общие для смежных ПЕ узлы. В блоке 12 в файл ПЕ добавляется информация о единичных загружениях. В блоке 17 в файл ПЕ добавляется информация о вычисленных усилиях взаимодействия.

По результатам расчёта ПЕ от действия единичных усилий подсистема сбора и анализа результатов расчёта формирует файл с данными о перемещениях общих узлов (рис. 2, блок 19), а по итогам всех расчётов выполняет проверку решения (рис. 2, блок 22).

Вычисление усилий взаимодействия выполняется подсистемой формирования и решения систем уравнений (рис. 2, блоки 14, 16). Система формируется на основании результатов расчёта ПЕ от единичных усилий, которые посредством подсистемы клиент-серверного взаимодействия передаются в БД сервера.

Расчёты ПЕ от единичных усилий (рис. 2, блок 13) и усилий взаимодействия (рис. 2, блок 18) выполняются на базовых программных средствах, управление которыми осуществляется через подсистему управления базовым программным средством.

Разработанная система автоматизирует и достаточно строго формализует технологию расчёта сложных строительных объектов методом разделения на проектные единицы.

Литература

1. Супрун А.Н., Кислицын Д.И. Распараллеливание вычислительных расчётов строительных конструкций на персональных компьютерах // Известия ВУЗов. Строительство. Новосибирск. 2006. №5.
2. Кислицын Д.И., Супрун А.Н. Программный модуль для расширения функциональных возможностей вычислительного комплекса «Лира» // International Journal-for Computational Civil and Structural Engineering. 2008. V. 4, Iss. 2. Moscow, New York.
3. Супрун А.Н., Кислицын Д.И., Скороходов В.В. Проблемы построения автоматизированных систем конструкторского расчёта строительных объектов в распределённых вычислительных средах // Актуальные проблемы компьютерного моделирования конструкций и сооружений: Материалы III Международного симпозиума / Юж.-Рос. гос. техн. ун-т. (НПИ). Новочеркасск, 2010.
4. Suprun A.N., Kislitsyn D.I. Distributed computing for construction project design by division into project design units // Computing in Civil and Building Engineering, Proceedings of the International Conference. Nottingham, 2010.
5. Suprun A.N., Kislitsyn D.I. The multilevel parallelization of structural design calculation in distributed computing environment //14th International Conference on Computing in Civil and Building Engineering (14th ICCCBE). Moscow, 2012.

ОРГАНИЗАЦИЯ ПОЗИЦИОННО-МОДУЛЯРНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ВЫСОКОТОЧНЫХ ЧИСЛЕННЫХ РАСЧЕТОВ

В.С. Князьков, К.С. Исупов, А.В. Логинов

Вятский госуниверситет

Рассматриваются программные решения комплекса High Precision Digit-Parallel Solver (HPDP-Solver), в основу которого заложен новый формат представления вещественных чисел в форме с плавающей точкой, что позволяет организовать распараллеливание вычислений до уровня знакопозиций мантисс чисел в многоядерных процессорах. В веденном модулярно-позиционном (квазимодулярном) формате данных для представления мантисс чисел используется многомодульная модулярная система счисления (система остаточных классов), а порядки чисел представляются в позиционной системе. Задача использования арифметики систем остаточных классов для численных расчетов с плавающей точкой на многоядерных процессорах общего назначения до настоящего времени не решалась.

Введение

Эффективность применения модулярных систем при обработке больших массивов целых чисел продемонстрирована в работах [1–3]. В работе [4] предложен способ применения модулярных систем счисления для ослабления влияния ошибок округления компьютерных вычислений с фиксированной точкой при обработке рациональных и комплексно-рациональных чисел. Однако до сих пор модулярные системы счисления не применялись для вычислений с плавающей точкой на платформах универсальных многоядерных процессоров. Именно на решение этой задачи нацелена предлагаемая разработка.

Модулярно-позиционный формат для высокоточных разрядно-параллельных вычислений с плавающей точкой

В двоичных форматах с плавающей точкой (рассматриваются форматы IEEE-754) любое вещественное число представляется трехэлементным набором:

$$\left[M, e, s \,\middle|\, M \in [0, 2),\, e \in [e_{\min}, e_{\max}], s \in \{0,1\}\right], \qquad (1)$$

где M – рациональная мантисса, e – порядок, $e_{\min} = 2 - 2^{w-1}$, $e_{\max} = 2^{w-1} - 1$, s – знак числа.

Значение чисел, записанных в таком формате, определяется в общем виде формулой $-1^{s} \cdot M \cdot 2^{e}$. Машинными представлениями чисел вида (1) являются $(w+t+1)$-разрядные двоичные векторы $\langle s r_w ... r_2 r_1 d_t ... d_2 d_1 \rangle$, где разряды с d_1 по d_t отводятся под представление рациональных двоичных мантисс $M = d_t . d_{t-1} ... d_2 d_1$, разряды с r_1 по r_w отводятся под представление целочисленных порядков e, записанных в форме с избытком $E = r_w r_{w-1} ... r_2 r_1 = e + e_{\max}$, разряд s выражает знак числа.

Принимая во внимание условие, что под целую часть рациональной мантиссы $M = d_t . d_{t-1} ... d_2 d_1$ в двоичных форматах вида (1) отводится 1 бит d_t, определим целочисленную мантиссу $M' = d_t d_{t-1} ... d_2 d_1$ как t-разрядное неотрицательное целое двоичное

число, такое, что $M = M' \cdot 2^{1-t}$. Определим перемещенный порядок λ, как целое двоичное число со знаком, такое, что $\lambda = e - t + 1$, где e – w-разрядный порядок числа, представленного в двоичном формате (1).

Зададим n целочисленных положительных оснований системы остаточных классов [5, 6] $p_1, p_2, ..., p_n$ таких, что:

$$\begin{cases} \forall i_1, i_2 \in \{1, 2, ..., n\}, i_1 \neq i_2 : \gcd(p_{i_1}, p_{i_2}) = 1, \\ p_i = 2^{f_i} - 1, i = 1, 2, ..., n, \end{cases}$$

где $\gcd(p_{i_1}, p_{i_2})$ – наибольший общий делитель для p_{i_1} и p_{i_2}.

Целочисленную мантиссу $M' = d_t d_{t-1} ... d_2 d_1$ преобразуем в систему остаточных классов (СОК) с заданными основаниями $p_1, p_2, ..., p_n$, получая тем самым *модулярную мантиссу* $\tilde{M} = \langle m_1, m_2, ..., m_n \rangle$:

$$\tilde{M} = \langle m_1, m_2, ..., m_n \rangle = \left\langle |M'|_{p_1}, |M'|_{p_2}, ..., |M'|_{p_n} \right\rangle,$$

где $m_i \in [0, p_i - 1], i = 1, 2, ..., n$ – цифры (модулярные разряды) модулярной мантиссы $\tilde{M}$, $|M'|_{p_i}$ – операция получения остатка от деления M' на i-е основание p_i.

С порядком λ оперируют в позиционной двоичной системе счисления. Таким образом, любое число с плавающей точкой (ПТ) вида (1) можно представить в следующем модулярно-позиционном формате (рис. 1):

$$\left[\langle m_1, m_2, ..., m_n \rangle, \lambda, s \middle| m_i \in [0, p_i - 1], \lambda \in [\lambda'_{\min}, \lambda'_{\max}], s \in \{0, 1\} \right], \quad (2)$$

где $\langle m_1, m_2, ..., m_n \rangle$ – набор разрядов-цифр модулярной мантиссы $\tilde{M}$; λ – позиционный перемещенный порядок, представляющий собой целое двоичное число со знаком.

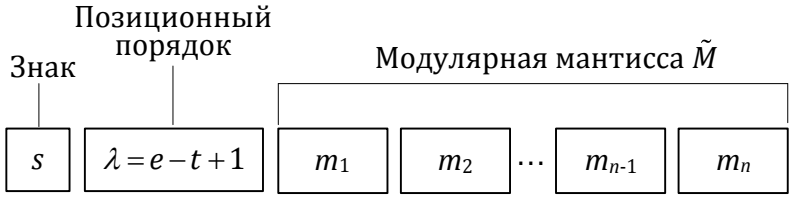


Рис. 1. Модулярно-позиционный формат с плавающей точкой

Представление чисел с ПТ в модулярно-позиционном формате (2) позволяет решить сразу несколько проблем, присущих позиционным представлениям вида (1):

- во-первых, увеличивается точность вычислений, которая определяется разрядностью мантисс (модулярные мантиссы $\tilde{M} = \langle m_1, m_2, ..., m_n \rangle$ в общем случае могут изменяться в пределах интервала $[0, P - 1]$, где $P = \prod_{i=1}^{n} p_i$ – произведение оснований $p_1, p_2, ..., p_n$), поэтому для достижения точности 4*double достаточно выбрать восемь 32-битных оснований $p_1, p_2, ..., p_8$;
- во-вторых, ввиду взаимной независимости цифр-знакопозиций $m_i, i \in [1, n]$, модулярных мантисс обеспечивается возможность параллельного оперирования с отдельными цифрами-знакопозициями операндов при арифметических вычислениях;
- в-третьих, решается задача минимизации скорости вычислений при выполнении арифметических операций в зависимости от требуемой точности (при добавлении

дополнительных оснований p_i для увеличения точности, в случае если их число превышает число вычислительных устройств, время выполнения операций увеличивается линейно, а не экспоненциально, как в высокоточных позиционных библиотеках).

Модулярно-позиционные структуры могут быть эффективно использованы при высокоточном решении задач, когда использование форматов с фиксированной точкой принципиально невозможно в силу природы самих задач, а машинные форматы с плавающей точкой не обеспечивают требуемой точности. Использование модулярно-позиционного формата видится целесообразным при решении задач над массивами данных большой размерности, когда время, затрачиваемое на прямое и обратное преобразование несопоставимо мало, по сравнению со временем непосредственно расчетов.

В качестве примера на рис. 2 представлены схемы параллельных алгоритмов умножения и сложения модулярно-позиционных операндов. Для реализации операций деления и сравнения по величине модулярных мантисс, определения их выхода за допустимый диапазон представления и преобразования модулярных мантисс в позиционную систему, являющихся наиболее сложными операциями в СОК, предлагается использовать эффективные методы и алгоритмы, предложенные в работах [5, 6].

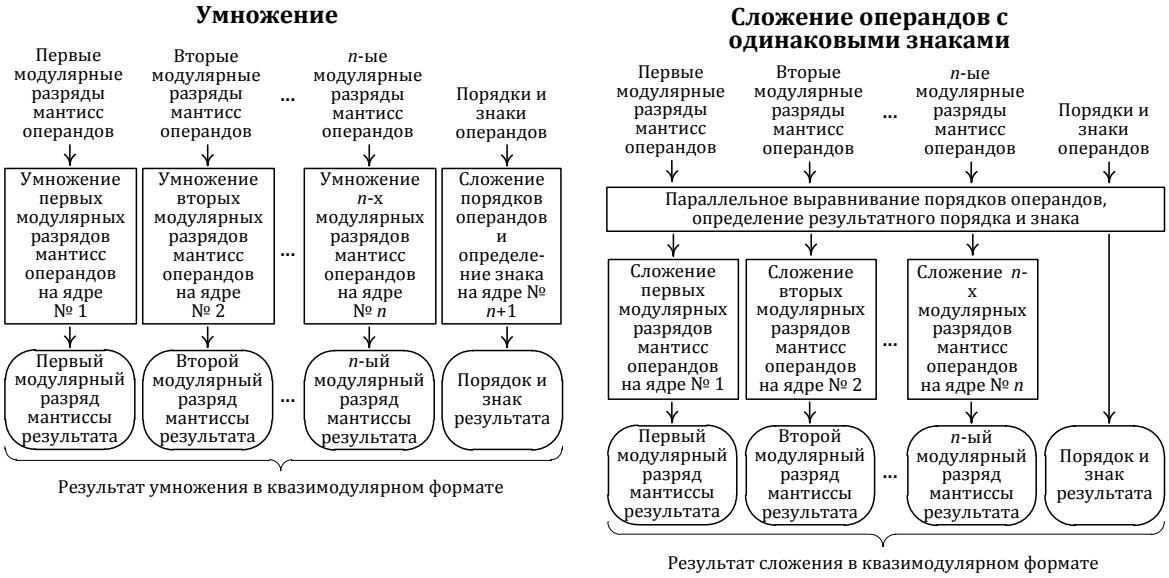


Рис. 2. Примеры алгоритмов выполнения арифметических операций с плавающей точкой в модулярно-позиционном формате

Введение новых способов организации арифметических вычислений с использованием модулярно-позиционного формата с плавающей точкой для представления вещественных чисел вызывает необходимость проектирования оригинальных схем декомпозиции данных по узлам и ядрам вычислительных устройств. Поэтому в комплексе используются специальные схемы распределения модулярно-позиционных структур данных между узлами и вычислительными ядрами в пределах узлов.

Для оценки эффективности применения модулярно-позиционного формата при высокоточном решении численных задач над массивами данных с плавающей точкой был проведен эксперимент, направленный на исследование быстродействия серверной части пакета HPDP-Solver в сравнении с широко известной программной библиотекой The GNU MP Bignum Library (GMP).

В качестве алгоритмической базы для эксперимента выступали высокоточные параллельные алгоритмы умножения плотных матриц $C = A \cdot B,\ A,B,C \in R^{n \times n}$ и скалярного произведения векторов $c = a \times b,\ a,b,c \in R^n$.

На рис. 3 представлен график зависимости времени умножения матриц 1500×1500 от точности счета при распараллеливании вычислений на восемь ядер.

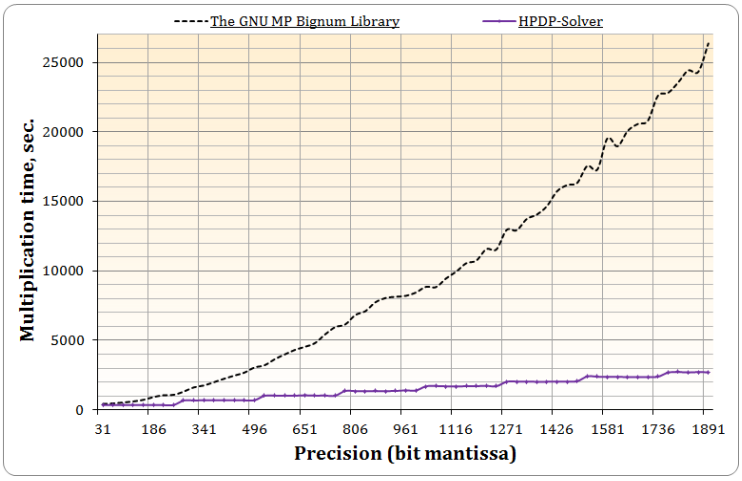


Рис. 3. Изменение времени умножения матриц при росте разрядности мантисс: счет на 8 ядрах

Литература

1. Исупов К.С., Князьков В.С. Инструментальный комплекс для проектирования параллельных масштабируемых программ численных расчетов // Научно-технический вестник СПбГУ ИТМО. 2010. № 6(70). С. 68–72.
2. Knyazkov V., Isupov K. Tool complex for performance of precision parallel calcula-tions in basis of matrix algebra // Information technologies in science, education and busi-ness: Catalog CeBIT 2011 the Ministry of Education and Sience of the Russian Federation. Participants Vyatka State University. 2011. 9 p.
3. Исупов К.С. Параллельные вычисления над многоразрядными числами в системе остаточных классов // Международная суперкомпьютерная конференция «Научный сервис в сети Интернет: экзафлопсное будущее»: Сб. трудов. М.: МГУ, 2011. С. 534–540.
4. Оцоков Ш.А. Структурно-алгоритмические методы организации высокоточных вычислений на основе теоретических обобщений в модулярной системе счисления: диссертация доктора технических наук. М., 2010. 287 с.
5. Sabbagh A., Navi K. New Arithmetic Residue to Binary Converters // IJCSES International Journal of Computer Sciences and Engineering Systems. 2007. V. 1, No. 4. P. 295–299.
6. Chang C., Lai Y. A Division Algorithm For Residue Numbers // Applied Mathematics and Computation. 2006. No. 172(1). P. 368–378.

ОРГАНИЗАЦИЯ АРИФМЕТИЧЕСКОГО РАЗРЯДНО-ПАРАЛЛЕЛЬНОГО СБИС-ПРОЦЕССОРА ДЛЯ МАССОВЫХ ВЫЧИСЛЕНИЙ

В.С. Князьков, И.П. Осинин, Т.В. Волченская

Вятский государственный университет

Рассматривается организация мультиоперандного суммирования *m n*-разрядных двоичных чисел [2] на основе техники подсчета количества единичных бит в разрядных срезах, формируемых из разрядов суммируемых чисел. Техническим результатом от использования способа организации вычислений суммы *n m*-разрядных чисел является повышение скорости вычислений за счет замены серии из *n* арифметических операций сложения *m* параллельно-разрядными исполняемыми операциями подсчета количества единичных бит. В статье приведена техническая реализация способа в виде устройства суммирования двоичных чисел на базе пирамидального выделения переносов в последующие разрядные срезы. Для приведения к мультиоперандному суммированию таких арифметических операций как вычитание, умножение и деление, необходимо преобразование исходных операндов. Схемотехническая реализация подобных преобразователей также приведена в статье.

Введение

Решением задачи повышения скорости вычислений в арифметико-логических устройствах (АЛУ) при приемлемых аппаратных затратах является построение СБИС-процессоров, операционная часть которых представляет собой универсальное вычислительное пространство, перестраиваемое для выполнения конвейерно-параллельной обработки различных арифметико-логических операций [1].

Одноразрядный конвейерный сумматор

Для построения мультиоперандного сумматора используется одноразрядный сумматор, в котором процесс суммирования операндов является параллельно-конвейерным. На рис. 1 приведена функциональная схема предлагаемого одноразрядного конвейерного сумматора, где a, b – информационные входы; clk – вход синхронизации; s – выход устройства; AND2 – двухвходовой элемент «И»; XOR – элемент «ИСКЛЮЧАЮЩЕЕ ИЛИ»; OR2 – двухвходовой элемент «ИЛИ»; NOT – элемент «НЕ»; DFF – информационный триггер.

Одноразрядный конвейерный сумматор состоит из однобитного полного сумматора и элементов памяти, которые выполнены на информационных триггерах. Элементы памяти необходимы для организации конвейерного режима обработки чисел, причем в канале переноса необходим двухступенчатый триггер для исключения гонок при изменении состояния, выполненный из двух информационных триггеров (inst3 и inst8) и элемента «НЕ».

Данный сумматор работает следующим образом: на входы a и b последовательно такт за тактом подаются разряды первого и второго слагаемого, начиная с младших разрядов. На выходе s формируются разряды суммы в той же последовательности. Время заполнения конвейера равно одному такту, то есть i-й разряд суммы будет доступен на выходе устройства в (i+1)-м такте, где $i \in [1, n]$.

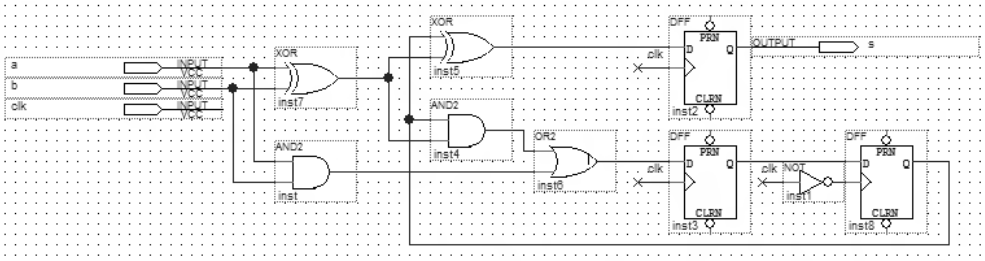


Рис. 1. Функциональная схема одноразрядного конвейерного сумматора

Так как в устройстве цепь распространения сигнала имеет три логических элемента, время задержки распространения сигнала составляет 3**t*, где *t* – время задержки сигнала одним логическим элементом. На рис. 2 представлены результаты экспериментального моделирования данной схемы.

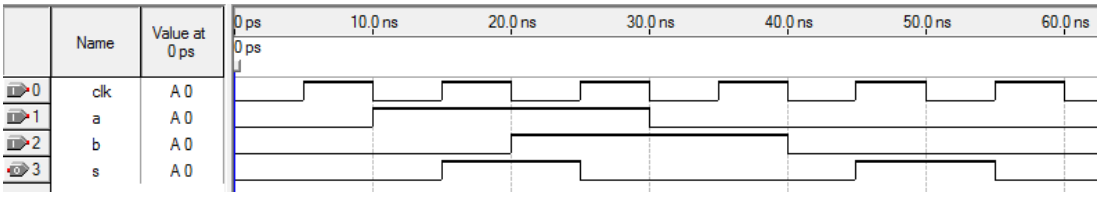


Рис. 2. Временная диаграмма работы одноразрядного конвейерного сумматора

Исходные слагаемые *a*=11 и *b*=110 поразрядно подаются на входы устройства каждый такт синхронизации, начиная с младших. С 15 нс каждый такт выдаются разряды искомой суммы *s*=1001. Время такта равно 10 нс.

Мультиоперандный сумматор

Устройство для конвейерных вычислений суммы *m* *n*-разрядных чисел (мультиоперандный сумматор) представляет собой однородную вычислительную среду, в роли ячейки которой выступает рассмотренный одноразрядный конвейерный сумматор. На рис. 3 представлено устройство для конвейерных вычислений суммы восьми *n*-разрядных чисел, где ОКС – одноразрядные конвейерные сумматоры (ячейки однородной вычислительной среды); информационные входы X_1-X_8; информационный выход Y. В общем виде устройство имеет также пирамидальную структуру, в каждом следующем столбце в два раза меньше ячеек, чем в предыдущем, причем количество столбцов равно $\log_2 m$.

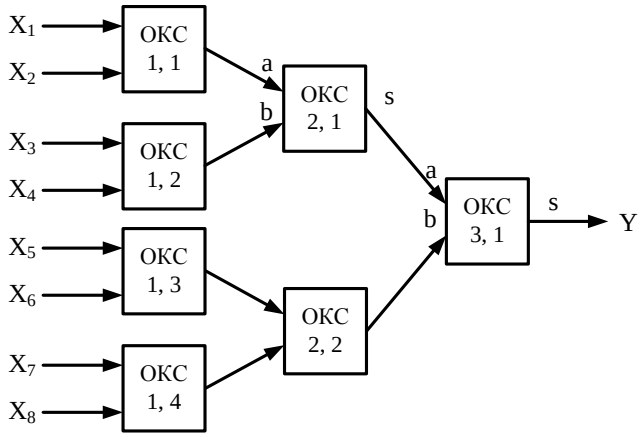


Рис. 3. Структурная схема мультиоперандного сумматора

Общее количество ячеек равно m-1. Вход синхронизации однородной вычислительной среды соединен с входами синхронизации всех ячеек. Массив исходных m n-разрядных двоичных чисел поступает на обработку в виде n двоичных m-мерных векторов, причем, m должно быть кратно степени двойки, первый и второй информационные входы ячеек первого столбца однородной вычислительной среды соединены с соответствующими разрядами m-разрядного входного вектора. Информационный выход каждой (i-1, j)-й и (i, j)-й ячейки подключен соответственно к первому и второму информационному входу (i/2, j+1)-й ячейки, причем $i \in [2, m/2^j]$, i принимает лишь четные значения.

Количество ячеек в j-м столбце однородной вычислительной среды равно $m/2^j$, количество столбцов однородной вычислительной среды равно p, где $p=\log_2 m$. Информационный выход ячейки последнего столбца однородной вычислительной среды является выходом схемы, с которого снимается результат.

Каждое i-е двоичное позиционное слагаемое можно представить в виде последовательности бит $A_i(a_n, a_{n-1}, \ldots, a_1)$, где n-разрядность числа, $i \in [1, m]$. Тогда m слагаемых можно представить в виде матрицы:

$$\begin{pmatrix} a_{1,n}, a_{1,n-1}, \ldots, a_{1,1} \\ a_{2,n}, a_{2,n-1}, \ldots, a_{2,1} \\ \ldots \\ a_{m,n}, a_{m,n-1}, \ldots, a_{m,1} \end{pmatrix}. \qquad (1)$$

Столбцы матрицы с элементами ($a_{1,j}$, $a_{2,j}$, …, $a_{m,j}$) являются входными векторами, которые поступают на обработку в однородную вычислительную среду.

В каждом такте на входы синхронизации всех триггеров подается сигнал синхронизации. На первый и второй информационные входы ячеек первого столбца подаются соответствующие биты разрядных срезов, причем каждый следующий разрядный срез подается на следующий такт работы устройства. С каждым следующим продвижением количество разрядов вектора уменьшается вдвое. Так продолжается до тех пор, пока количество разрядов передаваемого вектора не станет равным единице. Пройдя все столбцы однородной вычислительной среды, количество бит в i-м разрядном срезе сокращается до одного, данный бит является i-м разрядом искомой суммы исходных операндов. Формирующиеся при этом единицы переносов посредством обратной связи в ячейке передаются на обработку в следующий разрядный срез.

В результате через $\log_2 m$ тактов работы устройства формируется младший бит суммы m n-разрядных чисел, причем m должно быть кратно степени двойки. После этого конвейер является заполненным и биты результата доступны на выходе устройства каждый последующий такт работы. Так как в каждом такте работы устройства вектор передается в соседний справа столбец матрицы, на вход устройства каждый такт должен подаваться следующий вектор. Таким образом, устройство реализует конвейерный принцип обработки информации. Так как в ячейке самая длинная цепочка распространения сигнала имеет три логических элемента, время задержки распространения сигнала составляет $3*t$, где t – время задержки сигнала одним логическим элементом.

Если принять за время суммирования пары n-разрядных чисел n тактов работы устройства, то время вычисления суммы в предлагаемом устройстве в конвейерном режиме равно n тактов, в то время как время суммирования пирамидальным способом на базе традиционных сумматоров равно $p*n$ тактов, где $p=\log_2 m$. Таким образом, быстродействие устройства на базе описанного способа в $\log_2 m$ раз выше по сравнению с быстродействием устройства на базе известного итерационного способа суммирования. Например, при количестве слагаемых m=64 быстродействие предлагаемого устройства больше в 8 раз.

Количество тактов, необходимое для параллельно-разрядного вычисления суммы или разности *m n*-разрядных чисел в предложенном параллельно-разрядном устройстве не зависит от количества участвующих операндов и равно *n*. Если принять за время суммирования пары операндов обычным сумматором *n* тактов работы устройства, то количество тактов суммирования известным итерационным способом (с помощью накопителя частных сумм) равно $m*n$. Количество тактов суммирования известным пирамидальным способом (складываются пары операндов, затем пары результатов частных сумм) равно $\log_2 m*n$. Например, при количестве операндов m=64 вычисления будет выполняться в 64 раза быстрее, чем при итерационном способе, и в восемь раз быстрее, чем при пирамидальном. Рисунок 4 отражает зависимость количества тактов работы для разных типов сумматоров от разрядности операндов (количество операндов m=16).

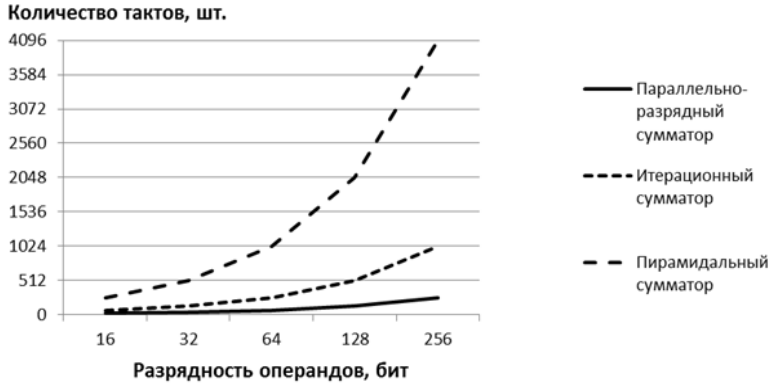


Рис. 4. Диаграмма зависимости количества тактов работы сумматора от разрядности операндов

Устройство для умножения чисел на базе мультиоперандного сумматора

В статье [2] предлагается оригинальный способ умножения двух *m*-разрядных целых двоичных чисел без остатка, основанный на подсчете количества единичных бит в разрядных срезах, формируемых из разрядов множителя. Техническая реализация данного способа требует преобразования исходных чисел для последующего мультиоперандного суммирования. Предлагается специальная функциональная схема ячейки формирователя разрядных срезов сомножителей (рис. 5), где: x1, x2 – информационные входы; clk – вход синхронизации; z1 – управляющий вход; DFF – информационный триггер; AND2 – двухвходовой элемент «И»; z2, x3 – управляющие выходы; y – информационный выход.

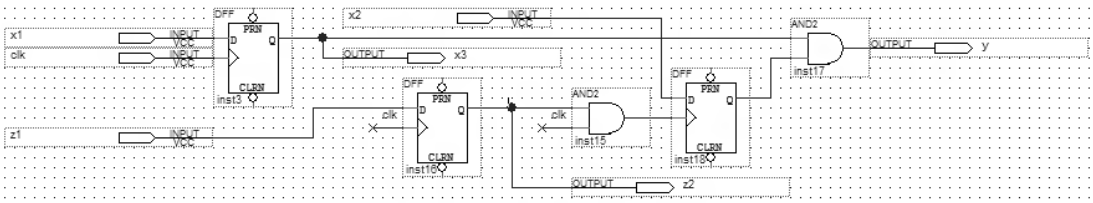


Рис. 5. Функциональная схема ячейки формирователя разрядных срезов сомножителей

Перед началом операции умножения на управляющий вход устройства в течение одного такта подается значение логической единицы. В последующие такты на управляющий вход устройства подается значение логического нуля, а на входы x1 и x2 последовательно подаются значения разрядов сомножителей, начиная с младших. Через $\log_2 m$+1 тактов после начала операции умножение на выходе мультиоперандного сумматора будет доступен младший бит искомого результата умножения. Каждый после-

дующий такт в течение 2*m тактов работы устройства соответствующий бит результата умножения будет доступен на выходе устройства.

Количество тактов, необходимое для параллельно-разрядного вычисления произведения двух n-разрядных чисел в предложенном устройстве равно 2*n. При этом количество тактов умножения известным итерационным способом (с помощью накопителя частных сумм) равно n^2. Например, при разрядности операндов n=64 вычисления в предложенном устройстве будут выполняться в 32 раза быстрее, чем при итерационном способе умножения. Рисунок 6 отражает зависимость числа тактов работы устройств рассмотренных типов от разрядности операндов.

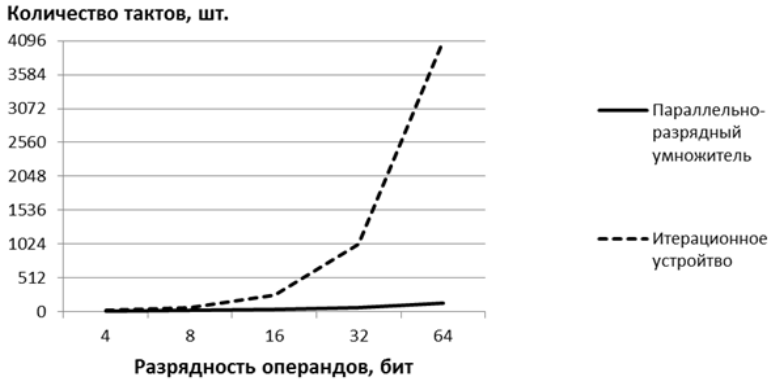


Рис. 6. Диаграмма зависимости количества тактов работы устройства умножения (деления) от разрядности операндов

Литература

1. Осинин И.П., Князьков В.С. Способ и устройство для конвейерных вычислений суммы m n-разрядных чисел // Сборник материалов VI Всероссийской молодежной научно-инновационной школы «Математика и математическое моделирование». Саров, 17–20 апреля, 2012. С. 89–97.
2. Осинин И.П., Князьков В.С. Способ организации вычислений суммы n m-разрядных чисел // Первый Всероссийский конгресс молодых ученых: Тез. докл. Санкт-Петербург, 10–13 апреля, 2012. – С. 87–88.
3. Осинин И.П., Князьков В.С. Концепция модулярно-систолической структуры для массовых арифметических вычислений // Сборник трудов XIX Всероссийской научно-методической конференции «Телематика'2011»: Тез. докл. Санкт-Петербург, 25 - 28 июня, 2012. С. 323–324.

АЛГОРИТМ ПОИСКА СОВПАДАЮЩИХ ПОДСТРОК ДВУХ ГЕНОМОВ И ЕГО РАСПАРАЛЛЕЛИВАНИЕ

Е.А.Коваль, О.А. Коваль

Объединенный институт ядерных исследований, Дубна

Введение

В последние годы поиск совпадающих последовательностей ДНК, так называемая (DNA contamination problem), являющихся основным компонентом геномов, превратилось в неотъемлемую часть генной инженерии.

Причины значительного количества совпадений довольно больших участков в геномах, а также высокого разнообразия их состава до конца не изучены. До недавнего времени алгоритмы поиска точно совпадающих подстрок были слишком ресурсозатратны и трудоемки для поиска совпадающих последовательностей больших геномов, таких, как геномы животных и растений, и позволяли находить лишь приближенную схожесть участков геномов (BLAT [1] и др.). Появление новых алгоритмов поиска точно совпадающих последовательностей (в частности Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals [2], SSAHA [3], ACMES [4]) позволило группе Jeff Reneker-а из Университета Миссури получить выдающиеся результаты, в частности, были обнаружены *ультра консервативные элементы (УКЭ) – достаточно большие участки геномов, имеющиеся в абсолютно идентичном виде в геномах не только многих животных, но и растений* [5]. Полагают, что они играют важную роль в стабилизации и поддержании структуры хромосом, участвуют в «узнавании» и правильном их расхождении во время митоза и мейоза, выступают точками рекомбинации, выполняют функцию защиты теломер и участвуют в регуляции активности генов. Неудивительно, что они привлекают внимание исследователей. Поэтому важной задачей является поиск новых генетических маркеров.

При этом приходится обрабатывать огромные количества информации – сравнивать геномы друг с другом на совпадение подстрок любой длины, большей некоторой, причем размеры геномов млекопитающих достигают порядка гигабайтов символов (1024^3 элементов). При этом разработка эффективных по требованиям к памяти быстрых параллельных алгоритмов выступает на первый план.

1. Описание задачи и простейший алгоритм решения проблемы

Задача: Поиск совпадающих подстрок, длиной более некоторой минимальной, при попарном сравнении «эталонного» генома с набором других геномов.

Опишем задачу на простейшем примере.

Имеется «эталонный» геном и набор сравниваемых с ним геномов:

- Reference (эталонный геном): TTAAGATCAATT,
- первый сравниваемый геном (Test_input1): TTGATCTT,
- второй сравниваемый геном (Test_input2): TTGATCTTAGCAT.

Задача заключается в нахождении в них одинаковых подстрок, например длиной более двух символов, причем если они пересекаются, то мы объединяем их в более длинную подстроку. Принципиальный простейший и очень наглядный алгоритм поис-

ка совпадающих подстрок иллюстрируют представленные ниже рисунки. Заранее мы не знаем, какие подстроки двух геномов совпадают, поэтому алгоритмы поиска заранее известного паттерна в строке, такие, как алгоритмы Кнута–Морриса–Пратта, Бойера–Мура и другие, неприменимы (мы не упомянули алгоритм Рабина–Карпа, т.к. обобщение его идеи послужило базисом нашего решения). Простейший алгоритм «грубой силы» состоит в составлении таблицы совпадения каждого символа эталонного генома с каждым символом сравниваемого генома и выделении наиболее длинных цепочек рядом стоящих символов (указано стрелками на рис. 1). В качестве ответа выводятся положение подстроки (номер начального символа и конечного символа) в эталонном геноме и положение аналогичного участка в сравниваемом геноме.

	T	T	A	A	G	A	T	C	A	A	T	T
T	1	1									1	1
T	1	2									1	2
G					1							
A			1	1		2			1	1		
T	1	1					3				2	
C								4				
T	1	1					1				1	1
T	1	2					1				1	2

Рис. 1. Простейший алгоритм решения проблемы нахождения совпадающих подстрок. Таблица сравнения reference строки (по горизонтали) и test_input1 (по вертикали)

Из рис.1 можно видеть, что ответом на поставленную задачу являются подстроки (напомним, что ищутся совпадающие подстроки длиной больше либо равные 2):

- test_sequence_1
  1 2 1 2 (TT)
  1 2 7 8 (TT)
  5 8 3 6 (GATC)
  10 11 4 5 (AT)
  11 12 1 2 (TT)
  11 12 7 8 (TT)

	T	T	A	A	G	A	T	C	A	A	T	T
T	1	1									1	1
T	1	2									1	2
G					1							
A			1	1		2			1	1		
T	1	1					3				2	
C								4				
T	1	1					1				1	1
T	1	2					1				1	2
A			3	1		1				1		
G					2							
C								1				
A			1	1		1			2	1		
T	1	1					2				2	

Рис. 2 Таблица сравнения reference строки (по горизонтали) и test_input2 (по вертикали)

Аналогично решением для второго генома из примера, как следует из рис. 2, являются подстроки:

- test_sequence_2
  1 2 1 2 (TT)
  1 3 7 9 (TTA)
  4 5 9 10 (AG)
  5 8 3 6 (GATC)
  6 7 12 13 (AT)
  8 9 11 12 (CA)
  10 11 4 5 (AT)
  10 11 12 13 (AT)
  11 12 1 2 (TT)
  11 12 7 8 (TT)

Задача решалась в рамках конкурса Intel Accelerate 2012 и организаторы любезно предоставили участникам реализацию простейшего алгоритма в виде референсного кода. Таким образом, задача в рамках конкурса ставилась в виде: ускорить и распараллелить референсное решение [6] с полным сохранением условий вывода. (На входе имеются *reference* и набор *input* строк, необходимо найти все совпадающие подстроки длиной больше *minMatchLength* с дополнительным условием фильтрации результатов: максимизирующими тройку (i,j,length), где i – индекс подстроки в *reference* строке, i – индекс подстроки в *input* строке).

1.1. Описание входных данных

Входные данные

В качестве входных данных используются файлы геномов в формате FASTA (.fa), наиболее распространенном формате для хранения и обработке геномов:

```
>reference
GATGAGCATGTGTTGAATCCTCA
>test_input_1
GATGAGCATGTGTTGAATCCTCAGATGAGCATGTGTTGAATCCTCA
>test_input_2
  GTCCTCCAGTTTGTAGCATGTGTATTTATTAATTCTGTCATGTGCAAATGGTTGTCCTGCATATAAAAG
  TATCTGGTCATTTCAGTTTGGTTTGTA
```

***Вывод*:**

```
>test_input_1
1 23 1 23
1 23 24 46
>test_input_2
5 13 15 23
```

Пример вызова программы

```
./run 8 22 refseq.txt input1.txt input2.txt …
```

где 8 – количество потоков, разрешенных создавать программе, 22 – минимальная длина совпадающих подстрок (minMatchLength).

1.2. Алгоритм решения задачи

Последовательный алгоритм

Классическое решение проблемы – построение суффиксных структур, описанных в книге Gusfield D. Algorithms on strings, trees and sequences [7]. Мы не будем их здесь разбирать.

Нашим решением было использование модификации алгоритма Рабина–Карпа, позволяющего достичь тех же времен, что и при построении суффиксных структур. Алгоритм в большей степени ориентирован на **поиск большого количества коротких**

паттернов. Это как раз тот случай, когда имеется много паттернов поиска и необходимо производить поиск сразу всех паттернов.

Идея алгоритма Раббина–Карпа: заменим поиск совпадающих подстрок на поиск совпадений хэш-функций от подстрок, значения которых можно вычислять за линейное время.

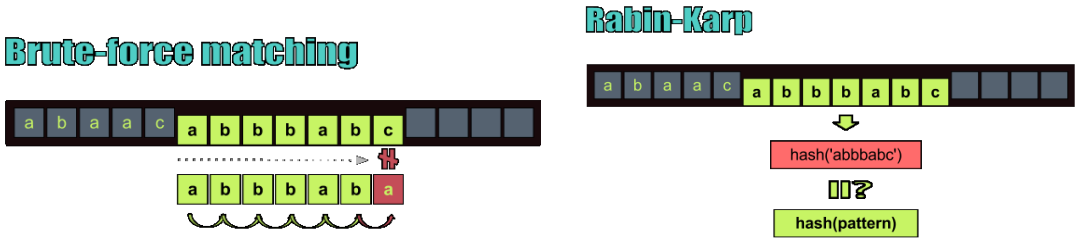


Рис. 3. Идея алгоритма Рабина–Карпа – заменить сравнение подстрок сравнением хэш-функции от подстрок, которую можно вычислять быстро

В качестве хэш-функции была выбрана кольцевая хэш-функция из работы [1], полностью отображающая подстроки на значения хэш-функции, таким образом, *мы избегаем сравнения подстрок между собой*:

$$f(substr) := \sum_{r=1}^{m} f(substr[r]) \cdot 4^r, \qquad (1)$$

$$f(A) := 1,\ f(G) := 2,\ f(T) := 3,\ f(C) := 4.$$

Вычисление хэша следующей подстроки, использующего хэш подстроки, сдвинутой на 1 символ назад, основан на свойствах функции f и действует как «скользящее окно» – достаточно вычесть значение первого элемента (слева), поделить оставшееся на 4 и прибавить значение нового элемента (справа), умноженное на 4^m, т.е. окно как бы двигается как целое слева направо и вычисление хэш значения следующей подстроки из хэш значения текущей подстроки вычисляется за O(1).

Выбор $f(A) := 1$ не нулем, а единицей, сделан для различения подстрок A и AAAAAA.

Таким образом, поиск сводится к построению хэш-таблицы от *input* строк с парами – ключ (значение хэш-функции от подстроки) и положение её первого элемента, – и быстрым вычислением благодаря последовательному сдвигу хэш-функции от подстрок *reference* строки («эталонного» генома) с поиском вычисленного значения в хэш-таблице. Нахождение значения в хэш-таблице означает обнаружение совпадающей подстроки длин *m*. Найденные совпадения записываются в хэш-таблицу результатов и фильтруются после прохождения основного этапа для отбора как возможно более длинных совпадающих подстрок.

Пример вычисления хэш-функции от первого сравниваемого генома *test_input_1* (“TTGATCTT” → “22312422”) для *minMatchlength*=3:

“223” → $2*4^1+2*4^2+3*4^3 = 8 + 32 + 192 = 232$

“231” → $(232 – 2*4^1)/4 + 1*4^3 = 56 + 64 = 120$ и т.д.

Keys, f(substr)	I, index of substr in ref.string
120	2
176	6
...	...
192	5
232	1

Рис. 4. Хэш-таблица подстрок первого сравниваемого генома

Пробег по значениям хэш-функции на подстроках reference строки с поиском каждой в хэш-таблице находит все совпадающие подстроки длиной *m* равной 31, если minMatchlength больше 31, и самому minMatchlength, если меньше. Ограничения на максимальное значение *m* в виде 31 определяется из пределов значений целого типа unsigned long long, в которых мы можем хранить значения хэш-функции.

Использование хэш-таблиц предоставляет следующие преимущества: хранение только заполненных данных, быстрота обращения к ячейке (по сравнению с деревьями).

Результатом работы алгоритма будет хэш-таблица со значениями положений совпадающих подстрок длиной *m* в исходных строках.

2. Параллелизация, улучшения алгоритма и результаты

Если ***Длина максимального инпута > Длина reference строки,*** то меняем ролями *reference* и *input* строки – хэш-таблицу строим от *reference* строки, а пробегаемся по инпутам.

Это более медленная операция, т.е. сводя все хэш-таблицы от разных инпутов в одну с сортировкой по хэшу, мы существенно экономим время поиска.

Параллелизация по стратегии «Разделяй и властвуй» "***Divide and conqueror***". Каждому потоку отдается часть reference строки в которой он запускает последовательный алгоритм. Достаточно поделить поиск подстрок в *reference* (или *input*) строке между потоками, они абсолютно независимы после заполнения хэш-таблицы инпутов (*reference* строки). Пример для случая двух потоков показан на рис. 5.

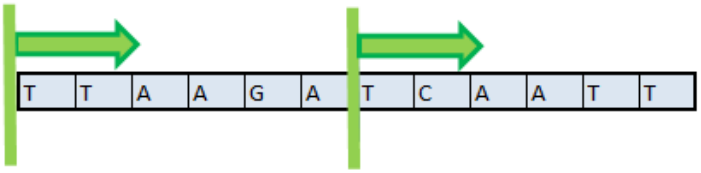


Рис. 5. Стратегия «Разделяй и властвуй» для случая двух потоков

3. Результаты

В результате проведенных тестов замеры времени работы программ показали (табл. 1), что в среднем *ускорение работы алгоритма* с реализацией хэш-таблицы в виде STL *map* (естественно, не лучшем) составляет 19 *раз*.

Таблица 1. Ускорение алгоритма по сравнению с алгоритмом «грубой силы»

Тест	Количество потоков	Алгоритм «грубой силы»	Хэш алгоритм	Ускорение
refseq_3 (5 kb) vs input_3 (200 Kb)	1	22.3 sec	1.197 sec	в 18.8 раз

4. Масштабируемость

Использование реализации хэш-таблицы в виде *map* структуры библиотеки STL не позволило добиться максимальной масштабируемости в силу многопотоковой небезопасности структуры. Для улучшения масшатабируемости рекомендуется использовать хэш-таблицы библиотеки Intel Threading Building Blocks (TBB) или Google SparseHash, DenseHash.

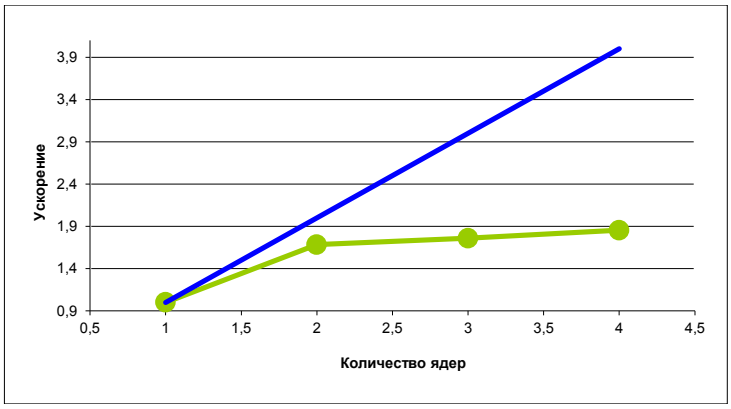


Рис. 6. Core i7 (total : 4 cores) with 8Gb of ram and a 500Go Hard Drive

5. Заключение

Преимуществами алгоритма являются простота реализации и хорошая эффективность, особенно для экстремальных задач – длина reference или одной из input строк много больше, чем длины остальных строк. *Сравнение большого количества коротких геномов с большим «эталонным» – наилучший случай для оптимальной скорости работы алгоритма*. Существует большое количество эффективных и по скорости, и по памяти хэш-таблиц для эффективной реализации в коде. Также представленный алгоритм прост в распараллеливании.

Литература

1. Altschul S., Gish W., Miller W., Myers E., Lipman D. Basic local alignment search tool // Journal of Molecular Biology. 1990. 215. P. 403-410.
2. Reneker J., Shyu C.-R. Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals // BMC Bioinformatics. 2005. 6: 111 – [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1131890/].
3. Ning Z., Cox A., Mullikin J. SSAHA – A Fast Search Method for Large DNA Databases // Genome Research. 2001. 11: P. 1725-1729.
4. Ning Z., Cox A., Mullikin J. SSAHA – A Fast Search Method for Large DNA Databases // Genome Research. 2001. 11: P. 1725-1729 – [http://www.biomedcentral.com/pubmed/15215469].
5. Reneker J., Lyons E. et al. Long identical multispecies elements in plant and animal genomes // Proc Natl Acad Sci USA. 2012. May 8;109(19) – [http://www.ncbi.nlm.nih.gov/pubmed/22496592].
6. Accelerate 2012 Parallel Programming Contest – [http://software.intel.com/ru-ru/articles/contest-accelerate-2012-problem/].
7. Gusfield D. Algorithms on Strings, Trees and Sequences. Computer Science and Computational Biology. Cambridge, UK: Cambridge University Press, 1997.

ОБ ОДНОМ ПОДХОДЕ К ПРЕДСТАВЛЕНИЮ ДАННЫХ В СИСТЕМЕ «ВИЗОГРАФ»

Е.А. Козинов

Нижегородский госуниверситет им. Н.И. Лобачевского

Разработка параллельных программ является сложной задачей. Одно из активно развиваемых направлений, уменьшающих сложность, – создание сред визуального проектирования и разработки параллельных программ. В рамках статьи рассматривается одна из таких систем – «Визограф». Для системы рассматривается один из подходов к описанию типов данных, позволяющий контролировать как синтаксическую, так и логическую правильность получаемых визуальных схем.

Введение

На протяжении многих лет одной из наиболее острых проблем эффективного использования потенциала компьютерных технологий является трудоемкость разработки программного обеспечения (ПО). Широкий спектр практических приложений, для решения которых разрабатываются программы, разнообразие компьютерных систем и их непрерывное совершенствование, существенное повышение сложности применяемых алгоритмов обработки – все это вместе взятое чрезвычайно затрудняет разработку ПО. В отличие от компьютерного оборудования, производительность которого практически удваивается каждые 18 месяцев (широко известное утверждение Мура), технологии разработки программ совершенствуются крайне медленно – к сожалению, закон Мура для ПО не действует.

Для параллельных вычислительных систем проблема трудоемкости разработки становится критической. Дополнительные сложности вызывает относительная новизна области параллельного программирования, наличие качественно новых эффектов при выполнении вычислительных процессов (тупики, гонки потоков, избыточная синхронизация и т.п.). Особая острота проблемы связана с необходимостью оперативной массовой разработки параллельных программ вследствие коренного перехода компьютерной техники на использование многоядерных процессоров.

Одно из активно развиваемых направлений, уменьшающих сложность, – создание сред визуального проектирования и разработки параллельных программ [1]. Смысл подхода заключается в том, что разрабатываемый алгоритм разделяется на независимые части. Каждая часть представляется в виде компонента или модуля. Для получения параллельного алгоритма модули компонуются по средствам визуальных схем.

Визуальная система «Визограф»

В Нижегородском государственном университете им. Н.И. Лобачевского разрабатывается новая среда визуального проектирования и разработки параллельных программ «Визограф» [2]. Основные усилия направлены на решение следующих основных задач:

1. Поиск и разработка методов улучшения визуального (наглядного) представления разрабатываемых параллельных алгоритмов и программ.
2. Расширение визуального подхода средствами автоматической верификации и контроля правильности построенных вычислительных схем.
3. Разработка – наряду с возможностями «ручного» конструирования визуальных схем – автоматических или полуавтоматических средств поддержки пользователя в разработке алгоритмов для решения поставленной задачи.

В разрабатываемой системе для представления алгоритмов используются ряд визуальных элементов. Основными элементами являются модули, параметры и линкеры. Каждый модуль соответствует некоторой реализации определенного алгоритма или его части и может быть представлен, например, в виде программного модуля dll-библиотеки. Параметры используются для визуального представления входных и выходных данных модулей. Линкеры применяются для установления соответствия (связывания) между параметрами модулей. Подобные элементы встречаются в большинстве визуальных систем разработки параллельных программ, использующих визуальные элементы для отображения потока данных [1]. Примеры модулей можно увидеть на рис. 1, блоки 2 и 6.

Одно из основных назначений визуальных схем – упрощение процесса разработки алгоритма. Если модулей, выделенных в алгоритме, много, то за счет большого количества связей визуальная схема становится сложной для восприятия. Для упрощения представления алгоритма, как правило, применяется иерархическое представление. В системе «Визограф» иерархическое представление обеспечивается путем введения макромодулей (см. рис. 1, блок 1).

Для повышения компактности изображения алгоритмов в системе «Визограф» дополнительно введены новые элементы «сплиттер» и «агрегатор». Благодаря этим элементам, данные можно разделить на части для последующей параллельной обработки и объединить результаты вычислений. Пример сплиттера и агрегатора можно увидеть на рис. 1, блоки 3 и 5.

Рис. 1. Основные элементы системы «Визограф» на примере алгоритма умножения матрицы на вектор

Для повышения контроля над данными в процессе исполнения в визуальной системе «Визограф» введены визуальные элементы обозначающие параметры связности. На рис. 1. благодаря введенным элементам, отображено и контролируется одно из условий – для перемножения матрицы на вектор порядок матрицы A должен совпадать с размером вектора b.

Более подробную информацию о визуальных элементах, используемых во время разработки алгоритмов, и методах их применения можно получить в [2]. Далее рассматривается один из возможных методов представления данных в системе «Визограф».

Представление типов данных в системе «Визограф»

В разрабатываемой системе «Визограф» предполагается, что схема корректна, если выполнен ряд условий. Одно из таких условий: все параметры модулей должны быть соединены с помощью линкеров (циклические связи при этом не допускаются).

Для обеспечения синтаксической корректности построенных схем недостаточно того, чтобы все параметры в построенной схеме были соединены посредством линкеров. Введенные параметры связности позволяют лишь частично проверить логическую правильность построенных схем. Основное же их назначение – помочь в отладке получаемых параллельных программ в процессе исполнения. Для обеспечения синтаксической корректности построенной схемы необходимо, чтобы параметры, соединенные линкерами, содержали данные одного типа. Для представления типов данных в системе «Визограф» введен ряд дополнительных визуальных элементов.

На начальном этапе использования системы «Визограф» пользователю доступны только *базовые типы данных*. Предполагается, что параметры модулей могут быть сопоставлены только с данными типа матриц, векторов или скаляров. Сами базовые типы данных могут содержать элементы разных типов данных, таких как целые (integer), вещественные (double, float) и др. Для обозначения типов данных используются прямоугольники с пиктограммами в правом верхнем углу. Пиктограмма визуально обозначает используемый базовый тип данных. Внутри прямоугольников явно указывается тип элементов. Пример базового типа данных – матрица вещественного типа – приведен на рис. 2, блок 1.

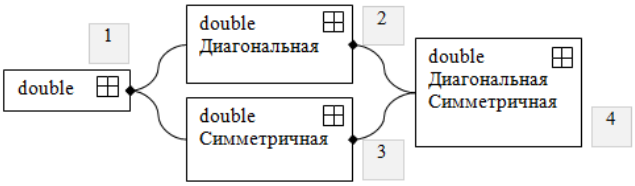


Рис. 2. Представление типов данных в системе «Визограф»

При разработке алгоритмов часто не хватает информации о том, что на вход модуля поступили данные того или иного типа. Для выполнения многих алгоритмов необходимо, что бы данные обладали определенным набором свойств. Например, разрабатываемый алгоритм может быть рассчитан на то, что используется матрица диагональная или симметричная. Пример таких задач легко найти, например, среди методов решения систем линейных уравнений.

Для разработки сложных алгоритмов необходимо иметь возможность включения автоматической проверки корректности принимаемых и получаемых данных. Подобная проверка позволяет сконцентрировать усилия на разработке самого алгоритма, а не проверке входных данных.

Для проверки корректности данных в системе «Визограф» введены *производные типы данных*. Производный тип данных может быть получен путем добавления свойств из системы «Визограф» к существующему типу данных.

Необходимые свойства могут быть добавлены в систему «Визограф» благодаря разработке макромодуля, удовлетворяющего ряду требований. Данный макромодуль через входящий параметр должен принимать базовый или уже существующий тип данных. Через исходящий параметр макромодуль в виде скаляра, содержащего логическое значение, должен возвращать результат проверки свойства. Любые свойства, реализуемые пользователем, сохраняются в системе и могут быть повторно использованы.

Во время исполнения алгоритма можно включить проверку свойств. В этом случае при получении и передаче данных для каждого параметра выполняется запуск алгоритмов проверки свойств. Алгоритмы проверки свойств вызываются из макромодулей и могут исполняться параллельно.

Изображение производных типов похоже на изображение базовых типов. В изображение, кроме типа элементов, добавляется список свойств. Пример определения двух производных типов от матрицы изображен на рис. 2, блоки 2 и 3.

В некоторых случаях необходимо получить тип данных путем агрегации свойств уже существующих типов данных. Для предоставления такой возможности в системе «Визограф» поддерживается агрегация типов данных. Свойства агрегируемых типов данных объединяются. Пример агрегированного типа данных изображен на рис. 2, блок 4.

Логическое описание типов данных в системе «Визограф»

При разработке алгоритмов совпадение типов данных параметров, соединенных посредством линкеров, не гарантирует правильность работы всего алгоритма в целом. Корректная визуальная схема с точки зрения синтаксиса не гарантирует корректности работы алгоритма с логической точки зрения. Логику алгоритма нельзя описать только физическими свойствами параметров. Для описания логики нужны дополнительные метасвойства, поясняющие назначение параметров. В качестве таких метасвойств в системе «Визограф» используются *дескрипторы*.

Дескрипторы добавляются в типы данных так же, как и свойства. Все дескрипторы отображаются в списке свойств. Для отделения физических свойств от дескрипторов типов данных дескрипторы отображаются после вертикальной черты. При агрегации типов данных множество дескрипторов, заключенных в типах данных, также объединяется. Дескрипторы являются глобальными для системы «Визограф» и могут быть повторно использованы.

На рис. 3, блок 1, представлен пример определенного типа данных – матрица вещественного типа, используемая в операциях умножения матрицы. Дополнительные примеры использования дескрипторов для описания логического назначения определяемых типов данных представлены на рис. 3, блоки 2 и 3.

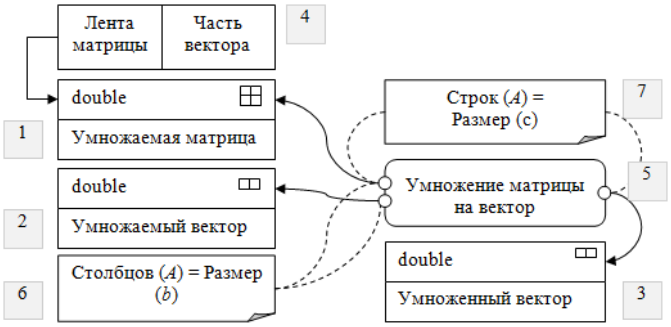


Рис. 3. Пример регистрации модуля в системе «Визограф»

Не редка ситуация, когда однажды разработанный модуль может быть применен для разработки частей алгоритмов, использующих разные с логической точки зрения наборы параметров, но физически реализующий один и тот же алгоритм. Пример такого модуля – умножение матрицы на вектор. Алгоритм умножения матрицы на вектор не изменяется ни в случае, когда умножается матрица на вектор, ни в случае, когда умножается лента матрицы на вектор. Отличие состоит в том, что при умножении ленты матрицы на вектор в результате получается только часть результирующего вектора.

Для описания логических альтернатив применения типов данных в системе «Визограф» вводится понятие *дерево альтернативных дескрипторов*. Если для типа данных есть возможность добавить дополнительные логические описания, которые расширяют возможность применения типа данных, то это изображается в виде прямоугольника, разделенного на две части вертикальной линией. В левой части прямоугольника перечисляются дескрипторы, добавляемые к типу данных. В правой части перечисляются дескрипторы, которые автоматически добавятся к каждому типу данных, соответствующему исходящему параметру модуля, использующему тип данных с учетом расширения. Для связывания подобных структур используются линкеры. Иерархия альтерна-

тивных дескрипторов в системе «Визограф» не ограничена. Каждый новый уровень добавляет логические свойства к получаемому типу данных. Пример альтернативного дескриптора изображен на рис. 3, блок 4.

В системе «Визограф» присутствуют особые элементы – сплиттеры и агрегаторы. Для поддержания логической связности схем слиттеры и агрегаторы автоматически добавляют и удаляют дескрипторы, связанные с разделением или объединением данных.

Введенное логическое описание, добавляемое к типам данных, закладывает основы для автоматического построения схем. Благодаря дескрипторам, пользователь системы может сформулировать решаемую задачу и предоставить системе возможность помочь найти решение в виде схемы алгоритма.

Регистрация модулей в системе «Визограф»

Базовые типы данных, свойства и дескрипторы позволяют сконструировать необходимые типы данных. Далее построенные типы данных используются для регистрации модулей в системе.

Общий процесс регистрации модуля в системе проходит в несколько этапов. Вначале пользователь системы конструирует необходимые или выбирает наиболее подходящие среди существующих типы данных. Затем инструментальными средствами создается пустой модуль. При создании модуля ему задается имя. Далее к модулю добавляется необходимое количество входящих и исходящих параметров. Для каждого параметра задается имя. Затем параметры модуля связываются с типами данных, содержащими логическое и синтаксическое описания. Выполнив данную последовательность действий, модуль можно зарегистрировать в системе. На рис. 3, блок 5, представлен зарегистрированный модуль умножения матрицы на вектор.

Важно заметить, что при регистрации модулей для параметров можно указать параметры связности. Указание параметров связности позволяет снизить вероятность появления ошибок на ранних этапах разработки. На рис. 3, блоки 6 и 7, показан пример добавления параметров связности.

Зарегистрированные модули могут быть использованы для разработки алгоритмов. При связывании модулей контролируется логическая и синтаксическая правильность построенных схем.

Заключение

В Нижегородском государственном университете им. Н.И. Лобачевского разрабатывается новая среда визуального проектирования и разработки параллельных программ «Визограф». В рамках статьи рассматривается один из подходов к описанию типов данных, используемых в системе. Представленное описание позволяет осуществлять автоматическую верификацию и контроль правильности построенных вычислительных схем. Также представленное описание потенциально может быть применено для автоматической или полуавтоматической поддержки пользователя в разработке алгоритмов для решения поставленных задач.

Литература

1. Гергель В.П., Козинов Е.А. Обзор визуальных систем разработки параллельных программ // Вестник ННГУ им. Н.И. Лобачевского. [принята к печати]
2. Гергель В.П., Козинов Е.А. Об одном подходе к визуальной разработке программ // Вестник ННГУ им. Н.И. Лобачевского. 2012. № 4. С. 247-252.

АЛГОРИТМ ПОИСКА НАИБОЛЬШЕГО ЗНАЧЕНИЯ В КЛАССЕ ФУНКЦИЙ, ОПРЕДЕЛЯЕМОМ КУСОЧНО-ЛИНЕЙНОЙ МАЖОРАНТОЙ

А.Г. Коротченко, В.М. Сморякова

Нижегородский госуниверситет им. Н.И. Лобачевского

Вопросы, связанные с исследованием эффективности численных методов оптимизации, представляют значительный интерес как с теоретической, так и с практической точек зрения. С теоретической точки зрения такие исследования интересны выяснением потенциальных возможностей алгоритмов оптимизации на рассматриваемых классах задач. С точки зрения практики использование эффективных методов в качестве вспомогательных процедур является важным при решении сложных задач, требующих большого объёма вычислений или проведения дорогостоящих экспериментов.

В рамках построения эффективных методов оптимизации большое значение приобретает рассмотрение классов функций одной переменной, что объясняется следующими обстоятельствами.

Во-первых, такие методы используются как вспомогательные процедуры во многих алгоритмах поиска экстремума функций многих переменных.

Во-вторых, во многих задачах оптимального проектирования система ограничений, описывающих требования, которые накладываются на характеристики проектируемого устройства, может содержать характеристики, зависящие как от вектора варьируемых параметров, так и от параметра, значения которого принадлежат некоторому отрезку. Таким параметром, например, могут быть частота, время, температура и т.п.

Эффективность алгоритма во многом определяется классом рассматриваемых задач.

В работе [1] построены оптимальный одношаговый и приближенно оптимальный алгоритмы для классов функций, определяемых кусочно-линейной мажорантой.

В работах [2] и [3] построены приближенно оптимальные алгоритмы в классах функций, определяемых кусочно-степенными мажорантами.

В работе рассматриваются подклассы класса функций, определяемых кусочно-линейной мажорантой, позволяющие построить простой и достаточно эффективный алгоритм поиска их наибольшего значения.

Будем говорить, что непрерывная функция $f(x)$, определённая на отрезке $[a, b]$, принадлежит классу функций $F_1(a,b,K_1,K_2)$, если выполняются следующие соотношения:

$$\frac{f(x_2)-K_1}{x_2-a} \le \frac{f(x_1)-K_1}{x_1-a}, \quad \frac{f(x_2)-K_2}{b-x_2} \ge \frac{f(x_1)-K_2}{b-x_1}, \tag{1}$$

где $x_1 > x_2$, $x_1, x_2 \in [a, b]$, $K_1, K_2 \in R$, $K_1 > K_2$.

Непрерывная функция $f(x)$, определённая на отрезке $[a, b]$, принадлежит классу функций $F_2(a,b,K_1,K_2)$, если выполняются соотношения (1) при $K_2 > K_1$.

При $K_1 = K_2 = K$ получаем класс функций $F(a,b,K)$, рассмотренный ранее в [1].

Для классов $F_1(a,b,K_1,K_2)$ и $F_2(a,b,K_1,K_2)$ выполняются следующие свойства: если функция $f(x) \in F_1(a,b,K_1,K_2)$ ($F_2(a,b,K_1,K_2)$), то $f(x) \geq \min(K_1, K_2)$, класс функций $F_1(a,b,K_1,K_2)$ ($F_2(a,b,K_1,K_2)$) замкнут относительно операции взятия максимума, минимума и суммирования с неотрицательными коэффициентами. Данным классам принадлежат вогнутые, выпуклые и липшицевы функции.

Алгоритм основан на построении точной верхней мажоранты всех функций из класса $F_1(a,b,K_1,K_2)$ ($F_2(a,b,K_1,K_2)$), которые на фиксированном наборе точек принимают заданные значения. Очередная точка вычислений функции выбирается на основе концепции одношаговой оптимальности.

Построенный алгоритм был реализован в виде программной библиотеки. В рамках вычислительного эксперимента были проведены сравнения работы данного алгоритма и алгоритма, рассмотренного в [1]. В качестве тестовых функций выступали кусочно-линейные функции, кусочно-квадратичные и кусочно-вогнутые функции. Результаты экспериментов показывают, что рассмотренный алгоритм обладает более высокой скоростью сходимости по сравнению с алгоритмом, рассмотренным в [1].

Литература

1. Коротченко А.Г. Об одном алгоритме поиска наибольшего значения одномерных функций // Журнал вычислительной математики и математической физики. 1978. Т. 18, № 3. С. 563-573.

2. Коротченко А.Г. О приближенно-оптимальных алгоритмах поиска экстремума в одном классе функций // Журнал вычислительной математики и математической физики. 1990. Т. 30, № 3. С. 355-365.

3. Коротченко А.Г. Приближенно-оптимальный алгоритм поиска экстремума одного класса функций // Журнал вычислительной математики и математической физики. 1996. Т. 36, № 5. С. 30-39.

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ФОРМИРОВАНИЯ И РЕШЕНИЯ ДОПОЛНЕНИЯ ШУРА

С.П. Копысов, И.М. Кузьмин, Н.С. Недожогин, А.К. Новиков

Институт механики УрО РАН, Ижевск

Введение

Впервые метод дополнения Шура был математически сформулирован и описан в работах Issai Shur, а название своё получил после выхода работы [1], хотя до этого метод был известен в механике как метод подструктур и впервые встречается в работе [2]. В России этот метод известен, как метод суперэлементов [3]. Изначально он рассматривался как метод блочной декомпозиции для вычислений на системах с ограниченными вычислительными мощностями. В дальнейшем использовался для построения методов декомпозиции области без перекрытий на параллельных вычислительных системах. В настоящее время его все чаще рассматривают как гибридный решатель систем линейных алгебраических уравнений [4], сочетающий как прямые, так и итерационные методы для операций над матрицами, и который учитывает различные архитектуры вычислительных систем. В основе метода дополнения Шура лежит идея независимого расчета отдельных подобластей и последующего учета их взаимодействия. В данном методе можно реализовать два уровня распараллеливания: первый уровень связан с независимыми вычислениями между подобластями [5, 6], второй – с параллельной реализацией методов, которые используются для формирования внутри отдельной подобласти. Помимо этого, распараллеливанию поддается и решение интерфейсной системы. В представленной работе рассматривается второй уровень распараллеливания, для которого предложены алгоритмы формирования матрицы дополнения Шура, а также решение интерфейсной системы уравнений с использованием технологии CUDA и нескольких графических ускорителей (GPU). Был рассмотрен метод сопряженных градиентов с декомпозицией матрицы, направленной на минимизацию обмена данных между потоками.

1. Метод дополнения Шура

Пусть область Ω разбита на n_Ω непересекающихся подобластей:

$$\Omega = \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_{n_\Omega}, \text{ где } \Omega_i \cap \Omega_j = \varnothing,\ \Gamma_B = \bigcup_{i=1}^{n_\Omega} \partial\Omega_i / \partial\Omega. \quad (1)$$

Разделение на подобласти (подструктуры) наследуется от процесса разделения дуального графа расчетной сетки $G(V,E) = \bigcup_{k=1}^{n_\Omega} G_k(V,E)$, здесь множество вершин графа V_G – это множество конечных элементов расчетной сетки, множество ребер графа E_G – множество смежных конечных элементов. Множество конечных элементов, образующих подструктуру – $V_{G_k} \subset V_G$.

Узлы расчетной сетки образуют множество $\hat{V}$ и условно разделяются на внешние $\hat{V}_{B_k}$ (принадлежат границе области) и внутренние $\hat{V}_{I_k}$, связанные с узлами подобласти

сетки, соответствующие подграфу $G_k(V,E)$. Из множества внешних узлов выделяются интерфейсные $\hat{V}_{C_k} \subset \hat{V}_{B_k}$, связанные с узлами из других подобластей.

Для каждой подобласти Ω_i строятся системы уравнений, причем степени свободы, связанные с внутренними и внешними (граничными) узлами, разделяются:

$$\begin{pmatrix} A_{II}^i & A_{IB}^i \\ A_{BI}^i & A_{BB}^i \end{pmatrix} \begin{pmatrix} u_I^i \\ u_B^i \end{pmatrix} = \begin{pmatrix} f_I^i \\ f_B^i \end{pmatrix}, \qquad (2)$$

где индексы I, B относятся к внутренним и граничным степеням свободы соответственно.

В методе дополнения Шура полагается, что все подграфы $G_k(V,E)$ связные, в противном случае система уравнений (2) для подобласти Ω_k распадается на несвязанные системы уравнений.

Система для общих граничных узлов определяется как

$$S_{BB} u_{\tilde{B}} = f_{\tilde{B}}, \quad S_{BB} = \sum_i^{n_\Omega} A_{BB}^i - A_{BI}^i A_{II}^{i\,-1} A_{IB}^{\,i}, \quad f_{\tilde{B}} = f_B - A_{BI} A_{II}^{-1} f_I \,, \qquad (3)$$

здесь S_{BB} – матрица граничных жесткостей или дополнение Шура для подобласти i, вектор $f_{\tilde{B}}$ – вектор правых частей. Матрицы S_{BB}, A_{II} положительно определены и симметричны. Отметим, что порядок матрицы S_{BB} значительно меньше, чем исходной матрицы, но может быть все-таки большим для ее обращения и хранения, поэтому для системы с дополнением Шура необходимо применять итерационные методы решения и распределённые схемы хранения.

2. Формирование матрицы дополнения Шура на нескольких GPU

Одной из самых затратных операций формирования дополнения Шура является обращение матрицы A_{II}. Обычно на практике методами нахождения обратной матрицы являются плохо распараллеливаемые прямые методы, например метод обращения на основе LL^T-разложения. Для последовательного варианта использовалась объектно-ориентированная реализация метода декомпозиции [6] в рамках пакета программ FEStudio [7], в котором операции над матрицами реализованы в формате DCSR (Dynamic Compressed Storage Row – динамический сжатый построчный формат хранения). В этом случае матрица хранится в виде двух списков AV и AVC. Элементами каждого списка являются массивы, которые хранят значения и номера столбцов ненулевых элементов исходной матрицы (AV – значения, AVC – номера столбцов). При этом i-й элемент списка хранит информацию, взятую из i-й строки матрицы. Однако ограничения технологии CUDA не позволяют в полной мере использовать объектно-ориентированный подход, в параллельном варианте данные удобнее хранить небольшим числом массивов, и в этом случае применим формат CSR (Compressed Storage Row) – сжатый построчный формат хранения. При использовании такого формата матрица хранится в виде трех массивов: AV – массив значения ненулевых элементов, AVC – массив номеров столбцов ненулевых элементов и AVN – массив номеров из AV, с которых начинается новая строка исходной матрицы. Отметим, что затраты на перевод матрицы из DCSR в CSR несущественны, что позволяет матричные операции на CPU выполнять в DCSR формате, а CSR формат использовать для вычислений на GPU.

Рассматриваемый в работе алгоритм вычисления обратной матрицы для параллельной реализации заключается в решении матричной системы вида $A_{II} X = E$, где E – единичная матрица $n_I \times n_I$. Система решается на GPU предобусловленным алгоритмом сопря-

женных градиентов (см. раздел 3). Если в правой части системы вместо матрицы E подставить A_{IB}, то её решением будет матрица $A'_{IB} = A_{II}^{-1} A_{IB}$. Такое представление позволяет заменить операции обращения матрицы и матричного произведения на решение n_B систем, каждая из которых решается независимо, что позволяет использовать одновременно несколько GPU. Дополнение Шура или матрица граничных жесткостей вычисляется по соотношениям (3), где $A_{BB} \in R^{n_B \times n_B}$, $A_{II} \in R^{n_I \times n_I}$, $A_{BI} \in R^{n_B \times n_I}$, $A_{IB} \in R^{n_I \times n_B}$.

Пусть $\tilde{n}_B^i$ – число столбцов матрицы A_{BB}, пересылаемых на i-й GPU, $\tilde{A}_{BB}^i \in R^{n_B \times \tilde{n}_B^i}$ – матрица, состоящая из столбцов матрицы A_{BB} с номерами от $\sum_{j=0}^{i} \tilde{n}_B^j$ до $\sum_{j=0}^{i+1} \tilde{n}_B^j$, где i – номер GPU. Каждый графический ускоритель решает $\tilde{n}_B^i$ систем $A_{II} a^k = A_{IB}^k$, здесь $k \in \left[\sum_{j=0}^{i} \tilde{n}_B^j, \sum_{j=0}^{i+1} \tilde{n}_B^j\right]$, а $A'_{IB} = \{a^1, a^2, \ldots, a^{n_B}\}$. В реализации подхода независимого решения систем уравнений на нескольких графических ускорителях используется технология OpenMP. Для этого создаются несколько нитей (число которых равно числу доступных устройств), определяется номер нити и каждой назначается графический ускоритель с тем же номером.

Ниже представлен параллельный алгоритм формирования дополнения Шура на каждой подобласти Ω_i (индекс i опущен).

Решаем систему $A_{II} A'_{IB} = A_{IB}$ *(матрицы хранятся на GPU в CSR формате, решение – по столбцам)*;

$S_{BB} = A_{BB} - A_{BI} A'_{IB}$ *(S_{BB} и результат произведения матриц – построчно, A_{BB} – в CSR формате)*;

$\hat{f}_B = f_B - A_{BI} x$ *(x – решение системы $A_{II} x = f_I$)*;

Формируем и решаем: $S_{BB} u_B = \hat{f}_B$ *(S_{BB} формируется в DCSR на CPU, а затем копируется в CSR на GPU)*;

Определяем $u_I = x - A'_{IB} u_B$ *(x и A'_{IB} вычислены ранее и хранятся на CPU)*.

На каждом GPU после решения систем остаётся матрица $A'^i_{IB} \in R^{n_I \times \tilde{n}_B^i}$, состоящая из столбцов матрицы A'_{IB} с номерами от $\sum_{j=0}^{i} \tilde{n}_B^j$ до $\sum_{j=0}^{i+1} \tilde{n}_B^j$. Это позволяет без дополнительных коммуникаций выполнить оставшиеся операции (произведение и разность матриц, см. соотношения (3)) для каждой матрицы A'^i_{IB} независимо на нескольких GPU, которые используются при вычислении матриц S_{BB}^i. Далее формируется глобальная матрица дополнения Шура S_{BB} (четвертый шаг алгоритма), состоящая из локальных S_{BB}^i, принадлежащих i-й подобласти.

Локальные матрицы дополнения Шура S_{BB}^i хранятся в ленточном виде. Матрица S_{BB} после формирования из локальных S_{BB}^i преобразуется из ленточного к тому формату, в котором будет наиболее удобно с ней работать на GPU, например CSR.

В рассмотренных примерах максимальное ускорение при параллельном формировании по отношению к времени CPU составляет 1.87 и достигается при небольшом числе подобластей $n_\Omega = 16$ с использованием восьми GPU. С ростом числа подобластей размерности матриц уменьшаются, а вычисления на CPU становятся эффективней.

3. Решение интерфейсной системы уравнений на GPU

Отметим, что система для дополнения Шура является разреженной, но её заполненность на один-два порядка больше заполненности исходной конечно-элементной

системы. Для эффективного решения подобных задач на параллельных системах необходимо построение предобуславливателей, обладающих достаточным ресурсом параллелизма. Отметим, что многие предобуславливатели плохо распараллеливаются как на стадии построения, так и на стадии применения в итерационном алгоритме. В работе использовались методы сопряжённых градиентов с диагональным предобуславливателем (DIAG), предобуславливанием по методу симметричной последовательной верхней релаксации (SSOR) и предобуславливателем, построенным после масштабирования системы (DIP). Оптимальным выбором для вычислений на GPU будут предобуславливатели, в которых считается, что известна аппроксимация обратной матрицы системы. В этом случае дополнительные операции, вызванные переходом к предобусловленной системе, сводятся к матрично-векторному произведению. При его выполнении вектор хранится в текстурной памяти, которая кэшируется, что даёт более быстрый доступ и уменьшает временные затраты. Для вычисления каждой компоненты вектора результата используется от 2 до 32 потоков в зависимости от разреженности матрицы. Вычисление предобуславливателя выполняется либо на GPU, либо на CPU в зависимости от его типа.

Для решения интерфейсной системы (3) реализован блочный алгоритм сопряженных градиентов, использующий несколько GPU. Граф матрицы S_{BB} системы (3) разделяется на блоки по числу GPU с использованием методов из библиотеки [8]. Вершины графа делятся на две группы: внутренние и граничные (связанные хотя бы с одной вершиной другого блока). На основе такого разделения в каждом блоке формируется несколько матриц: S_I – матрица внутренних неизвестных; S_{IB} – матрица связи внутренних и граничных неизвестных; $S_B^{i,j}$ – матрица связи граничных неизвестных с j-м блоком, здесь i – номер блока. Тогда одна из самых затратных операции сопряженных градиентов, матрично-векторное произведение $q = S_{BB} p$, примет вид:

$$\begin{pmatrix} q_1 \\ q_B^1 \\ q_2 \\ q_B^2 \\ \vdots \\ q_N \\ q_B^N \end{pmatrix} = \begin{pmatrix} S_1 & S_{1B} & 0 & 0 & \cdots & 0 & 0 \\ S_{B1} & S_B^{1,1} & 0 & S_B^{1,2} & \cdots & 0 & S_B^{1,N} \\ 0 & 0 & S_2 & S_{2B} & \cdots & 0 & 0 \\ 0 & S_B^{2,1} & S_{B2} & S_B^{2,2} & \cdots & 0 & S_B^{2,N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & S_N & S_{NB} \\ 0 & S_B^{N,1} & 0 & S_B^{N,2} & \cdots & S_{BN} & S_B^{N,N} \end{pmatrix} \begin{pmatrix} p_1 \\ p_B^1 \\ p_2 \\ p_B^2 \\ \vdots \\ p_N \\ p_B^N \end{pmatrix}. \qquad (4)$$

В выражении (4) в каждом потоке вычисляются векторы:

$$q_B^{loc} = \sum_{j=1, j\neq i}^{j\leq N} S_B^{I,j} p_B^j, \qquad (5)$$

$$q_B^I = S_{BI} p_I + S_B^{II} p_B^I, \quad q_I = S_I p_I + S_{IB} p_B^I, \qquad (6)$$

$$q_B^I = q_B^I + q_B^{loc}, \qquad (7)$$

где I – номер потока. Вектор q_B^{loc} (5) вычисляется на CPU, остальные операции (6), (7) – на GPU. Перед выполнением операции с каждого GPU отправляются векторы p_B^j на CPU, которые при минимизации границ имеют размер гораздо меньший по сравнению с размерностью вектора p, где $j = 1,2,\ldots,N$. Обмен данными между GPU осуществляется асинхронно через память CPU. После выполнения (5) вектор q_B^{loc} пересылается на GPU и вычисляется выражение (7). Стоит отметить, что обмен векторами q_I и q_B^I между потоками не требуется. Векторные операции сложения и умножения на скаляр вы-

полняются независимо на каждом потоке отдельно для компонент, принадлежащих внутренним узлам и граничным. Скалярное произведение векторов вычисляется суммированием локальных произведений в глобальной переменной.

Решение интерфейсной СЛАУ предобусловленным методом сопряженных градиентов позволяет достичь ускорения на одном GPU по сравнению с CPU, равное 67, при разделении на 64 блока, а при решении системы с использованием восьми GPU ускорение увеличивается более чем в 4 раза. С увеличением числа блоков разделения до 1024 растет и ускорение, равное 93, однако использование нескольких GPU практически нецелесообразно при этом размере блоков. При этом использование различных предобуславливателей не дает различий по общему времени выполнения. Решение интерфейсной СЛАУ на тестовых задачах с одинаковой вычислительной нагрузкой на каждый GPU показало, что с ростом числа ускорителей алгоритм хорошо масштабируется.

Результаты, полученные в численных экспериментах, показывают, что при решении задач с помощью метода дополнения Шура оптимальный выбор алгоритма зависит от числа и размера подобластей, на которые делится расчётная сетка. Если в одной подобласти находится относительно небольшое число ячеек сетки (<5000), то для формирования матрицы дополнения Шура лучше использовать прямые методы нахождения обратных матриц и, как следствие, задействовать только CPU. При больших размерах подобластей наиболее эффективны алгоритмы с итерационными методами, использующие для вычислений несколько GPU. Решение же системы уравнений дополнений Шура эффективнее производить на GPU. В этом случае время решения сокращается в десятки и сотни раз.

Численные эксперименты проводились на гибридных кластерах ИМ УрО РАН и «Уран» ИММ УрО РАН.

Работа выполняется в рамках программы Президиума РАН №18 при поддержке УрО РАН (проект 12-П-1-1034) и гранта РФФИ №11-01-00275-а.

Литература

1. Haynsworth E.V. On the Shur complement // Basel Mathematical Notes. 1968. № 20. 17 p.
2. Przemieniecki J.S. Theory of Matrix Structural Analysis. New York: McGaw-Hill, 1968. 480 p.
3. Постнов В.А. Метод суперэлементов в расчетах инженерных сооружений. Л.: Судостроение, 1979. 288 с.
4. Gaidamour J., Henon P. A parallel direct/iterative solver based on a Shur complement approach // IEEE 11th International Conference on Computational Science and Engineering, San Paulo, Brazil, 2008. – P. 98–105.
5. Копысов С.П., Кузьмин И.М., Новиков А.К., Сагдеева Ю.А. Двухуровневое разделение области для параллельного метода подструктур // Актуальные проблемы математики, механики, информатики: Сб. статей ИМСС УрО РАН. Пермь, 2011. С. 69–76.
6. Копысов С.П., Красноперов И.В., Рычков В.Н. Объектно-ориентированный метод декомпозиции области // Вычислительные методы и программирование. 2003. Т. 4, № 1. С. 176–193.
7. Копысов С.П., Новиков А.К., Пономарёв А.Б., Рычков В.Н, Сагдеева Ю.А. Программная среда расчётных сеточных моделей для параллельных вычислений // Программные продукты и системы. 2008. № 2. С. 87–89.
8. Karypis G., Kumar V. 500 METIS – Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0. Tech. Report. 1995.

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ И ТЕХНОЛОГИИ РЕШЕНИЯ СОПРЯЖЕННЫХ ЗАДАЧ FSI

С.П. Копысов, И.М. Кузьмин, В.Н. Рычков, Л.Е. Тонков

Институт механики УрО РАН, Ижевск

Рассматривается подход к созданию универсальной открытой программной среды, предназначенной для связывания двумерных/трехмерных конечно-элементных и конечно-объемных методов, и ее дальнейшее использование в параллельных вычислениях.

Введение

При решении сопряженных задач взаимодействия потока газа и деформируемой конструкции (FSI) в рамках разделённого подхода [1] согласование получаемых решений на границах подобластей является неотъемлемым и весьма важным этапом, существенно влияющим на адекватность получаемых результатов. Под согласованием понимается прежде всего совокупность операций по обмену результатами вычислений между соответствующими решателями физических подзадач, интерполяции значений между узлами расчётных сеток, деформирование сетки газодинамической области и, возможно, реализация семейства итерационных процедур при жестком связывании.

Актуальным является построение параллельных алгоритмов решения отдельных физических подзадач динамики деформирования и газодинамики, а также непосредственно решающих всю совокупность задач сопряжения (интерполяция, деформирование расчетных сеток, итерационное связывание) и создание прикладного программного интерфейса с выделением самостоятельного приложения для синхронизации и обмена данными, в котором реализуются отдельные подзадачи связывания.

В данной работе основное внимание уделено построению эффективных параллельных алгоритмов для гибридных вычислительных систем при решении задачи динамики деформирования, интерполяции данных, деформирования сетки и обеспечения обмена данными между отдельными подзадачами.

1. Решение задачи динамики деформирования

Обычно в методе конечных элементов (МКЭ) считается, что конечно-элементная сетка собрана и каждый узел принадлежит нескольким элементам. Предположим обратное, множество элементов разобрано, так что любой узел принадлежит одному элементу, и нумерация степеней свободы в элементах выполнена от $1\ldots N_e$. Тогда поэлементная сборка эффективной матрицы жесткости $\hat{K}=\sum_{e=1}^{m} C_e^T\left(K_e+a_0M_e\right)C_e$ заменяется на $\hat{K}=C^T\tilde{K}C$ с блочно-диагональной матрицей $\tilde{K}$. Здесь $C=\left[C_e\right]_{1\times m}\in Z^{N_e\times N}$ – матрица связности, определяющая соответствие между степенями свободы элементов и независимыми перемещениями для всей области; $\hat{K}\in R^{N\times N}$ – глобальная эффективная матрица жесткости; $\tilde{K}_e\in R^{N_e\times N_e}$ – локальная (элементная) эффективная матрица жесткости, являющаяся блоком матрицы $\tilde{K}=\left[\tilde{K}_e\right]_{m\times m}$.

Таким образом, вычисления, выполняемые с матрицей $\hat{K}$, переносятся на матрицы $\tilde{K}_e$, что позволяет при перестроении системы уравнений, вызванном адаптацией сетки или локальным изменением степени аппроксимирующих полиномов, корректировать только матрицы перестраиваемых элементов и списки связности. Кроме того, применение методов подпространств Крылова без непосредственной сборки матрицы $\hat{K}$ позволяет эффективно решать системы уравнений на параллельных вычислительных системах [2, 3].

Дискретизация по времени основывается на схеме Ньюмарка. Выбирается временной шаг Δt, параметры α и δ, по которым вычисляются постоянные $a_0, a_1, \ldots, a_7$. Формируется эффективная матрица жесткости $\hat{K} : \hat{K} = K + a_0 M$. Для каждого временного шага $t + \Delta t$ вычисляется эффективная нагрузка $\hat{R}_{t+\Delta t} = R_{t+\Delta t} + M\left(a_0 u_t + a_2 \dot{u}_t + a_3 \ddot{u}_t\right)$, находятся перемещения из решения системы $\hat{K} u_{t+\Delta t} = \hat{R}_{t+\Delta t}$. Вычисляются ускорения и скорости: $\ddot{u}_{t+\Delta t} = a_0\left(u_{t+\Delta t} - u_t\right) - a_2 \dot{u}_t - a_3 \ddot{u}_t$; $\dot{u}_{t+\Delta t} = \dot{u}_t + a_6 \ddot{u}_t + a_7 \ddot{u}_{t+\Delta t}$.

Наиболее трудоемкая операция алгоритма Ньюмарка – матрично-векторное произведение, которое применяется как в основном алгоритме, так и на каждом шаге метода сопряженных градиентов для решения системы $\hat{K} u_{t+\Delta t} = \hat{R}_{t+\Delta t}$. Кроме того, в зависимости от формата хранения матриц жесткости и масс рассматриваются различные подходы к матрично-векторному произведению.

В элементной схеме метода сопряженных градиентов будем выполнять произведения $\hat{K} p = C^T \tilde{K} C p$ следующим образом: находим вектор $\bar{p} = Cp,\ p \in R^N$, вычисляем $\tilde{K}\bar{p}$, выполняем операцию $C^T \bar{p}$. Второй этап легко выполнить параллельно, распределив вычисления между ядрами вычислительной системы. При вычислениях на центральном процессоре (CPU) умножение на матрицу связности на первом и третьем этапах заменяется на косвенную адресацию к требуемым компонентам вектора. Для вычисления $\tilde{K}\bar{p}$ используются блоки $\tilde{K}$ и соответствующие компоненты $\bar{p}$, которые сгруппированы по конечным элементам, что позволяет помещать данные в кэш-память CPU, уменьшая общее время вычисления произведения $\tilde{K}\bar{p}$. В случае вычислений на графическом ускорителе (GPU) косвенная адресация на первом и третьем этапах может быть заменена на оптимизированный вариант матрично-векторного произведения, учитывающий разреженную структуру и значения элементов C.

В случае явной сборки матрицы $\hat{K}$ она хранилась в сжатом, упакованном по строкам формате. Исходя из того, что матрица $\hat{K} = \hat{K}_1 + \hat{K}_1^T - diag(\hat{K}_1)$ всегда получается симметричной, уместно хранить только половину матрицы для сокращения затрат памяти, например $\hat{K}_1$ – верхнюю треугольную матрицу, включая главную диагональ. Простейший вариант распараллеливания с помощью OpenMP содержал именно такое представление матрицы. Вычисление матрично-векторного произведения $\hat{K} p$ исходя из того, что хранится только половина матрицы (верхний треугольник, включая диагональ), выполнялось в два потока, в которых вычислялись произведения $\hat{K}_1 p$ и $\hat{K}_1^T p$ соответственно. Произведение $diag\left(\hat{K}_1\right)p$, а также операции сложения и вычитания результирующих векторов выполнялись последовательно.

Результаты вычислительного эксперимента показали, что использование GPU при решении системы уравнений дает ускорение от 40 до 120 раз в зависимости от геометрии конечного элемента и формата схемы хранения эффективной матрицы жесткости и матрицы масс.

2. Решение задач интерполяции и деформирования сетки

Для решения задач интерполяции и деформирования сетки применялся подход, связанный с использованием радиальных базисных функций [4], обладающий рядом преимуществ: возможность использовать движения отдельных точек, а не сетки (позволяет учитывать только данные об узлах без учета данных, связанных с сеткой); возможность рассмотрения систем меньшего размера при использовании компактных функций; высокая параллельная эффективность, связанная прежде всего с тем, что решение задачи сводится к решению системы уравнений, поиск которого можно достаточно эффективно распараллелить.

Основной идеей метода является поиск необходимого интерполянта в виде линейной комбинации, образованной радиальными базисными функциями. В этом случае интерполянт в виде линейной комбинации запишется следующим образом:

$$\omega_i(x) = \sum_{j=1}^{n_s} \lambda_j \phi\left(\| x - x_{s_j} \|\right) + q(x), \quad i = \{s, g\}, \quad \omega_i = \{u, pn\},$$

где $\omega_i = \{u, pn\}$ принимает значения перемещений u и давления pn на границе сопряжения; индексы s и g относятся к задаче деформирования и газовой динамики соответственно; коэффициенты λ_j и полином q определяются из условия $w_s(x_{s_j}) = W_{s_j}$ и требования $\sum_{j=1}^{n_s} \lambda_j s(x_{s_j}) = 0.$ Тогда система уравнений для неизвестных коэффициентов в матричной форме запишется в виде:

$$\begin{bmatrix} \Phi_{ss} & \mathrm{B}_s \\ \mathrm{B}_s^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} W_s^i \\ 0 \end{bmatrix}, \text{ где } \Phi_{ss} = \phi\left(\| x_{s_i} - x_{s_j} \|\right), \ \mathrm{B}_s = \left[1 x_{s_j} y_{s_j} z_{s_j}\right].$$

Для параметров на газодинамической сетке запишем

$$\begin{bmatrix} \Phi_{gs} & \mathrm{B}_g \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = W_g^i.$$

Кроме того, значения на газодинамической сетке могут быть получены и непосредственно из соотношения

$$W_g^i = \begin{bmatrix} \Phi_{sg} & \mathrm{B}_g \end{bmatrix} \begin{bmatrix} \Phi_{ss} & \mathrm{B}_s \\ \mathrm{B}_s^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} W_s^i \\ 0 \end{bmatrix}.$$

Аналогично для задачи деформирования газодинамической сетки построим интерполирующую функцию в виде линейной комбинации, образованной радиальными базисными функциями. Для внутренних узлов сетки в Ω_g приращения Δx_{Ω_g} находятся как $\Delta x_j(\Omega_g) = s(x_j(\Omega_g))$.

Таким образом, рассматриваемые выше задачи сводятся к решению систем линейных алгебраических уравнений с матрицами, разреженность и порядок которых зависят от выбора радиальных базисных функций. Такие системы можно эффективно решать предобусловленным методом сопряженных градиентов.

3. Структура и функции связывающего приложения

Предлагаемый подход к моделированию задач взаимодействия потока газа и деформируемой конструкции/тела основан на выделении этапа согласования и взаимодействий в отдельную программную подсистему, не зависящую от используемых реализаций расчетов, газодинамических полей и напряженно-деформированного состояния. Таким образом, может быть обеспечено взаимодействие для произвольных комбинаций программ решения задач газовой динамики и механики деформирования. В этом случае сопрягаются различные типы сеток, разные виды аппроксимаций. Точность аппроксимации уравнений сохраняется как в областях расчета газодинамики и механики деформирования, так и на границе взаимодействия.

В рассматриваемом случае система уравнений газовой динамики решается произвольным лагранжево-эйлеровым конечно-объемным методом в библиотеке OpenFOAM (http://www.openfoam.com), а на этапе решения задач деформирования применяется конечно-элементный пакет FEStudio [5]. Важно отметить, что оба пакета написаны в рамках объектно-ориентированного подхода и это существенно облегчает их совместное использование.

Приложение FEStudioFSI реализовано в соответствии с моделью «клиент-сервер» на объектно-ориентированном промежуточном ПО Ice Zeroc (http://www.zeroc.com/ice.html). Клиентами называются активные сущности, запрашивающие определенные сервисы у сервера. Серверами называются пассивные сущности, предоставляющие сервисы в ответ на запросы клиентов.

Клиент использует для вызова Ice-объекта так называемый прокси (proxy). Прокси – это представитель Ice-объекта на стороне клиента, в его локальном адресном пространстве. Код прокси для определенного языка программирования генерируется Slice-компилятором. Прокси инкапсулирует в себе информацию, необходимую для вызова Ice-объекта: физический адрес сервера, идентификатор объекта и, дополнительно, идентификатор фасета. Для работы распределенной модели FEStudioFSI клиент приложения должен содержать информацию о прокси двух серверов OpenFOAM и FEStudio, каждый прокси обладает уникальным именем, которое служит идентификатором при обращении к соответствующему серверу. Удаленный вызов методов, реализованных на стороне сервера, осуществляется посредством созданных прокси. Кроме того, клиент хранит и обновляет векторы перемещения и давления для их последующего использования в расчете и управляет логикой работы всей программы.

Со стороны серверов поведение Ice-объектов реализуется с помощью сервантов. Со стороны серверов реализован ряд методов, необходимых для распределенного решения задачи: расчет перемещений (сервант FEStudio), расчет давления (сервант OpenFOAM), передача полученных решений клиенту, получение новых данных о перемещении и давлении, необходимых для следующего шага решения.

На рис. 1 показана распределенная модель приложения FEStudioFSI. Она представляет собой объединение трех Ice-объектов: серверов OpenFOAM и FEStudio и клиента FEStudioFSI. В целом функционирование распределенной программы выглядит следующим образом. На процессорах запускаются серверные объекты OpenFOAM и FEStudio, в управляющем процессе создается клиентский объект FEStudioFSI, который содержит прокси к соответствующим сервантам. Во время запуска серверных объектов инициализируются переменные, необходимые для расчетов каждой из программ OpenFOAM и FEStudio, а также загружаются расчетные сетки. Далее, после инициализации, как только серверы сигнализируют о готовности, клиент запускает расчет и обмен данными, выполняя соответствующие запросы серверам. Кроме обеспечения взаимодействия двух серверов, клиент в рамках построенной модели занимается интерполяцией сетки. Время продолжительности расчета определяется клиентом и не зависит

от серверов. Описанный механизм связывания объектно-ориентированной модели и инфраструктуры Ice позволяет путем замены объектов получить распределенную объектно-ориентированную программу. Переход к распределенным/параллельным вычислениям осуществляется при минимальной модификации последовательного кода.

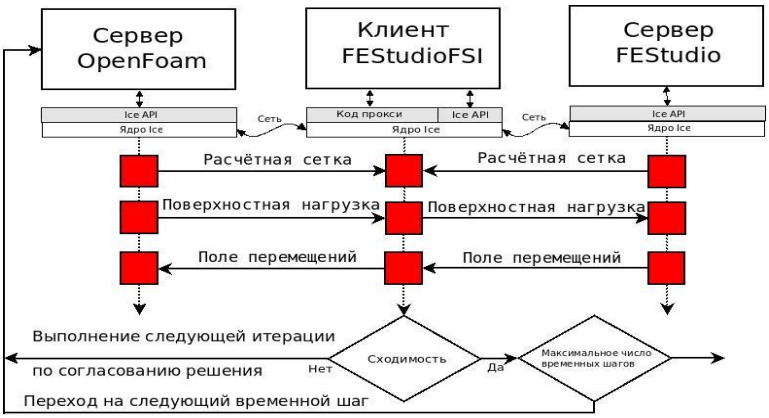


Рис. 1. Распределенная модель решения задачи FSI

Ключевые возможности разрабатываемого промежуточного ПО FEStudioFSI:

1. Возможность выбора алгоритмов для слабого связывания приложений (последовательные алгоритмы, обмен данными происходит на каждом временном шаге один раз) и сильного связывания (в пределах одного шага по времени реализуется итерационная процедура совместного решения уравнений с последовательной передачей нагрузок и перемещений).
2. Поддержка связывания различных независимых приложений посредством интерфейса прикладного уровня.
3. Поддержка различных методов переноса данных для несогласованных сеток, функций интерполяции и перераспределения данных, деформации расчетных сеток.

Разработанные алгоритмы и промежуточное ПО тестировалось на трехмерных задаче, описывающей математическую модель взаимодействия сверхзвукового потока газа и деформируемой панели в ударной трубе с различными условиями связывания и распределенного выполнения на гибридных кластерах. Выполненное сравнение результатов численного моделирования с экспериментальными данными позволяет заключить, что предложенные модели и алгоритмы адекватно описывают происходящие физические процессы как в газе и деформированной панели, так и при их взаимном влиянии.

Работа выполнена при поддержке РФФИ (проект 11-01-00275-а, 12-07-00080-а) и программы Президиума РАН N 18 при поддержке УрО РАН проект 12-П-1-1034.

Литература

1. Farhat C., Lessoine M., Le Tallec P. Load and motion transfer algorithms for fluid-structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity – Computer Methods in Applied Mechanics and Engineering. 1998. V. 157, № 1-2. P. 95–114.
2. Kopysov S.P., Novikov A.K. Parallel element-by-element conjugate gradient method with decreased communications costs // The International Summer School Iterative Methods and Matrix Computations (2-9 June 2002) / Eds. Gene H. Golub, Lev A. Krukier. Rostov-on-Don, Russia, 2002. P. 450–454.

3. Копысов С.П., Новиков А.К. Метод декомпозиции для параллельного адаптивного конечно-элементного метода // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2010. № 3. С.141–154.
4. Buhmann M.D. Radial Basis Function. Theory and implementations. Cambridge University Press, 2003. 259 p.
5. Копысов С.П., Новиков А.К., Пономарёв А.Б., Рычков В.Н., Сагдеева Ю.А. Программная среда расчётных сеточных моделей для параллельных вычислений // Программные продукты и системы. 2008. № 2. С. 87–89.

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С МНОГОЯДЕРНЫМИ УСКОРИТЕЛЯМИ В РАМКАХ ПРИЛОЖЕНИЙ БАЗ ДАННЫХ

П.С. Костенецкий, А.И. Семенов

Южно-Уральский госуниверситет, Челябинск

На сегодняшний день уже существуют работы, посвященные оптимизации процесса интеллектуального анализа данных [2, 3] и ускорению обработки запросов к базе данных с использованием графических ускорителей [1, 4]. Однако работы в этих научных направлениях не привели к созданию полноценных прототипов и, тем более, промышленных параллельных СУБД. В первую очередь это связано с ограничениями скорости системной шины, по которой данные передаются между CPU и GPU. Необходимость учитывать множество узких мест в архитектуре гибридных узлов значительно усложняет разработку параллельных систем баз данных. В связи с этим возникает вопрос о необходимости исследования и анализа гибридных аппаратных архитектур при помощи математических моделей. Данная работа посвящена моделированию подобных систем баз данных и поиску оптимальных гибридных аппаратных архитектур для обработки запросов не только с использованием центрального процессора, но и с использованием GPU либо многоядерных ускорителей класса Intel Xeon Phy.

В качестве первого этапа работы был реализован эмулятор СУБД, моделирующий параллельную обработку запросов [7] на графическом ускорителе, а также обработку этих же запросов на многоядерном центральном процессоре. При помощи эмулятора исследована эффективность использования графических ускорителей для хранения и обработки баз данных. В результате исследования определено, что с использованием многоядерных графических ускорителей можно добиться многократного ускорения обработки запроса SELECT или вычисления агрегатных функций. По сравнению с обработкой на 4 ядрах центрального процессора удалось добиться ускорения в 83 раза при обработке на 512 нитях GPU. В то же время значительно ускорить алгоритм JOIN с использованием GPU не удается [6]. Исходя из результатов экспериментов, можно сделать вывод, что эффективным будет следующий комбинированный алгоритм работы системы баз данных: обработка запроса JOIN выполняется на ядрах центрального процессора. В это же время на графическом ускорителе, установленном в вычислительном узле, выполняется обработка запросов SELECT или вычисление агрегатных функций, таких как MIN, MAX, SUM, AVG или COUNT. SELECT является наиболее распространенным запросом SQL, поэтому использование такой схемы работы может позволить значительно снизить нагрузку на центральные процессоры. Для реализации данного подхода будет эффективным использование нескольких многоядерных ускорителей. Каждый ускоритель обрабатывает свою часть отношения, над которым требуется выполнить запрос. Далее полученные результаты объединяются в результирующее отношение. Кроме того, ускорители могут быть установлены в вычислительные узлы, объединенные в кластер.

Для реализации подобного комбинированного подхода к обработке запросов в сверхбольших базах данных потребуются новые аппаратные архитектуры систем баз данных, содержащие ускорители GPU либо MIC и жесткие диски. Сборка прототипов подобных высокопроизводительных кластеров с гибридными узлами является чрезвы-

чайно дорогостоящей. В связи с этим для выбора оптимальной конфигурации системы баз данных предварительно необходимо выполнять моделирование возможных аппаратных архитектур при помощи математических моделей. Для моделирования аппаратных архитектур параллельных систем баз данных наиболее подходит *модель мультипроцессоров баз данных DMM* [5]. Однако для анализа гибридных иерархических архитектур необходимо выполнить расширение для данной модели. Это является направлением дальнейших исследований.

Работа выполнена при поддержке РФФИ, грант № 12-07-31082 (2012-2013 гг.)

Литература

1. Bakkum P., Skadron K. Accelerating SQL database operations on a GPU with CUDA // Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU'10), Pittsburgh, USA, March 14, 2010. ACM. P. 94–103.
2. Ding S., He J., Yan H., Suel T. Using graphics processors for high performance IR query processing. In WWW ’09: Proceedings of the 18th international conference on World wide web, New York, NY, USA, 2009. ACM. P. 421–430.
3. Fang W., Lau K., Lu M. et al. Parallel data mining on graphics processors: Technical report, Hong Kong University of Science and Technology, 2008.
4. Govindaraju N., Lloyd B., Wang W. et al. Fast computation of database operations using graphics processors. In SIGGRAPH ’05: ACM SIGGRAPH 2005 Courses, New York, NY, USA, 2005. ACM. P. 206.
5. Костенецкий П.С. Моделирование параллельных систем баз данных для вычислительных кластеров // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды всероссийской научной конференции (21-26 сентября 2009 г., Новороссийск). М.: Изд-во МГУ. 2009. С. 300–304.
6. Костенецкий П.С., Семенов А.И. Разработка методов обработки запросов к базе данных на многоядерных ускорителях, поддерживающих технологию CUDA // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., Новороссийск). М.: Изд-во МГУ, 2012. С. 435–441.
7. Соколинский Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической структурой // Программирование. 2001. № 6. С. 13–19.

ПРИМЕНЕНИЕ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ДЛЯ ОПРЕДЕЛЕНИЯ ЗАВИСИМОСТИ ПОТЕНЦИАЛЬНОЙ ЭНЕРГИИ И РАВНОВЕСНОГО МЕЖАТОМНОГО РАССТОЯНИЯ ОТ ТЕМПЕРАТУРЫ В ГЦК-МЕТАЛЛАХ

В.И. Кочуров, И.Ю. Зубко

Пермский национальный исследовательский политехнический университет

Одним из способов теоретического исследования термомеханических свойств конденсированных сред является дискретный подход, основанный на прямом моделировании движения частиц, вызванного их взаимодействием и внешними воздействиями. Наиболее распространенным является метод молекулярной динамики (МД), когда для каждой частицы (атома или молекулы) решаются уравнения движения, содержащие силы и моменты взаимодействия выбранной частицы со всеми частицами тела или некоторой ее окрестности. Силы взаимодействия частиц определяются по потенциалу их взаимодействия, который представляет собой приближенный способ описания взаимодействия атомов, качественно отражающий их свойства отталкиваться на малых и притягиваться на больших расстояниях. Получаемые результаты зависят от числовых значений параметров потенциалов. Ограничения на размеры тел и период времени налагаются возможностями вычислительной техники и алгоритмов. Одной из проблем является идентификация параметров потенциала: свойства наночастиц, для которых МД работает хорошо, в реальных экспериментах исследованы мало, и для их определения часто используются расчеты, основанные на применении тех или иных потенциалов межатомного взаимодействия. До описания движения макроскопических тел, свойства которых изучены достаточно полно, метод МД «не дошел» – такие тела содержат более 10^{24} атомов, и в настоящее время провести прямые расчеты по решению системы соответствующего числа уравнений не представляется возможным.

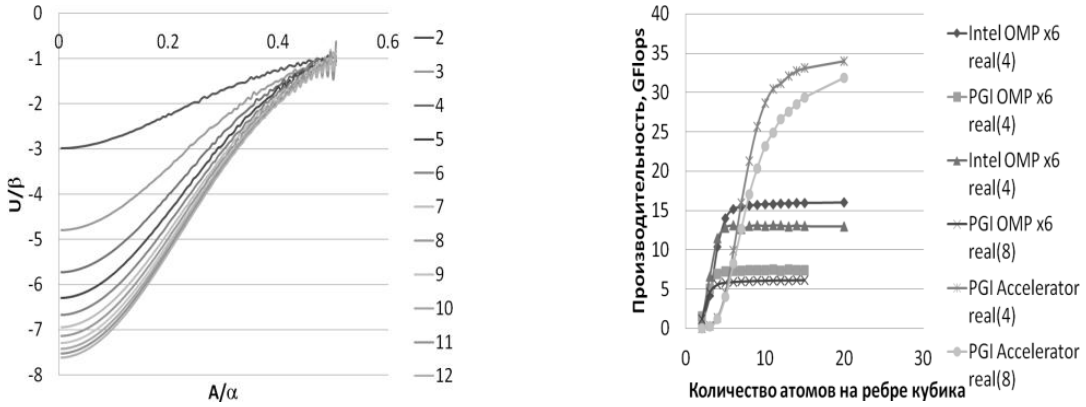
В работах [1, 2] предложена модификация метода частиц, позволяющая провести идентификацию параметров потенциала межатомного взаимодействия для металлических монокристаллов с ГЦК- и ОЦК-решетками по их макроскопическим параметрам – равновесному межатомному расстоянию и упругому модулю сдвига. В основе лежит рассмотрение статики взаимодействующих атомов при явном задании структуры решетки кристалла. Возможности подхода демонстрировались на примере потенциала Леннарда–Джонса, применимость которого обоснована тем, что при исследовании механических свойств нет необходимости рассматривать процессы, идущие при сверхнизких температурах или при скоростном деформировании, когда важную роль в поведении атомов могут играть квантовые эффекты. Использование статической постановки позволяет получить точные выражения для упругих модулей и равновесного межатомного расстояния в различных кристаллических решетках в зависимости от размера образца с числом N атомов на ребре от 3 до 20. Для перехода на макроуровень по этим точным решениям сделан предельный переход при $N \to \infty$. В [1, 2] не обсуждался вопрос об учете температуры при определении механических свойств. В представляемой работе ставится задача о способе задания температуры в статическом подходе. При моделировании поведения механических свойств различных тел методом МД температура вводится через кинетическую энергию движения атомов относительно среднего движения. При стабилизации системы энергия крупномасштабного движения атомов перехо-

дит в мелкомасштабные колебания с меньшей амплитудой. Эти мелкомасштабные колебания относительно не меняющихся со временем центров отождествляются с тепловым движением, и температура тела T определяется с помощью выражения $E = kT$, где E – средняя амплитуда пульсаций кинетической энергии, k – постоянная Больцмана. В статическом подходе нет возможности вводить температуру таким образом. Здесь тепловые колебания атомов с некоторой заданной амплитудой A имитируются наложением на систему атомов случайных смещений с этой амплитудой при равномерном распределении направления смещений в пространстве. Частота этих смещений (колебаний) в таком подходе рассматривается как независимый параметр, требующий идентификации.

Для определения зависимости a^* (равновесного межатомного расстояния монокристалла в форме куба) для каждого заданного A при произвольных a (параметр решетки), α и β (параметры потенциала $\varphi(r) = \beta\left((\alpha/r)^{12} - 2(\alpha/r)^6\right)$ Леннарда–Джонса) вычислялись силы на гранях куба и находился минимум суммы их квадратов. Минимум такой суммы не всегда равен нулю, поскольку для каждого отдельного распределения атомов не всегда выполнялось условие равновесия, справедливое для сил, полученных осреднением по набору реализаций возмущенной конфигурации атомов. Сила на грани образца равна $\beta(C_1^{(i)}(n,A)\alpha^6 - C_2^{(i)}(n,A)a^6)/a^{13}$, где индекс $i = \overline{1,6}$ задает номер грани, n – число атомов на ребре куба. Сумма квадратов сил имеет минимум при $a^* = \sqrt[6]{\sum_{i=1}^{6} C_1^{(i)} C_2^{(i)} / \sum_{i=1}^{6} C_2^{(i)2}}\,\alpha$.

Для каждого значения амплитуды A строились 1000 реализаций случайных конфигураций решетки и по ним определялось среднее значение межатомного расстояния. Также для каждой реализации при расстоянии $a = a^*$ определялась величина u потенциальной энергии всей системы атомов, отнесенной к числу атомов. Зависимость безразмерных величин u/β и A/α приведена на рис.1, а.

Для решения задачи использовалась рабочая станция с процессором AMD Phenom X6 1055T и графическим ускорителем Nvidia GeForce GTS460. В качестве языков программирования были выбраны Intel Fortran Composer XE и PGI Accelerator Fortran v.12. Расчеты показали более чем двукратное преимущество вычислений на графическом ускорителе (с использованием технологии PGI Accelerator) по сравнению с наиболее эффективными реализациями OpenMP-распараллеливания как для переменных одинарной, так и для переменных двойной точности (рис.1, б).



а) б)

Рис. 1. Результаты: а) зависимость u/β от A/α при разных N,
б) сравнение производительности расчетов (различные технологии)

Работа выполнена при финансовой поддержке РФФИ (грант №10-08-00156-а, грант №11-01-96033-р-Урал-а).

Литература

1. Зубко И.Ю., Трусов П.В. Определение упругих постоянных ГЦК-монокристаллов с помощью потенциала межатомного взаимодействия // Вестник ПГТУ. Механика. Пермь, 2011. Т.1. С.147–169.
2. Зубко И.Ю., Мелентьева О.В., Морозова В.П., Кочуров В.И. Вывод упругого закона монокристаллов металлов из потенциала межатомного взаимодействия // Вестник Нижегородского университета им. Н.И. Лобачевского. Н. Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2011. №4. Ч.5. С. 2181–2183.

УНИФИЦИРОВАННЫЙ УДАЛЕННЫЙ ДОСТУП К СУПЕРКОМПЬЮТЕРНЫМ РЕСУРСАМ

А.П. Крюков[1], Л.В. Шамардин[1], Н.В. Приходько[2], А.П. Демичев[1]

[1]Научно-исследовательский институт ядерной физики им. Д.В. Скобельцына МГУ им. М.В. Ломоносова, Москва

[2]Новгородский госуниверситет им. Ярослава Мудрого

Предложен новый подход к организации удаленного доступа к суперкомпьютерным (СК) ресурсам, который может быть применен как в гридах, так и для прямого доступа пользователей к СК. В настоящее время для запуска заданий в гриде традиционно используются программные шлюзы на основе Globus toolkit, а в случае прямого доступа пользователей – протокол ssh. Оба варианта существенно различаются, что затрудняет использование СК пользователями. Предлагается использовать программный шлюз к СК ресурсам, входной язык описания заданий (workflow) которого разработан и реализован в рамках проекта ГридННС. Это позволит унифицировать запуск заданий как в гриде, так и непосредственно на СК. От пользователя скрывается специфика запуска заданий на конкретном СК, что существенно облегчает работу на нескольких СК одновременно.

Проблема удаленного доступа к вычислительным ресурсам особенно актуальна в случае необходимости использования СК для выполнения исследований. Количество крупных СК установок достаточно мало, и далеко не каждый университет или научно-исследовательский институт может позволить себе иметь собственный СК. Тем более такой, как уникальная вычислительная установка СК «Ломоносов» [1], работающий в МГУ имени М.В. Ломоносова.

Для организации удаленного доступа пользователей к СК установкам используется несколько методов. Это может быть, например, прямой доступ к конкретному СК или доступ через грид. Рассмотрим их более подробно.

Наиболее распространенным в настоящее время методом удаленного доступа к СК ресурсам является предоставление прямого доступа к СК по протоколу ssh. Прямой доступ предоставляет пользователю гибкую возможность использовать средства, имеющиеся на СК. Однако этот способ создает значительные проблемы как для пользователей, так и для системных администраторов СК.

Проблемы пользователей состоят в том, что они должны изучить многие тонкости среды, в которой происходит запуск приложений. Сюда относится система пакетной обработки заданий, установленная на СК, особенности ОС и ее конкретной версии, особенности архитектуры СК и так далее. Например, СК «Ломоносов» имеет неоднородную структуру. Поэтому эффективное использование этого СК возможно только при учете этого факта.

Для системных администраторов проблема удаленного доступа к СК заключается, в первую очередь, в администрировании большого количества пользователей из различных организаций и в обеспечении безопасности СК в этих условиях.

В случае гридов удаленный доступ к СК ресурсам организуется посредством промежуточного ПО, которое устанавливается на специальном сервере-шлюзе. Шлюз обеспечивает согласование грид-среды и среды исполнения конкретного СК.

В этом случае многие указанные проблемы уходят на второй план. Так, например, пользователям не требуется изучения особенностей конкретной среды исполнения,

имеющейся на каждом доступном СК. Система управления заданиями грида автоматически настроит задание пользователя в соответствии с требуемой средой исполнения, особенностями системы управления пакетной обработки заданий и корректно запустит задачу пользователя.

То же самое относится к проблеме администрирования пользователей и обеспечения безопасности СК. В гриде управление пользователями организовано через механизм виртуальных организаций (ВО), который обеспечивает гибкое управление правами доступа пользователей, а использование цифровых сертификатов стандарта X.509 обеспечивает безопасность доступа к СК. Проблема управления большого количества пользователей решается путем использования ограниченных пулов формальных локальных пользователей, на которые отображаются реальные пользователи на время выполнения заданий. При этом средства шлюза обеспечивают возможность проследить за всеми действиями пользователя (аудит).

В качестве базового ПО, которое используется для построения шлюзов, как правило, применяется Globus Toolkit (GT) [2]. Однако использование GT имеет свои недостатки. Так, GT версии 4 использует сложный в реализации стек стандартов WSRF [3]; GT5 является фактически возвращением к классическому инструментарию GT2 с учетом опыта последних лет.

В ходе разработки промежуточного ПО ГридННС [4] авторами впервые для целей грид были разработаны грид-сервисы [5, 6] с использованием архитектурного стиля REST [7] и протоколом обмена данными JSON [8]. Используя успешный опыт разработки RESTful-грид-сервисов, авторами был разработано промежуточное ПО для шлюза к СК ресурсу, в котором реализован новый подход к разработке подобных сервисов.

Основная задача, которую выполняет шлюз удаленного доступа к СК ресурсу (в дальнейшем просто шлюз), – это регистрация задачи пользователя и трансляция описания задачи в термины среды исполнения конкретного СК с последующей передачей задачи на выполнение на СК. Кратко это можно описать следующем алгоритмом:

- прием описания задачи и регистрация ее в БД шлюза;
- загрузка в домашнюю директорию файлов, необходимых для запуска задачи;
- трансляция описания задачи в термины системы управления пакетной обработки (СУПО);
- постановка задачи в очередь СУПО и выполнение задачи на СК;
- выгрузка результатов задачи;
- завершение задачи.

Описанный алгоритм обеспечивает выполнение задачи на СК в рамках одной единицы выполнения СУПО СК. Реальные задачи часто требуют выполнения нескольких логически связанных шагов (композитные задания, workflow), не все из которых должны выполняться в одинаковых условиях запуска в рамках одного шага. Например, задача может включать в себя подготовку входных/выходных данных, интенсивный ввод-вывод, не требующую серьезных вычислительных ресурсов (например, работа с архивами файлов) предварительную компиляцию программы. Последний этап в ряде случаев вообще не может быть осуществлен в момент выполнения основного скрипта задачи из-за ограничений сред исполнения на конкретных СК.

Таким образом, приведенный алгоритм имеет существенные ограничения возможных сценариев использования как в случае грида, так и в случае обращения к конкретному СК через шлюз. Для устранения указанного ограничения было предложено распространить использование композитных заданий не только в среде грида, но и на шлюзах.

В качестве языка описания был использован язык описания заданий в ГридННС. Этот язык позволяет описать композитное задание, логика выполнения которого может быть представлена в виде направленного ациклического графа (НАГ). Однако есть одно

существенное отличие. А именно, НАГ содержит не один тип, как в ГридННС, а несколько типов вершин. Первый тип – это вершины, которые предписывают выполнить некоторые действия на СК, например запуск основной вычислительной части программы. Второй тип вершин – это действия, которые будут выполнены непосредственно на сервере шлюза и суть которых полностью предопределяется и контролируется администраторами шлюза. Примерами таких действий являются пересылка файлов из удаленного хранилища на СК либо работа с архивами файлов. Все такие действия имеют стандартный интерфейс. Для обеспечения безопасности список типов таких вершин определяется администратором шлюза и не может быть расширен пользователем. В качестве формата описания заданий используется формат JSON.

Безопасность шлюза обеспечивается широким комплексом мер. Аутентификация пользователей на шлюзе осуществляется при помощи цифровых сертификатов стандарта X.509. Авторизация осуществляется на основе принадлежности пользователей к той или иной виртуальной организации. Технически это осуществляется при помощи сервиса управления ВО – VOMS [9]. Такое решение обеспечивает большую гибкость управления правами доступа пользователей и, в то же время, не усложняет администрирование СК. Некоторым недостатком этого можно считать необходимость поддержки отдельного сервера управления ВО.

Шлюз может быть использован на СК, которые находятся под управлением СУПО PBS/Torque, Cleo и SLURM, что покрывает практически все СК установки России. В случае использования шлюза для прямого доступа пользователей к СК ресурсу, шлюз обеспечивает единый интерфейс к СК независимо от используемой на нем среды исполнения и особенностей настройки СК.

В настоящее время прототип шлюза проходит комплексные испытания в ГридННС.

Работа поддержана грантом РФФИ (№ 11-07-00434-а) и грантом Президента РФ «Ведущая научная школа» (НШ-3920.2012.2).

Литература

1. Суперкомпьютер Ломоносов – [http://parallel.ru/cluster/lomonosov.html].
2. Foster I., Kesselman C. The Globus Project: A Status Report // In Proc. Heteroge-neous Computing Workshop. IEEE Press. 1998. P. 4–18.
3. OASIS. Web Services Resource Framework (WSRF) – Primer v1.2, Committee Draft 02 - 23 May 2006. – [http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf].
4. Грид для Национальной нанотехнологической сети – [http://ngrid.ru/ngrid].
5. Демичев А., Ильин В., Крюков А., Шамардин Л. Реализация программного интерфейса грид-сервиса Pilot на основе архитектурного стиля REST // Вычисленные методы и программирование. 2010. Т. 11. С. 62–65.
6. Демичев А., Крюков А., Шамардин Л. Принципы построения грид с использованием Restful-веб-сервисов // Программные продукты и системы. 2009. №4. С. 172–176.
7. Fielding R.T. Architectural styles and the design of network-based software architectures // Dotoral dissertation, University of California, Irvine, 2000.
8. K.Zyp: A JSON Media Type for Describing the Structure and Meaning of JSON Documents. // Techniocal report, IETF Network Workong Group, draft-zyp-json-schema-02, March 2010.
9. VOMS – [http://vdt.cs.wisc.edu/components/voms.html].

СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ЗАДАЧЕ ПОИСКА ИЗОБРАЖЕНИЙ ПО ВИЗУАЛЬНОМУ ПОДОБИЮ

К.В. Кузнецов, В.М. Михелев, Д.А. Торопчин, Д.В. Петров

Белгородский государственный национальный исследовательский университет

Рассматривается применение суперкомпьютерных технологий в задаче поиска изображений по визуальному подобию с решением, основанным на сравнительно простых и быстро вычисляемых мерах, использующих цветовые признаки и яркостные переходы на изображении.

Введение

В настоящее время содержимое Интернета включает не только текстовые документы, но и графические объекты и видеоизображения, которые содержат информацию очень большого объема. Кроме этого, появились очень большие по объему информации базы данных изображений как у отдельного пользователя, так и у различных организаций. В этих условиях весьма актуальным становится вопрос о навигации среди этой массы информации, и о быстром и точном поиске нужных пользователю данных. Также актуально и создание высокопроизводительных облачных сервисов, реализующих параллельные алгоритмы распознавания нечетких дубликатов изображений.

Однако существующие в настоящее время алгоритмы поиска дубликатов нечетких изображений имеют следующие недостатки:

- высокая ресурсоемкость, особенно заметная в процессе поиске в большой коллекции;
- огромные временные затраты;
- значительная погрешность в полученных результатах.

В связи с этим нами был разработан алгоритм, который позволил использовать простые и быстро извлекаемые из изображений признаки, основанные на анализе цветовых признаков и яркостных переходов на изображении. Такой подход обеспечил высокопроизводительное решение задачи поиска изображений по визуальному подобию.

Теоретические основы

В качестве цветовой модели изображения была выбрана схема HSV, которая представляет каждый пиксель тремя компонентами: тон (hue), насыщенность (saturation) и значение (value). Это позволяет описать каждое изображение тремя вещественными функциями от координат, то есть изображение будет представлено тремя двумерными векторами. Предпочтение было отдано в пользу HSV вместо других моделей, таких как RGB и CMYK, по двум причинам. Во-первых, потому что модель HSV ближе к человеческому восприятию, так как модели RGB и CMYK определяют цвет как комбинацию базовых цветов, в то время как компоненты модели HSV отображают информацию о цвете в более привычной для человека форме. Эта модель позволяет определить, какой именно цвет: насыщенный он или тусклый, светлый или темный [1].

Разработанный нами алгоритм поиска изображений по визуальному подобию состоит из следующих этапов:

1. Приведение изображения к виду, удобному для анализа.
2. Вычисление меры схожести на основе цветовых признаков изображения.

3. Вычисление меры отличия между изображениями.
4. Определение значимых объектов, расположенных на изображении.
5. Определение схожести объектов по форме на основе гистограмм градиентов и яркостных переходов на изображении.
6. Принятие решения о схожести изображений по визуальному подобию.

Приведение изображения к виду, удобному для анализа

На этом этапе изображения переводятся к одинаковому разрешению и выполняется их предварительная обработка. Для размера изображения выбиралось значение 100*100 пикселей, такая размерность картинки позволяет человеческому глазу судить о схожести изображений. Также для изображений в качестве предобработки применялось размытие по Гауссу, что сглаживает некоторые детали.

Вычисление меры схожести на основе цветовых признаков изображения

Изображение квантуется по компоненте h(hue) на 32 цвета. Затем для каждого цвета запоминается количество его вхождений (количество пикселей данного цвета). Для сравнения изображений в дальнейшем будут использоваться 3 наиболее частых цвета на этих изображениях. Мера схожести вычисляется по следующей формуле:

$$D = \sum_{i=1}^{3} r(I_{1i}, I_2) + \sum_{i=1}^{3} r(I_{2i}, I_1),$$

где I – упорядоченный по частоте список цветов, a – цвет, c – частота цвета ‘a’ на первом изображении, d – частота цвета ‘a’ на втором изображении.

При этом в зависимости от наличия исследуемого цвета, например ‘a’, на втором изображении значение выражения $r(a, I)$ определяется следующим образом:

- $r(a,I) = |c-d|$, если цвет ‘a’ представлен на втором изображении,
- $r(a, I) = c$, если цвет ‘a’ отсутствует на втором изображении.

Вычисление меры отличия между изображениями

Мера отличия противоположна по смыслу мере сходства. В ней суммируются частоты цветов, не встречающихся в двух изображениях:

$$D_1 = \sum_{i=1}^{3} (I_{1i} + I_{2i}),$$

где I – это упорядоченный список частот по убыванию, таких, которых нет в другом изображении.

Пример: I_1= {‘200’:0.43 , ‘20’:0.30,‘120’:0.17, ‘280’:0.10}, I_2 {‘200’:0.80, ‘40’:0.30,‘80’:0.17, ‘180’:0.10}. Цвета ‘200’ совпадают, поэтому они не войдут в список I_1, I_2. Поэтому I_1 = {‘20’:0.30,‘120’:0.17,‘280’:0.10}, I_2 {‘40’:0.30,‘80’:0.17, ‘180’:0.10}, а мера отличия будет равна 1,14.

Определение значимых объектов расположенных на изображении

Для выделения схожих объектов воспользуемся обычным фильтром Собеля, выделяющим границы.

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A, \; G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A,$$

$$G = \sqrt{G_x^2 + G_y^2}, \; \theta = \arctan(G_y / G_x),$$

где A – изображение , G – значение градиента в точке, Θ – угол направления градиента.

После выделения пикселей, которые определяют яркостные переходы на изображении границами, группируем их. Группировка осуществляется рекурсивно. Выбирается первый пиксель границы и для него проверяется небольшая окрестность, если рядом существуют пиксели границы, то они входят в эту группу и для них запускается та же процедура поиска. После получения всех границ объектов, составляется отсортированный список объектов и их границ. Затем из этого списка удаляются те объекты, у которых значение границ в пикселях меньше некоего порога.

Определение схожести объектов по форме на основе гистограмм градиентов и яркостных переходов на изображении

Для сравнения границ использовались гистограммы ориентации градиента. В таком виде эти дескрипторы были впервые предложены и описаны исследователями Navneet Dalal и Bill Triggs. Рассматриваемый метод построения дескрипторов на основе HoG базируется на том, что наличие и форма локальных объектов на изображении могут быть описаны распределением интенсивности градиентов.

Для улучшения качества детекции локальные гистограммы могут быть нормализованы по контрасту путем вычисления меры интенсивности по большему региону изображения, так называемому блоку, и использования этого значения для нормализации гистограмм всех клеток в блоке. Такая нормализация ведет к лучшей инвариантности относительно изменений освещенности и теням. Было предложено строить гистограммы, используя не все изображение, а лишь окрестности границ, выделенных на предыдущем этапе.

Рис. 1. Гистограмма направлений градиентов

Далее сравнивались окрестности по цветовым признакам, описанным раньше, а также по гистограммам направлений градиентов:

$$F = 1 - rn_{peaks} \sum_{i=1}^{8} (H_{1i} - H_{2i}),$$

где n – число пиков гистограммы, больших некоторого порога; H – гистограммы градиентов по 8 направлениям, где элементом является значение G по углу Θ; r – нормализующий множитель. Окрестности границ также сравнивались по мере сходства и отличия.

Принятие решения о схожести изображений по визуальному подобию

Общий вывод о схожести изображений определяется на основе общей взвешенной меры, полученной в виде суммы из приведенных выше частных мер.

Вычислительный эксперимент

Вычислительный эксперимент проводился на коллекции в 20 тысяч картинок [2]. Для полного сравнения коллекции необходимо было около 200000 сравнений изображений между собой.

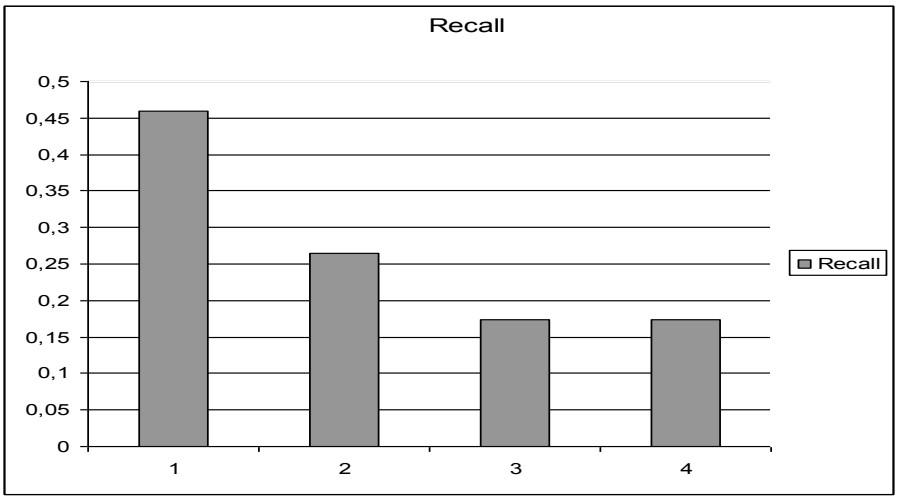


Рис. 2. Полнота

Основными характеристиками, определяющими значимость алгоритма поиска, являются полнота и точность. Оценку результатов поиска производили независимые асессоры [2], ими для оценки лучшего алгоритма использовался метод общего котла. На графиках (рис. 2 и рис. 3) представлены значения полноты и точности, полученные по результатам расчетов по разработанному нами алгоритму (номер 1) в сравнении с другими алгоритмами (номера 2, 3, 4) [2]. Преимущество разработанного нами метода объясняется не только комбинированием мер, основанных на цветовых компонентах, но и применением мер, вычисляемых с помощью гистограмм градиентов.

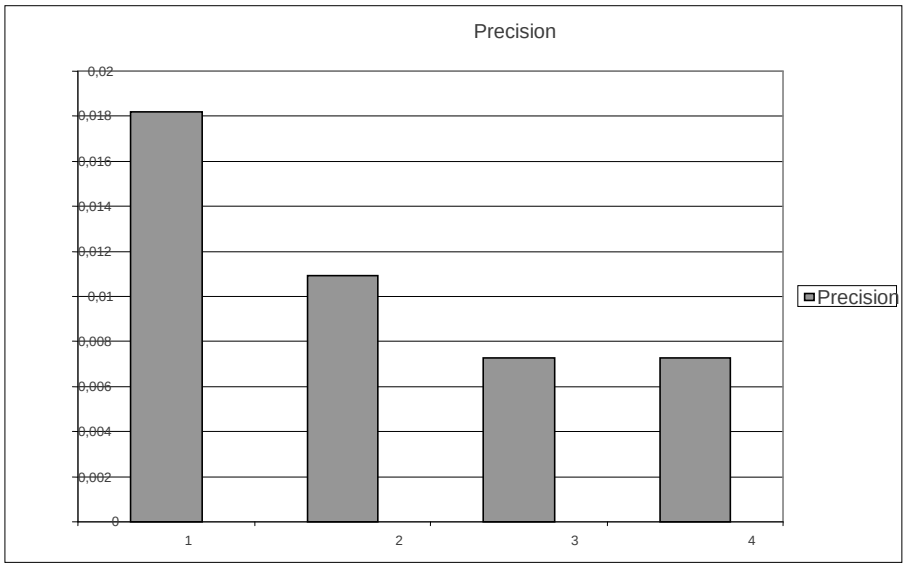


Рис. 3. Точность

Как показали вычислительные эксперименты, время расчета одной картинки составляло несколько сотых секунды, поэтому разработанный алгоритм обеспечивает высокопроизводительное решение задачи поиска изображений по визуальному подобию для больших коллекций. Однако, когда необходимо построить индекс для нескольких миллиардов картинок, необходимо данный алгоритм использовать в сочетании с со-

временными технологиями параллельного программирования. Поэтому даже при тестировании алгоритмов нами применялись гибридные технологии параллельного программирования MPI и CUDA. Технология MPI позволила осуществлять расчеты на большом количестве компьютеров (горизонтальное масштабирование), а технология CUDA обеспечила расчеты, хорошо подходящие под эту архитектуру, в частности выполнялись расчеты с использованием фильтров Собеля. Такие задачи могут также относиться к НТС; так, например, можно осуществлять построение индексов (вычисление самих признаков). После построения индекса соответственно можно применять такие технологии, как OpenMP, внутри каждого MPI процесса. Это сократит расход памяти, что существенно для данной задачи, потому что на хранение одного изображения уходит малое количество информации. При большом индексе мы сможем хранить больше изображений в оперативной памяти и меньше обращаться к жесткому диску, что существенно сократит время поиска.

Выводы

Разработан и исследован метод поиска изображений по визуальному подобию на основе анализа их визуальных характеристик, который обеспечивает высокопроизводительное решение задачи. Для решения использовались сравнительно простые и быстро вычислимые признаки, основанные на анализе цветовых гистограмм и яркостных переходов на изображении. Использование небольшого набора комбинаций глобальных цветовых и текстурных признаков оказалось эффективным при поиске по разнородной коллекции изображений.

Литература

1. Крешихин Д.Н., Неелова Н.В. Поиск изображений по визуальному подобию на основе цветовой модели HSV // Труды РОМИП. 2009.
2. Российский семинар по оценке методов информационного поиска – [http://romip.ru/].
3. Мельниченко А., Гончаров А. ЛММИИ на РОМИП-2009: Методы поиска изображений по визуальному подобию и детекции нечетких дубликатов изображений // Труды РОМИП. 2009.

ПРИМЕНЕНИЕ GPU ДЛЯ ЗАДАЧ КВАНТОВОЙ ХИМИИ: ПРЯМОЙ РАСЧЕТ МАТРИЦ ПЛОТНОСТИ СВЕРХБОЛЬШИХ МОЛЕКУЛ МЕТОДОМ PDM

М.Б. Кузьминский[1], А.М. Андреев[1], А.М. Чернецов[2]

[1]Институт органической химии им. Н.Д. Зелинского РАН, Москва
[2]Вычислительный центр им. А.А. Дородницына РАН, Москва

Предложена реализация метода PDM для прямого квантовохимического расчета матрицы плотности сверхбольших молекул по фокиану с использованием технологии разреженных матриц и аппаратных средств GPU Nvidia Tesla C2050. Исследование актуально не только собственно для задач химии и биохимии, но и для определения областей эффективного применения GPU, в частности, для задач квантовой химии, поскольку такие приложения (как и сами GPU) потенциально могут использоваться в будущих системах экзафлопсного уровня.

Введение

Одним из возможных путей постороения будущих экзафлопсных систем предполагается применение MPP-систем или кластеров, использующих GPU в узлах. Современные высокопроизводительные многоядерные микропроцессоры, например Intel Xeon или IBM Power7, имеют 64-разрядную пиковую производительность на уровне нескольких сот GFLOPS, в то время как аналогичная пиковая производительность широко используемых GPU Nvidia Tesla M2050 близка к 0.5 TFLOPS. Таким образом, мы имеем отличия в разы, а не на порядок, хотя это соотношение может быть в будущем улучшено в сторону GPU.

Кроме того, высокая эффективность (в смысле уровня производительности) применения GPU достигается при больших размерностях задач, что не всегда характерно, например, при работе с разреженными матрицами. Возможно, применение Intel Xeon Phi будет эффективным уже на задачах меньших размерностей.

Одной из типичных прикладных областей применения петафлопсных и экзафлопсных систем указывается квантовая химия. Практика показывает, что для достижения высокой эффективности распараллеливания на таких системах требуется применение специальных, подчас приближенных методов и программ (например, метода фрагментных молекулярных орбиталей, FMO, в рамках стандартной теории возмущений Меллера-Плессета), отличных от обычно используемых в большинстве HPC-систем.

Прямое построение матрицы плотности на GPU

Использование GPU в задачах квантовой химии сегодня достаточно ограничено и связано обычно с применением умножения матриц больших размерностей. В настоящей работе GPU применяется для расчетов методом PDM (Purification of Density Matrix) в ортогональном базисе с использованием технологии разреженных матриц.

Для полуэмпирических квантовохимических методов типа AM1, PM3 и т.д., а также для наиболее эффективных реализаций методов функционала плотности (DFT) лимитирующей стадией расчетов электронной структуры сверхбольших молекулярных систем является диагонализация матрицы фокиана. Для крупномасштабных серийных расчетов больших биомолекул (в том числе актуальных для задач конструирования лекарств докинг-комплексов протеин-лиганд) и наноструктур такой подход приводит к

неприемлемо высоким временам расчетов даже наиболее простыми полуэмпирическими методами.

В рамках проекта РФФИ 11-07-00470 нами создается быстродействующий квантовохимический метод с аппроксимацией фокиана DFT [2]. Метод должен обеспечить точность на уровне, близком к DFT, при быстродействии на уровне полуэмпирических методов. Вместо диагонализации фокиана в ортогональном базисе там планируется применять метод PDM с прямым построением матрицы плотности по фокиану без предварительной диагонализации последнего. При использовании технологий разреженных матриц это обеспечивает линейное масштабирование времени расчета с ростом размера молекулы (базиса).

Метод PDM реализован нами для блочно-n-диагонального портрета матриц (с исходным трехдиагональным их представлением) с автоматическим динамическим определением размера блоков. Этому отвечают квазилинейные молекулы, в частности, полимеры (включая гетерогенные биополимеры) и развернутые протеины.

Программа реализована на языке Fortran-9X в среде Linux/x86-64 (OpenSuSE 11.1). В работе использован GPU Nvidia Tesla C2050 с архитектурой Fermi. В качестве хоста применялся двухпроцессорный сервер на базе четырехъядерных процессоров Intel Xeon E5520 с микроархитектурой Nehalem. Использован компилятор PGI Accelerator Fortran 11.4, CUDA-3.2RC2 и соответствующая библиотека CUBLAS. При сопоставлении с расчетами на хосте (без GPU) использована библиотека MKL из состава Intel Fortran Professional 11.1.

В нашем случае программа умножения матриц dgemm работает с относительно небольшими матрицами (блоками полной матрицы), поэтому для обеспечения преимущества GPU над традиционными микропроцессорами необходима тщательная оптимизация программ. В частности, необходимо минимизировать время передачи матриц из хоста в GPU и обратно.

В результате оптимизации, в частности, благодаря применению закрепленной (pinned) памяти, программа ускорена в 2,6 раза. Для молекулы полиглицина с размером базиса порядка 2,8 тысяч АО и блоков размерностью 200 время расчета матрицы плотности по фокиану с применением GPU составило около 43 секунд. Это в 7 раз быстрее, чем на одном процессорном ядре сервера-хоста, и в 2 раза быстрее, чем на двух процессорах (восьми ядрах) сервера. Преимущество GPU возрастает с ростом размера блока.

Однако применение нами сервера с четырехъядерным процессором Intel E3-1420 с микроархитектурой последнего поколения Sandy Bridge позволяет существенно ускорить расчеты на одном процессорном ядре или на одном процессоре. Поэтому вопрос о выгодности применения GPU по сравнению с традиционными микропроцессорами в нашем случае требует дальнейшего изучения. В настоящее время осуществляется дальнейшая оптимизация PDM-программы.

Работа поддержана РФФИ, проект 11-07-00470.

Литература

1. Niklasson A.M.N., Tumczak C.J., Challacombe M. Trace resetting density matrix purification in O(N) self-consistent-field theory // J. Chem. Phys. 2003. V. 118. № 15. P. 8611–8620.
2. Аникин Н.А., Кузьминский М.Б. Новый метод модельного функционала электронной плотности для описания электронного строения сверхбольших нано- и бионаноструктур // Современные проблемы математики, информатики и биоинформатики: Тезисы докладов Международной конференции, 11-14 октября 2011 / Институт вычислительных технологий СО РАН. Новосибирск. 2011. С. 66–67.

ПРОЕКТ ПРАКТИКУМА ПО КУРСУ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ, ОСНОВАННЫЙ НА CUDA

А.О. Курсиков

Тульский госуниверситет

Введение

В настоящее время вычисления на GPU приобретают все большую актуальность. В связи с этим необходимо введение учебных курсов для подготовки студентов по направлению “Прикладная математика и информатика” для обучения основам программирования для GPU. Уже издана учебная литература [1], но ввиду недостаточного количества простых примеров, приведенных в данной книге, она не может быть использована в качестве основного учебного пособия для студентов третьего курса. В рамках создания курса “Вычислительные системы и параллельная обработка данных” в Тульском государственном университете на механико-математическом факультете разрабатывается учебный практикум, который должен охватывать все основные аспекты программирования вычислительных задач на современных параллельных архитектурах GPU. На практических занятиях будет предложено реализовать алгоритмы, изложенные в курсе “Численные методы” [2]. Такой выбор дает ряд преимуществ: у слушателей курса не будет необходимости изучать новые алгоритмы и будет возможность сравнить GPU-реализации с последовательными реализациями на CPU. В учебную программу планируется включить следующие задачи:

1) базовые задачи линейной алгебры;
2) численное интегрирование;
3) решение СЛАУ методом простой итерации;
4) более сложные задачи.

Простейшие задачи линейной алгебры

Для введения в параллельное программирование будет предложен классический пример сложения больших векторов. Одной из целей этого примера является знакомство со средой выполнения. Следующей лабораторной работой будет реализация параллельного суммирования (редукции), которая в дальнейшем будет использована в задаче численного интегрирования. Ввиду исчерпывающего описания в литературе, возможно предоставление для самостоятельной работы задачи перемножения матриц.

Численное интегрирование

Для численного интегрирования предлагается использовать простейшие формулы прямоугольников, трапеций, Симпсона и более сложные формулы Ньютона–Котеса. Например, для метода средних прямоугольников имеем следующую расчетную формулу:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} hf\left(x_i + \frac{h}{2}\right),$$

где f – подынтегральная функция, x_i – середина i-го интервала длины h.

Выполнены следующие реализации:

1) с параллельным вычислением значений подынтегральной функции на участках разбиения отрезка интегрирования и последующим последовательным суммированием результатов на CPU;

2) с параллельным суммированием (с применением библиотеки CUDPP, собственных реализаций с использованием разделяемой памяти (shared memory) и устранением дивергентного ветвления (divergent branching));

3) с использованием формул Ньютона–Котеса более высоких степеней.

Они позволяют последовательно подойти к изучению основ программирования на GPU. При этом в силу того, что рассматриваемые алгоритмы задачи численного интегрирования просты и изучены в курсе численных методов, трудностей, связанных с их пониманием, возникать не должно.

Решение СЛАУ методом простой итерации

Согласно методу простой итерации исходная система $Ax = b$ приводится к виду

$$x = Bx + \beta,$$

где x – вектор переменных, B – приведенная матрица системы, β – приведенный вектор свободных коэффициентов. Затем задается начальное значение x_0 и находится следующее приближение $x_1 = Bx_0 + \beta$. Процесс продолжается до достижения требуемой точности. Критерием остановки может служить близость по норме двух последних приближений.

Для сходимости процесса необходимо и достаточно того, чтобы все собственные значения матрицы B по модулю были меньше единицы

Пример решения СЛАУ методом простой итерации тоже крайне прост, и на нем можно продемонстрировать все основные аспекты использования ресурсов GPU. Были сделаны четыре реализации этого алгоритма:

1) “наивная” реализация, в которой параллельно выполняется операция скалярного произведения векторов;

2) разбиение задачи на блоки для уменьшения числа обращений к глобальной памяти, транспонирование матрицы для непрерывного чтения (coalesced reading);

3) использование разделяемой памяти;

4) выполнение базовых операций линейной алгебры с использованием библиотеки CUBLAS.

Как и в предыдущем случае, пример усложняется от реализации к реализации и освещает основные аспекты программирования на GPU. Особый интерес представляет третья реализация, поскольку при блочном решении задачи необходимо подобрать оптимальные параметры блока (количество используемых переменных и строк исходной матрицы). В противном случае получаем потерю производительности. Этот пример наглядно демонстрирует зависимость производительности вычислений от архитектуры GPU.

Более сложные задачи

В разрабатываемом учебном курсе центральное место занимают упомянутые выше задачи, поскольку они охватывают базовые основы программирования для GPU. Рассматривается вопрос о введении в курс более сложных задач, например, решение СЛАУ с трехдиагональной матрицей или реализацию одной из базовых операций с разреженными матрицами.

Литература

1. Боресков А.В. и др. Параллельные вычисления на GPU. Архитектура и модель CUDA: Учебное пособие. М.: Издательство МГУ, 2012. ISBN 978-5-211-06340-2.
2. Калиткин Н.Н. Численные методы. СПб.: БХВ-Петербург, 2011. ISBN 978-5-775-0500-0.

ОБ ОДНОМ ПОДХОДЕ К РЕШЕНИЮ ЗАДАЧИ ВИДЕОДЕТЕКТИРОВАНИЯ ТРАНСПОРТНЫХ СРЕДСТВ

В.Д. Кустикова

Нижегородский госуниверситет им. Н.И. Лобачевского
E-mail: valentina.kustikova@gmail.com

Рассматривается задача видеодетектирования транспортных средств. Приводится схема решения задачи с использованием методов поиска и сопровождения на потоке видеоданных. Описывается прототип разработанной системы для решения задачи видеодетектирования. Приводятся результаты вычислительных экспериментов.

Введение

Задача видеодетектирования транспортных средств является актуальной, ее решение позволяет оценивать пропускную способность дорожных участков, эффективно выполнять автоматическое регулирование движения, принимать обоснованные решения о необходимости строительства дорог-дублеров и развязок, проектировать дорожную сеть в масштабах города. Решение данной задачи может быть интересно не только дорожным организациям, но и рекламным службам, владельцам автостоянок и крупных торговых комплексов с точки зрения организации их работы.

1. Постановка задачи

Метод видеодетектирования автомобилей работает с потоком видеоданных. Обозначим входной видеопоток V. По существу, V можно представить в виде последовательности кадров или изображений $I_0, I_1, I_2, \dots, I_{N-1}$, где N – количество кадров.

Задача видеодетектирования транспортных средств состоит в том, чтобы каждому кадру I_k исходного видео V поставить в соответствие совокупность положений объектов BB_k. В зависимости от алгоритма под положением может пониматься расположение контуров объекта [1], особых точек или расположение трехмерной геометрической модели объекта [2]. Далее примем, что положение объекта определяется расположением прямоугольника, его окаймляющего, аналогично работам [3–6]. Тогда задача видеодетектирования сводится к построению отображения $\varphi: \{I_k, k = \overline{0, N-1}\} \to \{BB_k, k = \overline{0, N-1}\}$, где $BB_k = \{bb_l^k, l = \overline{0, s_k - 1}\}$ – множество окаймляющих прямоугольников, обнаруженных на кадре I_k; $s_k = |BB_k|$ – количество найденных прямоугольников. При этом каждый прямоугольник bb_l^k определяется набором следующих компонент: $bb_l^k = \left((x_1^l, y_1^l), (x_2^l, y_2^l)[, s^l, c^l]\right)$, где $(x_1^l, y_1^l), (x_2^l, y_2^l)$ – координаты левого верхнего и правого нижнего углов прямоугольника ($x_1^l \geq 0, y_1^l \geq 0, x_2^l \geq 0, y_2^l \geq 0$), $s^l \in \mathbb{R}$ – достоверность того, что объект продетектирован правильно, а $c^l \in C = \{'car', 'bus', \dots, 'train'\}$ – класс, которому принадлежит транспортное средство. Отметим, что в зависимости от специфики решаемой задачи и выбора метода ее решения последние две компоненты могут отсутствовать. Также положение прямоугольника может быть задано, например, координатами левого верхнего угла и значениями ширины и высоты сторон.

Результат работы выбранного алгоритма не всегда зависит только от информации, извлекаемой из текущего кадра. В общем случае отображение φ зависит от целого

набора параметров $\varphi\left(I_{k-l_1}, I_{k-l_1+1}, \dots, I_{k-1}, \boldsymbol{I_k}, I_{k+1}, \dots, I_{k+l_2}, P^{(d)}, P^{(t)}\right) = BB_k$: информация, полученная на l_1-предыдущих кадрах $I_{k-l_1}, I_{k-l_1+1}, \dots, I_{k-1}$ видео V $(l_1 \leq k)$; информация, извлекаемая из l_2-следующих кадров $I_{k+1}, \dots, I_{k+l_2}$ видео V $(l_2 \leq N-1-k)$; совокупность параметров $P^{(d)}$ алгоритма поиска объектов; набор параметров $P^{(t)}$ алгоритма сопровождения.

В рассматриваемой постановке необходимо дополнительно выполнять построение траекторий движения объектов. Требуется ввести алгоритм сопоставления ψ окаймляющих прямоугольников на паре последовательных кадров I_k и I_{k+1}. Обратим внимание, что на некотором кадре объект исчезает из поля зрения камеры. Поэтому требуется рассматривать множество образов, расширенное специально выделенным окаймляющим прямоугольником с несуществующими координатами, например, $bb = \left((-1,-1),(-1,-1)[,\forall s^l, \forall c^l]\right)$. Тогда $\psi: BB_k \to BB_{k+1} \cup \{bb\}$, $\forall k = \overline{0, N-2}$. Допустим, что I_k – первый кадр, на котором был обнаружен объект, содержащийся в прямоугольнике с индексом $r_0(k) \in \{0,1,2,\dots,s_k\}$, а q – количество последующих кадров, на которых объект виден наблюдателю. Тогда под траекторией движения $T^k_{r_0(k)}$ транспортного средства будем понимать последовательность положений соответствующих окаймляющих прямоугольников $\left(bb^k_{r_0},\ bb^{k+1}_{r_1},\ bb^{k+2}_{r_2}, \dots, bb^{k+q-1}_{r_{q-1}}\right)$, где $r_i \in \{0,1,2,\dots,s_{k+i}\}$, $bb^{k+i}_{r_i} = \psi\left(bb^{k+i-1}_{r_{i-1}}\right)$, $i = \overline{1, q-1}$.

2. Схема решения

Типичная схема работы систем видеодетектирования посредством обнаружения и сопровождения объектов не зависит от типа детектируемых объектов (человек, автомобиль и др.) и состоит из следующей последовательности действий:

1. Извлечение кадра из потока видеоданных.
2. Предобработка полученного статического изображения – кадра (фильтрация, подавление теней, сегментация, выделение областей интереса).
3. Поиск положения объекта на изображении и оценка степени достоверности нахождения объекта в данной области.
4. Сопровождение (трекинг) обнаруженных объектов. На данном этапе выполняется построение траекторий движения транспортных средств, выделенных на предыдущем шаге, от момента их обнаружения до их выхода из области обзора камеры.
5. Анализ результатов поиска и сопровождения (например, оценка направления движения объекта на основании траектории, подсчет числа объектов и т.п.). Функциональность блока анализа результатов в основном зависит от конкретной задачи, которая поставлена перед разработчиками системы видеодетектирования.

3. Программная реализация

Множество всех модулей системы можно разделить на 5 групп (рис. 1):

1. Модули подготовки тестовых данных:
   - модуль разметки окаймляющих прямоугольников для транспортных средств разных классов;
   - модуль формирования траекторий движения транспортных единиц на основании разметки с последующим их сохранением.
2. Модули детектирования и сопровождения:
   - модуль детектирования транспортных средств разных классов на отдельном кадре видео как самостоятельном изображении с помощью параллельной реализации алгоритма Latent SVM [3, 4];

- модуль сопровождения продетектированных транспортных средств на последовательности кадров видеопотока;
- приложения, содержащие реализации различных схем видеодетектирования транспортных средств.

3. Модули апробации результатов видеодетектирования транспортных средств:
   - модуль вычисления некоторых количественных показателей качества детектирования и сопровождения транспортных средств;
   - модуль VOCdevkit [7] для определения средней точности предсказания положения объектов – модуль третьих разработчиков.
4. Служебные модули визуализации окаймляющих прямоугольников и траекторий.
5. Служебные модули для автоматизации сбора численных показателей качества видеодетектирования.

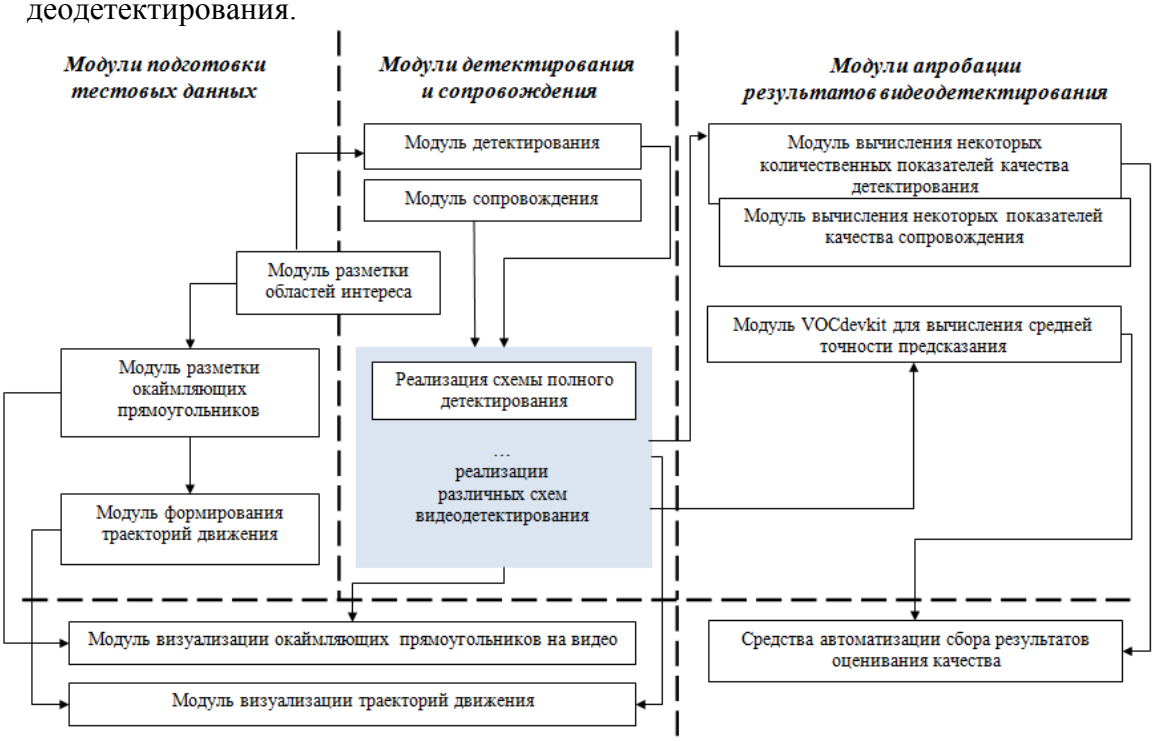


Рис. 1. Архитектура системы видеодетектирования транспортных средств

Модули подготовки тестовых данных, детектирования и сопровождения разработаны с использованием средств библиотеки OpenCV [8].

4. Критерии оценки качества поиска объектов

Выделяются два принципиально разных подхода к оцениванию качества работы систем видеодетектирования. Один из подходов предполагает определение числа правильно и неправильно продетектированных объектов на всем видеопотоке и вычисление различных относительных и абсолютных показателей. Согласно второму подходу, видео рассматривается как последовательность изображений, следовательно, качество поиска оценивается по аналогии с качеством алгоритмов детектирования объектов на изображениях.

В работах [9–11] используются количественные показатели, которые вычисляются на основании информации о числе правильно и неправильно продетектированных объектов:

1. Истинно положительный показатель (the true positive rate [9] или detection rate [10]) – отношение количества правильно продетектированных транспортных средств к общему числу транспортных средств $\mathrm{TPR} = \frac{\text{detected vehicles}}{\text{total number of vehicles}}$.
2. Показатель числа ложных срабатываний (the false detection rate) – отношение количества ложных срабатываний к общему числу срабатываний детектора $\mathrm{FDR} = \frac{\text{false positives}}{\text{detected vehicles+false positives}}$. При этом считается, что объект продетектирован правильно, если перекрытие продетектированного $\mathrm{bbox}_{\mathrm{detect}}$ и размеченного $\mathrm{bbox}_{\mathrm{marked}}$ окаймляющего прямоугольника превышает некоторое пороговое значение threshold (в [10] в качестве порога выбирается 80%) $\frac{\mathrm{S}(\mathrm{bbox}_{\mathrm{detect}} \cap \mathrm{bbox}_{\mathrm{marked}})}{\mathrm{S}(\mathrm{bbox}_{\mathrm{detect}} \cup \mathrm{bbox}_{\mathrm{marked}})} > threshold$.

Наряду с указанными метриками авторы [9] вводят несколько относительных показателей качества:

1. Количество ложных срабатываний, которое в среднем приходится на один кадр (the average false positives per frame) $FPperFrame = \frac{false\ positives}{total\ number\ of\ processed\ frames}$.
2. Среднее количество ложных срабатываний, приходящееся на объект (the average false positives per object) $FPperObject = \frac{false\ positives}{true\ vehicles}$.

Отметим, что авторы [9] не рассматривают задачу классификации транспортных средств, тем не менее, приведенные показатели можно распространить на случай многоклассового детектирования транспортных средств, если рассматривать эти метрики для каждого класса в отдельности.

Авторы [12] предлагают рассматривать видео как последовательность кадров видеопотока и использовать для оценки качества среднюю точность предсказания (average precision). В состав VOC Development Kit [7] входит Matlab-модуль, позволяющий вычислить данный показатель на основании результатов детектирования, записанных в специальном формате. Метрика определяется как математическое ожидание точностей:

$$AP = \frac{1}{11} \sum_{r \in \{0;0.1;0.2;\ldots;1\}} p(r)\,, p(r) = \max_{\overline{r}, \overline{r} \geq r} p(\overline{r}),$$

где $p(\overline{r}) = \frac{1}{a+c}$ – точность, $\overline{r}$ – процент перекрытия детектированного окаймляющего прямоугольника и прямоугольника, который был размечен на исходном изображении как окаймляющий $\overline{r} \in \{0; 0.1; 0.2; \ldots; 1\}$, a – количество объектов, для которых процент перекрытия не меньше, чем $\overline{r}$ (т. е. считается, что объект детектирован правильно), c – количество объектов с процентом перекрытия, меньшим, чем $\overline{r}$ (объект найден ошибочно).

5. Тестовые данные

Для проведения экспериментов по оцениванию качества видеодетектирования транспортных средств было снято несколько треков (25FPS, 720x405 пикселей), из которых выбраны отдельные последовательности кадров:

- *track_09_0-2000* (2000 кадров = 80 секунд) содержит большое количество мелких транспортных средств трех классов (CAR, TRAIN, BUS), движущихся в двух противоположных направлениях (рис. 2, слева). Отметим, что значительную площадь кадра занимает фон.
- *track_10_5000-7000* (2000 кадров = 80 секунд) – видео с транспортными средствами единственного класса CAR, которые двигаются в 4 полосы одного направления (рис. 2, справа). Отличительной чертой является наличие большого количества кадров, не содержащих объектов.

- *track_10_7000-8000* (1000 кадров = 40 секунд) – видео, снятое при тех же условиях, что и track_10_5000-7000. Содержит объекты двух классов CAR и BUS.

Рис. 2. Кадры треков track_09_0-2000, track_10_5000-7000

6. Результаты экспериментов

Результаты видеодетектирования с помощью разработанных схем ниже сведены в общей таблице (таблица 1):

1 – схема полного покадрового детектирования;

2 – схема полного детектирования с отсечением прямоугольников, находящихся вне областей интереса;

3 – схема детектирования с одношаговым трекингом и отсечением. Идея применения трекинга на один кадр вперед состоит в том, чтобы обеспечить дополнительный уровень проверки корректности работы детектора с целью отсечения ложных срабатываний;

4 – схема детектирования с одношаговым трекингом без отсечения;

5 – схема детектирования с трекингом на 5 кадров вперед и отсечением;

6 – схема детектирования с трекингом на 5 кадров вперед без отсечения;

7 – схема детектирования с трекингом на 3 кадра вперед и отсечением.

Таблица 1. Сводная таблица результатов видеодетектирования

№ схемы	Название тестового видео	Средняя точность предсказания (average precision)		Значение показателя *TPR*		Значение показателя *FDR*		Значение показателя *FPperFrame*		Значение показателя *FPperObject*	
		BUS	**CAR**	**BUS**	**CAR**	**BUS**	**CAR**	**BUS**	**CAR**	**BUS**	**CAR**
1	**track_09_0-2000**	0.213	0.202	0.291	0.201	0.924	0.786	1.627	5.898	3.530	0.738
	track_10_5000-7000	-	0.684	-	0.728	-	0.735	-	2.942	-	2.015
	track_10_7000-8000	0.514	0.674	0.657	0.745	0.910	0.813	0.660	3.567	6.677	3.243
2	**track_09_0-2000**	0.217	0.149	0.287	0.114	0.802	0.663	0.532	1.792	1.168	0.224
	track_10_5000-7000	-	0.684	-	0.726	-	0.676	-	2.209	-	1.513
	track_10_7000-8000	0.514	0.674	0.657	0.745	0.905	0.784	0.619	2.980	6.263	2.709
3	**track_10_5000-7000**	-	**0.624**	-	**0.680**	-	**0.437**	-	**0.772**	-	**0.529**
	track_10_7000-8000	0.494	**0.617**	0.596	**0.697**	0.879	**0.631**	0.427	**1.313**	4.313	**1.193**
4	**track_10_5000-7000**	-	0.625	-	0.683	-	0.518	-	1.074	-	0.735
	track_10_7000-8000	0.494	0.617	0.596	0.697	0.884	0.674	0.448	1.583	4.525	1.440
5	**track_10_5000-7000**	-	0.534	-	0.586	-	0.508	-	0.885	-	0.606
	track_10_7000-8000	0.520	0.526	0.606	0.596	0.869	0.667	0.340	1.134	4.04	1.194
6	**track_10_5000-7000**	-	0.617	-	0.622	-	0.605	-	1.390	-	0.952
	track_10_7000-8000	0.520	0.602	0.606	0.621	0.879	0.721	0.436	1.770	4.404	1.609
7	**track_10_5000-7000**	-	0.619	-	0.644	-	0.521	-	1.023	-	0.700
	track_10_7000-8000	0.516	0.610	0.606	0.667	0.875	0.659	0.419	1.416	4.242	1.287

Количественные показатели качества (столбцы 4, 5, 6, 7) приведены для порогового значения процента пересечения размеченных и продетектированных окаймляющих прямоугольников, равного 0.5.

Переход от схемы 1 к схеме 2 – введение отсечения – позволил незначительно снизить количество ложных срабатываний детектора, о чем говорят значения показателей FDR, $FPperFrame$ и $FPperObject$. Применение алгоритмов сопровождения обеспечило еще б*о*льшее улучшение указанных показателей. При этом очевидно прослеживается снижение средней точности детектирования. Если для реализаций, использующих одношаговый трекинг (схемы 3 и 4), это не так заметно, то с многошаговым трекингом (схемы 5, 6, 7) падение точности наблюдается с увеличением шага для выполнения очередного детектирования. Причина состоит в том, что при увеличении шага увеличивается вероятность перекрытия транспортных средств другими объектами или частичный выход из кадра, как следствие, алгоритм детектирования может не найти данный объект, и ошибка распространится на столько предыдущих кадров, чему равен шаг.

В целом наиболее перспективной является схема с одношаговым сопровождением и отсечением. Повысив качество детектирования транспортных средств на каждом кадре за счет использования других алгоритмов и методов, вполне возможно улучшить полученные показатели качества.

Работа выполнена в рамках программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы», государственный контракт № 11.519.11.4015.

Литература

1. Shotton J., Blake A., Cipolla R. Contour-based learning for object detection // 10th IEEE International Conference on Computer Vision. 2005. Vol. 1. P. 503–510.
2. Buch N., Cracknell M., Orwell J., Velastin S. A. Vehicle localisation and classification in urban CCTV streams // 16th World Congress and Exhibition on Intelligent Transport Systems and Services. 2009.
3. Felzenszwalb P. F., Girshick R. B., McAllester D., Ramanan D. Object detection with discriminatively trained part based models // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. Vol. 32, No.9. P. 1627–1645.
4. Druzhkov P.N., Eruhimov V.L., Kozinov E.A. et al. On some new object detection features in OpenCV library // Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications. 2011. V. 21. No.3. P. 384–386.
5. Felzenszwalb P.F., Girshick R.B., McAllester D., Ramanan D. Cascade object detection with deformable path model // Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR'10). 2010. P. 2241–2248.
6. Sun Z., Bebis G., Miller R. On-road vehicle detection using Gabor filters and support vector machines // Digital Signal Processing. 2002. Vol. 2. P. 1019–1022.
7. PASCAL Visual Object Challenge Development Kit – [http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/#devkit].
8. Официальная страница библиотеки OpenCV – [http://opencv.org/].
9. Sivaraman S., Trivedi M.M. A General active-learning framework for on-road vehicle recognition and tracking // IEEE Transactions on intelligent transportation systems. 2010. Vol. 11, No. 2. P. 267–276.
10. Song X., Netavia R. A Model-based Vehicle Segmentation Method for Tracking // ICCV’05. 2005. Vol. 2. P. 1124–1131.
11. Kim Z.W., Malik J. Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking // ICCV’03. 2003. Vol. 1. P. 524–531.
12. Dollrar P., Wojek Ch., Schiele B., Perona P. Pedestrian Detection: A Benchmark // CVPR’09. 2009. P. 304–311.

РЕАЛИЗАЦИЯ И РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ, ОСНОВАННОГО НА ДЕРЕВЬЯХ РЕШЕНИЙ

Е.С. Лепшова, В.А. Биллиг

Тверской госуниверситет

Исследуется алгоритм, основанный на деревьях решений. Обсуждается его использование для решения задач классификации и интеллектуального анализа данных с выявлением скрытых знаний. Целью работы является повышение эффективности, быстродействия и адаптивности метода формирования дерева решений, основанного на алгоритме C4-5. Также рассматриваются возможности модификации алгоритма с использованием распаралеливания в процессе построения дерева.

Стремительное развитие информационных технологий, в частности прогресс в методах сбора, хранения и обработки данных, позволил многим организациям собирать огромные массивы данных, которые необходимо анализировать. Объемы этих данных настолько велики, что возможностей экспертов уже не хватает.

На сегодняшний день интенсивно развивается направление, связанное с интеллектуализацией методов обработки и анализа данных. Интеллектуальные системы анализа данных (ИСАД) призваны минимизировать усилия лица, принимающего решения (ЛПР), в процессе анализа данных, а также в настройке алгоритмов анализа. Многие ИСАД позволяют не только решать классические задачи принятия решения, но и способны выявлять причинно-следственные связи, скрытые закономерности в системе, подвергаемой анализу. Точность результатов исследования прямо пропорциональна объему тестовой выборки и обратно пропорциональна временным затратам. Суммарное время проведения исследований оказывается просто неприемлемым. Поэтому основной целью данной работы является актуальная научно-техническая задача реализации алгоритма и его распараллеливание.

Обоснование выбранного метода. Деревья решений

Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов XX века. Самая ранняя и известная работа Ханта и др., в которой излагается суть деревьев решений – “Эксперименты в индукции” (“Experiments in Induction”) – была опубликована в 1966 году.

Дерево решений – это способ представления правил в иерархической, последовательной структуре. Основа такой структуры – ответы “Да” или “Нет” в случае бинарных деревьев (алгоритм CART), или на ряд вопросов, определимых динамически в случае деревьев с произвольным числом потомков (алгоритм C4-5).

Чтобы решить задачу, т.е. принять решение, требуется ответить на ряд вопросов, которые находятся в узлах дерева, начиная с его корня. При положительном ответе на вопрос осуществляется переход к левой части дерева, называемой левой ветвью, при отрицательном – к правой части дерева. Таким образом, внутренний узел дерева является узлом проверки определенного условия. Далее идет следующий вопрос и т.д., пока не будет достигнут конечный узел дерева, являющийся узлом решения, т.е. листом дерева. Для каждой конкретной задачи существуют свои типы конечного узла в количестве n+1.

Модель, представленная в виде дерева решений, является интуитивной и упрощает понимание решаемой задачи. Это свойство деревьев решений не только важно при отнесении к определенному классу нового объекта, но и полезно при интерпретации модели классификации в целом. Дерево решений позволяет понять и объяснить, почему конкретный объект относится к тому или иному классу. Деревья решений дают возможность извлекать правила из базы данных на естественном языке.

Деревья решений позволяют создавать классификационные модели в тех областях, где аналитику достаточно сложно формализовать знания. Алгоритм конструирования дерева решений не требует от пользователя выбора входных атрибутов (независимых переменных). На вход алгоритма можно подавать все существующие атрибуты, алгоритм сам выберет наиболее значимые среди них, и только они будут использованы для построения дерева.

Точность моделей, созданных при помощи деревьев решений, сопоставима с другими методами построения классификационных моделей (статистические методы, нейронные сети). Разработан ряд масштабируемых алгоритмов, которые могут быть использованы для построения деревьев решения на сверхбольших базах данных.

Большинство алгоритмов конструирования деревьев решений имеют возможность специальной обработки пропущенных значений. Многие классические статистические методы, при помощи которых решаются задачи классификации, могут работать только с числовыми данными, в то время как деревья решений работают и с числовыми, и с категориальными типами данных.

Многие статистические методы являются параметрическими, и пользователь должен заранее владеть определенной информацией, например, знать вид модели, иметь гипотезу о виде зависимости между переменными, предполагать, какой вид распределения имеют данные. Деревья решений, в отличие от таких методов, строят непараметрические модели. Таким образом, деревья решений способны решать такие задачи Data Mining (раскопки данных), в которых отсутствует априорная информация о виде зависимости между исследуемыми данными.

На сегодняшний день существует большое число алгоритмов, реализующих деревья решений: CART, C4.5, CHAID, CN2, NewId, ITrule и другие. Деревья с произвольным числом потомков, как одна из разновидностей деревьев решений, благодаря своим свойствам являются более универсальным методом решения задач классификации. Именно такой алгоритм выбран в качестве области исследования.

Алгоритм C4.5

Алгоритм C4.5 моделирует дерево решений с неограниченным количеством ветвей у узла. Данный алгоритм может работать только с дискретным зависимым атрибутом и поэтому может решать только задачи классификации. C4.5 считается одним из самых известных и широко используемых алгоритмов построения деревьев классификации.

Для работы алгоритма C4.5 необходимо соблюдение следующих требований:

Каждая запись набора данных должна быть ассоциирована с одним из предопределенных классов, т.е. один из атрибутов набора данных должен являться меткой класса. Классы должны быть дискретными. Каждый пример должен однозначно относиться к одному из классов. Количество классов должно быть значительно меньше количества записей в исследуемом наборе данных. Алгоритм C4.5 медленно работает на сверхбольших и зашумленных наборах данных.

<u>Псевдокод Algorithm C4-5</u>

1. Input: database D
2. Tree = {}
3. If D is pure OR other stopping criteria Then

4. Terminate
5. End If
6. For all attribute a ϵ D do
7. Entropy for root
8. End For
9. abest = Best attribute entropy
10. Tree = Create a node then test a best in the root
11. Dv = sud_datasets from D
12. For all Dv do
13. Tree v= C4.5(Dv)
14. End for
15. Return Tree

Математическое обеспечение алгоритма

Задача алгоритма заключается в построении иерархической классификационной модели в виде дерева из множества примеров *T*. Процесс построения дерева будет происходить сверху вниз. Сначала создается корень дерева, затем потомки корня и т.д.

На первом шаге дерево пустое (имеется только корень) и исходное множество *T* (ассоциированное с корнем). Требуется разбить исходное множество на подмножества. Для этого выбрать один из атрибутов в качестве проверки. Тогда в результате разбиения получаются *n* (по числу значений атрибута) подмножеств и, соответственно, создаются n потомков корня, каждому из которых поставлено в соответствие свое подмножество, полученное при разбиении множества *T*. Затем эта процедура рекурсивно применяется ко всем подмножествам (потомкам корня) и т.д.

Очевидно, что из m (по числу атрибутов) возможных вариантов, требуется выбрать самый подходящий. Некоторые алгоритмы исключают повторное использование атрибута при построении дерева. Любой из атрибутов можно использовать неограниченное количество раз при построении дерева.

Пусть мы имеем проверку *X* (в качестве проверки может быть выбран любой атрибут), которая принимает n значений A_1, A_2 … A_n. Тогда разбиение *T* по проверке *X* даст нам подмножества T_1, T_2 … T_n, при *X*, равном соответственно A_1, A_2 … A_n. Единственная доступная информация – то, каким образом классы распределены во множестве *T* и его подмножествах, получаемых при разбиении по *X*. Именно это играет роль при определении критерия.

Пусть $freq(C_i, S)$ – количество примеров из некоторого множества *S*, относящихся к одному и тому же классу C_j. Тогда вероятность того, что случайно выбранный пример из множества *S* будет принадлежать к классу C_j, такова:

$$P = \frac{freq(C_i, S)}{|S|}.$$

Согласно теории информации, количество содержащейся в сообщении информации зависит от ее вероятности

$$log_2\left(\frac{1}{P}\right). \quad (1)$$

Выражение дает оценку среднего количества информации, необходимого для определения класса примера из множества *T*. В терминологии теории информации выражение

$$Info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} * log_2\left(\frac{freq(C_j, T)}{|T|}\right) \quad (2)$$

называется энтропией множества T.

Ту же оценку, но только уже после разбиения множества T по X, дает выражение:

$$Info_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * Info(T_i). \quad (3)$$

Тогда критерием для выбора атрибута будет являться формула:

$$Gain(X) = Info(T) - Info_x(T). \quad (4)$$

Критерий (4) считается для всех атрибутов. Выбирается атрибут, максимизирующий данное выражение. В случае с числовыми атрибутами следует выбрать порог, с которым должны сравниваться все значения атрибута. Пусть числовой атрибут имеет конечное число значений $\{v_1, v_2 \ldots v_n\}$, предварительно отсортированное. Тогда любое значение, лежащее между v_i и v_{i+1}, делит все примеры на два множества: те, которые лежат слева от этого значения $\{v_1, v_2 \ldots v_i\}$, и те, что справа $\{v_{i+1}, v_{i+2} \ldots v_n\}$. В качестве порога можно выбрать среднее между значениями v_i и v_{i+1}:

$$TH_i = \frac{v_i + v_{i+1}}{2}.$$

Оценка сложности алгоритма решения задачи. Параллелизм

Алгоритм решающего дерева имеет сложность, линейную по длине выборки. Если множество предикатов настолько богато, что на шаге всегда находится предикат, разбивающий выборку S на непустые подмножества S_0 и S_1, то алгоритм строит бинарное решающее дерево, безошибочно классифицирующее выборку X. Параллелизм (Concurrency) – это свойство систем, при котором несколько вычислений выполняются одновременно, и при этом, возможно, взаимодействуют друг с другом. Вычисления могут выполняться на нескольких ядрах одного чипа с вытесняющим разделением времени потоков вычислений на одном процессоре, либо выполняться на физически отдельных процессорах.

Алгоритм деревьев решений построен на индукции, является примером поглощающего «жадного» алгоритма и на сегодняшний день наиболее распространенной стратегией деревьев решений для данных, но это не только стратегия. Как отмечалось ранее, общая схема построения дерева принятия решений выглядит следующим образом.

Выбирается очередной атрибут, помещается в корень. Затем для всех его значений рекурсивно строится дерево в этом потомке. Начиная со второго шага, алгоритм может быть распараллелен (рис. 1).

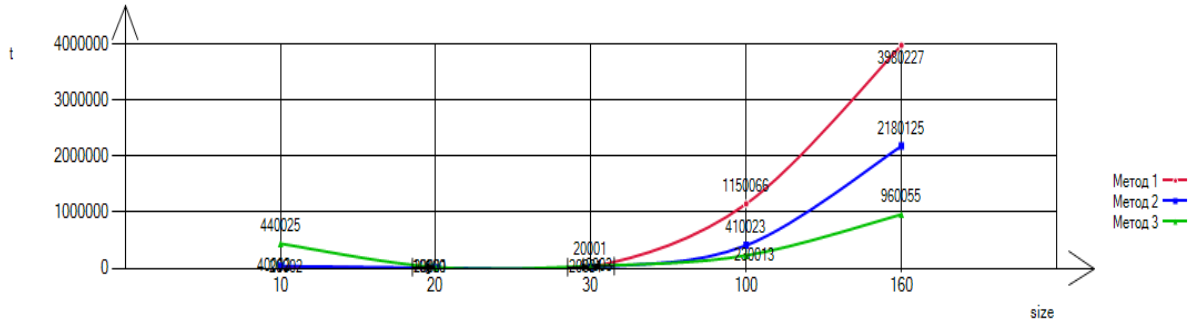


Рис. 1. Зависимость времени работы алгоритма от объема обучающей выборки (t, тик / size)

Метод 1 – Линейные вычисления (t, тик / size, количество строк шт. в базе)

Метод 2 – Распараллеливание в один поток (блокирование внешних потоков) (t, тик / size, количество строк шт. в базе)

Метод 3 – Полностью распараллеленный алгоритм на 2 потока (t, тик / size, количество строк шт. в базе).

Анализ полученных результатов распараллеливания

Из рис. 1 видно, что распараллеливание алгоритма деревьев решений имеет смысл проводить только на больших объемах данных. Для случаев, где выборка данных меньше 50 элементов, распараллеливание избыточно, линейный вариант алгоритма работает быстрее. Это связано с затратами приложения на процесс создания и работы с потоками. Распараллеливание в 1 поток значительно проще реализуется, однако дает результат, превосходящий линейный алгоритм, а значит, также является применимым для реализации данной задачи на практике. В результате, уже при объеме базы более ста строк алгоритм в распараллеленной версии позволяет экономить минуты. С увеличением базы этот результат будет подтверждаться, а разрыв во времени работы методов увеличиваться.

Выводы

Деревья решений – иерархическое, гибкое средство предсказания принадлежности объектов к определенному классу или прогнозирования значений числовых переменных. Среди большого числа разнообразных алгоритмов классификации деревья решений отличаются тем, что позволяют проводить интеллектуальный анализ данных, т.е. представлять в явном и доступном для понимания и интерпретации экспертами причинно-следственные связи в процессе отнесения объекта классификации к тому или иному классу.

Задача формирования дерева решений представляет собой оптимизационную задачу. Качество работы рассмотренного метода деревьев решений зависит как от выбора алгоритма, так и от набора исследуемых данных. Несмотря на все преимущества данного метода, следует помнить, что для того чтобы построить качественную модель, необходимо понимать природу взаимосвязи между зависимыми и независимыми переменными и подготовить достаточный набор данных.

Сравнивая алгоритм C4-5 с другими методами, можно отметить, что алгоритм деревьев решений превосходит по эффективности современные алгоритмы классификации. Кроме того, он является инструментом интеллектуального анализа данных для обнаружения скрытых знаний. В ходе апробации алгоритма на практических задачах классификации получены деревья решающих правил, представляющие интерес для специалистов в соответствующих проблемных областях.

Литература

1. Чубуков И.А. Data Mining. Учебный курс – [http://www.intuit.ru/department/database/datamining/].

2. Биллиг В.А. Основы программирования на C#. Бином. Лаборатория знаний, Интернет-университет информационных технологий. Серия: Основы информационных технологий, 2009. – [http://www.intuit.ru/department/se/prcsharp08/].

3. Шилдт Г. Теория и практика C#. СПб.: BHV-Санкт-Петербург, 1996. – 312 с.

4. Breiman, L., Friedman, J., Olshen, R., and Stone, C. Classification and Regression Trees. Wadsworth International Group, Belmont, CA. 1984.

5. Joydeep Ghosh and Alexander Liu, K-Means, The Top Ten Algorithms in Data Mining.

6. Dan Steinberg, CART: Classification and Regression Trees, The Top Ten Algorithms in Data Mining.

РАЗРАБОТКА САЙТА И МОДУЛЯ ПРОВЕРКИ ТЕСТОВ СИСТЕМЫ WCTEST ДИСТАНЦИОННОГО ТЕСТИРОВАНИЯ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ

Т.Н. Линева, Т.Д. Макарова

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Развитие и внедрение информационных технологий оказывает сильное воздействие на жизнь людей. Одна из передовых идей в области использования персональных компьютеров – реализация разнообразных способов обучения.

Особый интерес в настоящий момент вызывают системы дистанционного обучения. Их сущность заключается в том, что преподаватель и студенты территориально отделены друг от друга, но при этом они могут находиться в постоянном взаимодействии, организованном с помощью особых приемов построения учебного процесса, форм контроля, методов коммуникации с использованием современных информационных и телекоммуникационных технологий.

На данный момент большая часть специальностей вузов предусматривает обучение студентов программированию. Базовые предметы, на которых учат основам написания компьютерных программ, – информатику или программирование – изучают многие студенты. В задачах на программирование часто требуется написать программу на одном из языков программирования, которая считывает входные данные из указанного файла (или со стандартного потока ввода) в определённом формате, обрабатывает их в соответствии с постановкой задачи и выводит результат в определённом формате в указанный файл (или на стандартный поток вывода). При дистанционном обучении возникают некоторые трудности в проверке таких заданий. При очной форме обучения преподаватель, проверяющий прикладные компьютерные программы, может попросить студента объяснить соответствие алгоритма работы его программы с исходным текстом. При дистанционной форме обучения преподаватель, проверяющий подобную прикладную программу, не имеет возможности услышать комментарии студента и проверяет ее только по результатам работы программы. Один из признанных способов проверки правильности такого рода программ – запуск проверяемой программы на наборе тестов.

Процесс проверки работы программ можно автоматизировать. Одно из основных преимуществ использования автоматизированных тестирующих систем: компьютер проверяет решения быстро и качественно. Еще один плюс такой проверки – полная беспристрастность, все решения проверяются одинаково.

1. Постановка задачи

Цель работы состоит в повышении эффективности проверки знаний и практических навыков студентов по программированию на языке C++. Поставленная цель достигается с помощью системы автоматизированного тестирования задач по программированию.

Для достижения поставленной цели были решены следующие задачи:

1) анализ проблемы дистанционного тестирования задач по программированию;

2) разработка и создание сайта системы WCTEST;
3) разработка нового типа теста задач по программированию;
4) модификация проекта классов проверки задач по программированию;
5) расширение модуля тестирования задач по программированию;
6) интеграция классов проверки студенческих решений с другими компонентами системы;
7) разработка новых тестов для проверки знаний студентов.

2. Система дистанционного тестирования WCTEST

На рис. 1 представлена схема работы сайта системы дистанционного тестирования WCTEST. При сохранении общей архитектуры и функциональности система WCTEST реализована в виде WEB приложения, а также добавлен новый тип задания – тест-класс: «Написать объявление и описание класса».

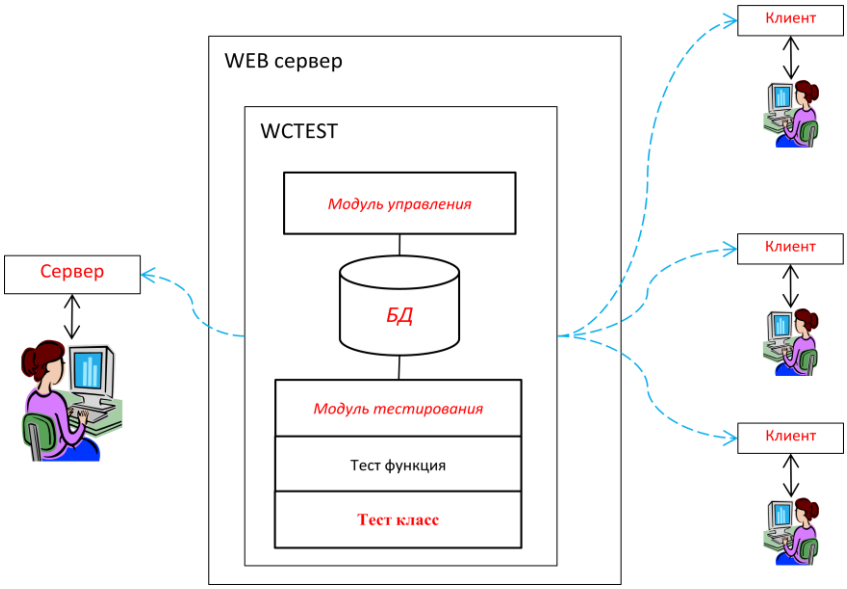


Рис. 1. Схема работы сайта системы WCTEST

3. Архитектура системы

Архитектура системы включает в себя: основные понятия, информационную модель и функциональную модель. В данной работе были введены такие понятия как: тестовое задание, тестовое решение, описание теста, тестовый набор. Информационная модель системы содержит основные наборы данных и представлена тремя подразделами, содержащими:

- описание тестов;
- описание результатов тестирования;
- описание заданий на тестирование.

Функциональная модель включает в себя описание всех действий по тестированию. При этом дистанционное тестирование задач должно рассматриваться как процесс, включающий следующие основные этапы:

- подготовка тестирования;
- проведение тестирования;
- анализ результатов тестирования.

4. Структура сайта

На рис. 2 представлена структура сайта.

Рис. 2. Структура сайта

Сайт состоит из двух основных разделов – для преподавателя и для студента.

Раздел для преподавателя включает следующие страницы:

- страницы регистрации и авторизации пользователей;
- страница управления аккаунтом, на которой можно изменить пароль;
- главная страница, содержащая общую информацию о сайте;
- страница студенческих групп, на которой можно:
  – добавлять, удалять и редактировать списки групп,
  – загружать список групп из файла,
  – скачивать список логинов и паролей студентов;
- страница тем с тестами, на которой можно:
  – добавлять, удалять, редактировать темы и задания на функцию или класс,
- страница проведения тестирования, на которой можно:
  – назначить и проверить тестирование,
  – удалить информацию обо всех назначавшихся ранее тестированиях.

Раздел для студента включает следующие страницы:

- страница авторизации;
- страница управления аккаунтом, на которой можно изменить пароль;
- главная страница, содержащая список назначенных заданий с описанием каждого задания и форму для отправки решений.

4. Схема проверки задач по программированию

На рис. 3 показана схема проверки задач по программированию.

Входными параметрами для подсистемы тестирования задач по программированию являются эталонное решение, студенческие решения и тестовые наборы.

Вначале проводится компиляция эталонного решения с созданием исполняемого файла, затем для всех тестовых наборов генерируются правильные ответы. Следующим этапом является компиляция студенческих решений также с созданием исполняемого файла и генерация ответов для всех тестовых наборов. Для каждого студенческого решения проводится сравнение сгенерированных ответов с ответами эталонного решения и формируется запись в файле результатов тестирования.

Рис. 3. Схема проверки задач по программированию

Таким образом, итогом работы подсистемы тестирования задач по программированию является таблица с результатами тестирования всех студенческих решений.

5. Описание тестового задания

1) Функция

- Формулировка задания
- Эталонное решение
  - Файл *.cpp
  - Тестовые наборы данных
- Файл оценок

2) Класс

- Формулировка задания
- Эталонное решение
  - <ClassName>.h
  - <ClassName>.cpp
  - Main.cpp
  - Тестовые наборы данных
- Файл оценок

6. Апробация системы

Апробация системы была проведена со студентами первого курса группы 8103 в присутствии преподавателей С.Н. Карпенко и Н.В. Шестаковой.

В тестировании принимали участие 8 студентов. На выполнение заданий было отведено 45 минут. При появлении загруженных заданий преподаватель нажимал кнопку «Проверить все назначенные тесты» и на экран выводились результаты проверки задания в виде оценок.

По итогам работы преподавателя с системой был сделан ряд пожеланий:

- создание единой базы групп;
- создание единой базы тем;
- объединение разделов добавления и редактирования заданий;
- добавление всплывающих окон подтверждения (например, «закрыть раздел редактирования без сохранения изменений?» при попытке закрыть этот раздел, предварительно не сохранив произведенные изменения);
- возможность просматривать решения студента;

- возможность просматривать одновременно решение студента и решение преподавателя;
- возможность просматривать ошибки, возникшие при компиляции решения студента;
- в таблице с оценками за тест для каждого теста указывать максимально возможную оценку;
- возможность назначать время начала и окончания тестирования, т.е. время, когда студентам будет доступен список назначенных им заданий и когда они могут отправлять свои решения;
- возможность сортировки списка студентов по алфавиту.
- возможность проверки студенческого решения на плагиат;
- возможность проверки качества кода решения.

часть предложений была реализована, а именно:

- в таблице с оценками за тест для каждого теста указывается максимально возможная оценка.

Остальные пожелания к работе системы оставляют поле деятельности для будущего развития программы.

7. Заключение

Основные результаты работы:

- Создан сайт системы WCTEST в архитектуре «клиент-сервер».
- Выполнены необходимые модификации в модуле управления и базе данных.
- Разработан новый тип теста – проверка задач на классы.
- Расширен модуль тестирования задач по программированию. Реализация выполнена в среде MS VisualStudio 2010 на языке C#.
- Проведена интеграция модуля тестирования с другими компонентами системы.
- Разработаны тесты для проверки знаний студентов для нового и существующего типов теста на различные темы.
- Произведена апробация системы, по итогам которой преподавателем был высказан ряд пожеланий:
  - реализовать возможность просмотра решений и результатов компиляции;
  - формирование ведомости тестирования.

А также были предложены следующие направления развития системы:

- анти-плагиат,
- качество кода теста.

Литература

1. Петренко А., Бритвина Е., Грошев С., Монахов А., Петренко О. Тестирование на основе моделей // Открытые системы, №09/2003.
2. Информационный портал «Дистанционное обучение» – [http://www.distance-learning.ru/].
3. Майерс Г. Искусство тестирования программ. М.: Финансы и статистика, 1982.
4. Материалы IV Международной конференции по дистанционному образованию – [http://www.iet.mesi.ru/broshur/broshur.htm].
5. Компания Competentum. Электронное обучение. Системы дистанционного обучения – [http://www.competentum.ru].
6. Бахтизин В.В., Глухова Л.А. Стандартизация и сертификация программного обеспечения: Учеб. пособие. Минск.: БГУИР, 2006. 200 с.

ОБЗОР МЕТОДОВ ПЕРЕУПОРЯДОЧЕНИЯ СИММЕТРИЧНЫХ РАЗРЕЖЕННЫХ МАТРИЦ

А.Ю. Малова

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача уменьшения заполнения факторов разреженных матриц. Приводится обзор методов переупорядочения матриц. Основное внимание уделяется методу минимальной степени, методу вложенных сечений и их модификациям. Дается обзор программного обеспечения, решающего указанную задачу.

Введение

Одной из актуальных задач алгебры разреженных матриц является решение систем линейных алгебраических уравнений $Ax = b$ с разреженной симметричной положительно определенной матрицей A. Для решения разреженных СЛАУ применяются прямые и итерационные методы. Прямой метод основан на факторизации матрицы системы (для симметричных положительно определенных матриц – разложение Холецкого) с последующим решением треугольных систем. Одна из основных особенностей данного подхода – возрастание числа ненулевых элементов матрицы в процессе факторизации. Это влечет за собой вычислительные трудности, которые могут повлиять на возможность и эффективность решения данной задачи на ЭВМ: необходимо выделять дополнительную память для хранения новых ненулевых элементов, с ростом их числа значительно увеличивается время, необходимое для факторизации матрицы. Кроме того, сильное заполнение влечет за собой накопление вычислительных ошибок, что может повлиять на саму возможность использования прямого метода.

Для решения данной проблемы к исходной матрице применяется процедура перестановки ее строк и столбцов с целью уменьшения заполнения фактора. Из всех $n!$ перестановок для матрицы порядка n одно или более обеспечивают минимальное заполнение и несколько упорядочений минимизируют число арифметических операций при факторизации. Нахождение таких перестановок является NP-полной задачей [51], и на практике применяются эвристические методы, использующие представление исходной матрицы A в виде графа.Основные методы переупорядочения, позволяющие снизить заполненность симметричной разреженной матрицы, можно разделить на две группы – подходы, основанные на методе вложенных сечений, и подходы, основанные на методе минимальной степени. Для ленточных матриц существует специальный класс алгоритмов типа Катхилла-Макки, которые оставим за рамками обзора.

В данной работе рассмотрены указанные методы переупорядочения симметричных разреженных матриц и их модификации, дано сравнение методов. Также приведен обзор программных библиотек для переупорядочения матриц и разделения графов. Результаты работы используются на кафедре математического обеспечения ЭВМ факультета ВМК ННГУ при разработке нового прямого решателя для больших разреженных систем линейных алгебраических уравнений [53, 54].

Метод вложенных сечений

Метод вложенных сечений (*nested dissection, ND*) был предложен А. Джорджем [8] в 1973 г. для решения матричных задач, возникающих в конечно-разностных и конеч-

но-элементных приложениях. В 1978 г. Джордж и Лю [11] обобщили алгоритм для задач с нерегулярной сеткой. Метод основан на многократном разбиении графа матрицы при помощи вершинных разделителей. На каждом шаге алгоритма некоторым образом выбирается разделитель графа, его вершины нумеруются и удаляются из графа. При этом граф распадается на две или более связные компоненты. Затем данная процедура повторяется рекурсивно для каждой новой связной компоненты графа, до тех пор, пока не будут пронумерованы все вершины. При таком подходе заполнение локализуется в областях матрицы, соответствующих выделенным подграфам и разделителю. Метод вложенных сечений для регулярной $k * k$ сетки дает минимальное заполнение – $O(k^2 \log k)$.

Модификации метода различаются, в первую очередь, способами выделения разделителя и связных подграфов. Так, в методе *автоматических вложенных сечений* (automatic nested dissection) [11] разделитель выбирается из середины структуры уровней смежности графа, построенной с корнем в псевдопериферийной вершине. Заполнение фактора матрицы после работы алгоритма автоматических вложенных сечений составляет $O(n \log n)$, где n – число вершин графа.

Липтон, Роуз и Тарьян в 1979 г. предложили *обобщенный метод вложенных сечений* (generalized nested dissection) [32]. Их алгоритм основан на теореме о $\sqrt{n}$-разделителе и дает заполнение фактора матрицы $O(n \log n)$. Теоретически метод был исследован в работах Липтона и Тарьяна [33], Гильберта и Тарьяна [15].

Для разбиения графа в ряде модификаций используется спектральная информация о его матрице. Потен, Симон и соавторы в 1992 г. предложили *спектральный метод вложенных сечений* (spectral nested dissection) [42], особенностью которого является алгоритм спектрального разделения графа [41]. В данном подходе граф разделяется по средней компоненте собственного вектора, соответствующего второму по величине собственному числу матрицы Лапласа графа. Самой трудоемкой операцией при этом является вычисление необходимого собственного вектора.

Группа методов вложенных сечений использует геометрическую информацию о вершинах. Метод *картезианских вложенных сечений* (cartesian parallel nested dissection) [19] был предложен Хифом и Рагхаван в 1991 г. Они использовали двумерные декартовы координаты для нахождения разделителя. В этом случае граф разделяется по значению одной из координат. *Геометрический метод* разделения графа был предложен Миллером, Тэнгом и соавторами в 1993 г. [36], а затем применен Гильбертом, Миллером, Тэнгом в 1994 г. [13]. В отличие от метода картезианских вложенных сечений, авторы использовали координаты вершин графа в d-мерном пространстве, спроецированные на поверхность $d + 1$-мерной сферы. Для нахождения разделителя вычисляется центральная точка, разделяющая исходное множество точек на две примерно равные части, а разделитель определяется как множество вершин, близких к проекции произвольного $d + 1$-мерного круга на исходную d-мерную поверхность. Недостатком геометрического подхода является его вычислительная трудоемкость при определении центральной точки. Некоторые эвристики для практического применения были рассмотрены в работе [13].

За последние годы были предложены модификации метода вложенных сечений, основанные на многоуровневой процедуре разделения графа с комбинированными техниками нахождения разделителя. Большинство из них используют специализированные алгоритмы для разделения графов (graph partitioning) и улучшения разделения (partition refinement). На каждом шаге переупорядочения многоуровневым методом поиск оптимального разделителя выполняется в три этапа.

- *Огрубление графа (coarsening)*: с помощью процедуры паросочетания вершин строится последовательность графов $G_1, G_2, \dots, G_m$ таких, что $|V| > |V_1| > \dots > |V_m|$.

- *Разделение графа (partitioning)*: поиск разделителя и выделение двух несвязных компонент наименьшего огрубленного графа G_m.
- *Развертывание графа (uncoarsening)*: разделение графа G_m проектируется на исходный граф Gчерез последовательность графов $G_{m-1}, G_{m-2}, \dots, G_1$. Во время этого процесса выполняется улучшение проецируемого разделение графа.

Такая схема позволяет находить достаточно хорошее разделение графа за приемлемое время, т.к. трудозатратные алгоритмы применяются к небольшим по размеру графам. Многоуровневый метод вложенных сечений был применен в работах Кариписа и Кумара [26], Эшкрафта и Лю [2], Хендриксона и Леланда [20], Хендриксона и Ротберга [23], Гупта [17] и др.

Для выполнения огрубления графа применяются различные методы *паросочетания (matching)*. Среди наиболее используемых техник нахождения паросочетаний – метод случайных паросочетаний (randommatching), паросочетание тяжелых ребер (heavy edge matching), паросочетание легких ребер (light edge matching), паросочетание тяжелых клик (heavy clique matching) [26]. Сложность таких алгоритмов составляет $O(|E|)$.

После огрубления графа выполняется разделение наименьшего графа G_m на две примерно равные части, затем выделяется вершинная или реберная граница разбиения. По типу используемого аппарата в алгоритмах для разбиения графов можно выделить три группы: спектральные, геометрические и комбинаторные. Подробный обзор методов разделения графа представлен в работе Потена [40].

Среди комбинаторных методов широко применяется группа алгоритмов, основанных на итерационном улучшении разбиения. Впервые такой подход был предложен Керниганом и Лином [31] в 1970 г., а затем модифицирован Фидуччиа и Матэйсесом [7] в 1982 г. за счет использования специальных структур данных. Модификация алгоритма Кернигана–Лина для вершинного разделителя была представлена Эшкрафтом и Лю [2] в 1996 г. На каждом шаге итерационных алгоритмов оценивается качество полученного разбиения и размер разделителя, а также целесообразность перемещения вершин из разделителя в одну из частей графа. Временная сложность одной итерации алгоритма Кернигана–Лина составляет $O(|E|\log|E|)$, для алгоритмов Фидуччиа–Матфейсеса и Эшкрафта–Лю эта оценка уменьшена до $O(|E|)$ при использовании соответствующих структур данных. Подобные алгоритмы преимущественно находят реберный разделитель графа, для определения вершинного разделителя затем находят наименьшую вершинную оболочку реберного разделителя.

Другие комбинаторные методы разделения графа основаны на «наращивании» подграфа вокруг некоторой вершины. К таким методам относятся алгоритм выращивания графа (graph growing algorithm, GGP) и жадный алгоритм выращивания графа (greedy graph growing algorithm, GGGP), использующий специальные веса. Их основным недостатком является чувствительность к выбору начальной вершины для «наращивания» подграфа.

На этапе развертывания графа на каждом шаге применяются локальные техники улучшения проецируемого разделения. Для этого обычно используются итерационные методы (модификации алгоритма Кернигана–Лина), где в качестве начального приближения выбрана проекция разделения предыдущего графа.

Метод минимальной степени

Одним из распространенных методов переупорядочения матрицы является *метод минимальной степени* (*minimum degree, MD*), предложенный Тиннеем и Уолкером в 1969 г. [47]. Этот алгоритм является симметричным аналогом алгоритма Марковица [35] (1957 г.) и основан на локальной стратегии уменьшения заполнения фактора мат-

рицы. Теоретические основы метода минимальной степени приведены в работах Роуза [43, 44].

Метод минимальной степени в некотором смысле моделирует процесс симметричного Гауссова исключения. Вершина, исключаемая из графа на каждом шаге алгоритма, соответствуют выбору ведущего элемента в строках и столбцах, обеспечивающих внесение наименьшего числа ненулевых элементов. В процессе упорядочения строится последовательность *графов исключения* (*elimination graphs*), в которой каждый следующий граф получен из предыдущего удалением вершины с минимальной степенью и созданием клики между всеми смежными с ней вершинами.

В такой постановке наиболее затратными операциями являются преобразование графа и пересчет степеней. Одно из улучшений алгоритма, использующее структуру только исходного графа, основано на использовании достижимых множеств [52]. Также для представления графов исключения используются фактор-графы (quotient graphs, Джордж, Лю, [9]), супер-элементы (superelements, Дафф, Рэйд [5]). Эти подходы основаны на слиянии связанных исключенных вершин в супер-узлы. Ряд модификаций алгоритма, направленных на одновременное исключение последовательности вершин (*mass elimination*), был предложен Джорджем и Лю [9, 10] в 1980 г. В этих работах авторы ввели понятие *неразличимых вершин* (*indistinguishable nodes*) – вершин, смежные множества которых совпадают.

Существует ряд эвристик для сокращения вычислений при пересчете степеней. Например, Эйзенстат и соавторы [6] в 1977 г. использовали неполное обновление степеней (incomplete degree update). Они показали, что если смежное множество вершины u включено в смежное множество вершины v, то степень вершины v не нужно вычислять, пока не будет исключена вершина u (в этом случае вершина v называется *превосходимой* (*outmatched*)).

В 1985 г. Лю предложил *множественный метод минимальной степени* (*multiple minimum degree, MMD*) [34], основанный на одновременной нумерации всех возможных вершин минимальной степени до этапа пересчета степеней. Такой подход ухудшает качество получаемой перестановки, однако позволяет уменьшить время вычислений за счет отложенного пересчета степеней вершин. Лю предложил использовать в алгоритме внешние степени вершин вместо их действительных степеней. Этот прием уменьшает размер клики, создаваемой после удаления вершины с минимальной степенью.

Одним из активно развивающихся подходов к модификации метода минимальной степени является использование оценок степеней вершин вместо их значений. *Приближенный метод минимальной степени* (approximate minimum degree, AMD) был предложен Амстеем, Дэвисом и Даффом [1] в 1996 г. Данный прием также использовали Гильберт, Молер, Шрейбер [14], Дэвис, Гильберт, Ларимор и Нг [4] и др.

Подробный обзор модификаций метода минимальной степени представлен в работе Джорджа, Лю [12]. Отметим, что метод минимальной степени дает наименьшее заполнение фактора, если граф матрицы был деревом, в других случаях заполнение не минимально.

Обзор программного обеспечения

Мировым лидером среди программных пакетов для разделения графов и переупорядочения разреженных матриц является группа библиотек METIS, разработанная в Университете Миннесоты (США). В пакете METIS [25] для переупорядочения матриц и разделения графов реализована последовательная версия многоуровневого метода вложенных сечений, основанная на работах [26, 28]. В пакете ParMETIS [30] содержится параллельная MPI-версия многоуровневого метода вложенных сечений, основанная на работах [27, 29, 45].

Другая широко используемая академическая разработка – библиотека Scotch [39] и ее параллельная версия PT-Scotch [3] от лаборатории LaBRI Университета Бордо (Франция). Библиотеки предназначены для разделения графов и гиперграфов, переупорядочения разреженных матриц. Для переупорядочения матриц используется комбинированный подход [38]. Сравнение работы пакетов PT-Scotch, ParMETIS и WSMP [18] можно найти в работе [16].

К другим известным библиотекам относятся Chaco и JOSTLE. Библиотека Chaco [22], предназначенная для разделения графов, была разработана в лаборатории Sandia National Laboratories (США). В ней реализованы модификации многоуровневого и спектрального методов вложенных сечений на основе работ [20, 21, 23]. Программный комплекс JOSTLE [48], разработанный в Гринвичском университете (Великобритания), включает средства для последовательного и параллельного многоуровневого разделения графов. Особенности реализации многоуровневой схемы основаны на работах авторов [49, 50].

В библиотеках для решения систем линейных уравнений с разреженной матрицей, как правило, имеются встроенные методы для переупорядочения, а также обеспечивается интерфейс для подключения специализированных библиотек. Например, в коммерческом пакете для разреженных матриц MKL Pardiso, входящем в состав библиотеки Intel® MKL [24], используется оптимизированная версия метода вложенных сечений из библиотеки METIS, а также собственная реализация множественного метода минимальной степени. Один из ведущих академических параллельных прямых решателей, MUMPS [37], содержит реализацию ряда модификаций метода минимальной степени (приближенный, приближенный с выбором квазиплотной строки (QAMD), метод минимизации заполнения (approximate minimum fill-in, AMF)). Также библиотека имеет интерфейс для использования переупорядочения из пакета PORD [46], распространяемого вместе с MUMPS, и внешних библиотек METIS, SCOTCH, ParMETIS, PT-SCOTCH.

Заключение

Задача нахождения переупорядочения, минимизирующего заполнение фактора Холецкого матрицы, является NP-сложной [51]. Для ее решения разработаны различные эвристические методы, среди которых широко применяются классы методов вложенных сечений и методов минимальной степени. Рассмотренные подходы различаются стратегией обработки матрицы: метод минимальной степени использует локальную стратегию уменьшения заполнения, а метод вложенных сечений – глобальную.

В настоящее время в практических приложениях широко используется подход, основанный на комбинировании метода вложенных сечений и метода минимальной степени. Чаще всего сначала процесс переупорядочения ведется методом вложенных сечений, а когда размер оставшихся подграфов будет достаточно малым, к ним применяется метод минимальной степени. Это позволяет использовать достоинства обоих алгоритмов: сбалансированность разделения графа вложенными сечениями и локальное уменьшение заполнения методом минимальной степени. Иногда множество разделителей, полученных на одном шаге метода вложенных сечений, затем переупорядочивается методом минимальной степени. Комбинированный подход был использован, в частности, в работах Хендриксона и Ротберга [23], Пеллигрини, Ротберга и Амстея [38], Гупта [17].

В случае параллельных вычислений большую роль играет структура порождаемого дерева исключения, построенного по портрету фактора матрицы. Предпочтительным является портрет с «коротким и широким», а не «узким и длинным» деревом исключения, поскольку первое оставляет большие ресурсы для параллелизма. Такой портрет

характерен для метода вложенных сечений. Дерево исключения, порождаемое при переупорядочении методом минимальной степени, имеет более «узкую и длинную» структуру. На практике этот метод чаще применяют для небольших матриц. В параллельных решателях преимущественно используют модификации метода вложенных сечений, чаще всего – многоуровневого. Это позволяет найти приемлемое по качеству и перспективам последующего распараллеливания переупорядочение за относительно небольшое время.

Работа выполнена в лаборатории ITLab ННГУ в рамках ФЦП при финансовой поддержке Минобрнауки России, гос. соглашение № 14.B37.21.0878.

Литература

1. Amestoy P.R., Davis T., Duff I. An approximate minimum degree ordering algorithm // SIAM. J. on Matrix Anal. Appl. – 1996. – Vol. 17, No. 4. – P. 886–905.
2. Aschcraft C., Liu J.W.H. A partition improvement algorithm for generalized nested dissection // Technical Report BCSTECH-92-020, Boeing computer Services. – 1994.
3. Chevalier C., Pellegrini F. PT-SCOTCH: a tool for efficient parallel graph ordering // Parallel Computing. – 2008. – Vol. 34, No. 6–8. – P. 318–331.
4. Davis T.A., Gilbert J.R., Larimore S., Ng E. A column approximate minimum degree ordering algorithm // Transactions on Mathematical Software. – 2004. – Vol. 30, No. 3. – P. 353–376.
5. Duff I.S., Reid J.K. The multifrontal solution of indefinite sparse symmetric linear equations // ACM Trans. Math. Software. – 1983. Vol. 9, No. 3. – P. 302–325.
6. Eisenstat S.C., Gursky M.C., Schultz M.H., Sherman A.H. Yale sparse matrix package, I: The symmetric codes. // International J. on Numerical Methods Engineering. – 1982. – Vol. 18. – P. 1145–1151.
7. Fiduccia C.M., Mattheyses R.M. A linear time heuristic for improving network partitions // Proceedings 19th IEEE Design Automation Conference. – 1982. P. 175–181.
8. George A. Nested dissection of a regular finite element mesh // SIAM J. on Numerical Analysis. – 1973. – Vol. 10, No. 2. – P. 345–363.
9. George A., Liu J.W.H. A minimal storage implementation of the minimum degree algorithm // SIAM J. Numerical Analysis. – 1980. – Vol. 17. – P. 282–299.
10. George A., Liu J.W.H. A fast implementation of the minimum degree algorithm using quotient graphs // ACM Trans. Math. Software. – 1980. – Vol. 6. – P. 337–358.
11. George A., Liu J.W.H. An automatic nested dissection algorithm for irregular finite element problems // SIAM J. on Numerical Analysis. – 1978. – Vol. 15, No. 5. – P. 1053–1069.
12. George A., Liu J.W.H. The evolution of the minimum degree ordering algorithm // SIAM Review. – 1989. – Vol. 31, No. 1. – P. 1–19.
13. Gilbert J.R., Miller G.L., Teng S.-H. Geometric mesh partitioning: Implementation and experiments // Technical Report CSL-94-13, Xerox Palo Alto Research Centre. – 1994.
14. Gilbert J.R., Moler C., Schreiber R. Sparse matrices in MATLAB: design and implementation // SIAM J. Matrix Anal. Applic. – 1992. – Vol. 13. – P. 333–356.
15. Gilbert J.R., Tarjan R.E. The analysis of a nested dissection algorithm // Numerische Mathematik. – 1987. – Vol. 50. – P. 377–404.
16. Gupta A. An evaluation of parallel graph partitioning and ordering software on a massively parallel computer // Technical Report RC 25008(W1006-029), IBM T.J. Watson Research Center. – 2010.

17. Gupta A. Fast and effective algorithms for graph partitioning and sparse matrix reordering // IBM Journal of Research and Development. – 1996. – Vol. 41, No. (1/2). – P. 171–183.
18. Gupta A. WSMP: Watson sparse matrix package (Part-I: Direct solution of symmetric sparse systems) // Technical Report RC 21886, IBM T. J. Watson Research Center. – 2000. URL: [http://www.cs.umn.edu/˜agupta/wsmp].
19. Heath M.T., Raghavan P. A Cartesian parallel nested dissection algorithm // SIAM J. on Matrix Analysis and Applications. – 1995. – Vol. 16, No. 1. – P. 235–253.
20. Hendrickson B., Leland R. A multilevel algorithm for partitioning graphs // Technical Report SAND93-1301, Sandia National Laboratories. – 1993.
21. Hendrickson B., Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations // SIAM J. on Scientific Computing. – 1995. – Vol. 16, No. 2. – P. 452–469.
22. Hendrickson B., Leland R. The Chaco user's guide: Version 2.0 // Technical Report SAND94–2692, Sandia National Laboratories. – 1994. – [http://www.sandia.gov/~bahendr/papers/guide.ps].
23. Hendrickson B., Rothberg. Improving the runtime and quality of nested dissection ordering // SIAM J. on Scientific Computing. – 1999. – Vol. 20. – P. 468–489.
24. Intel Math Kernel Library Reference Manual. URL: [http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].
25. Karipis G. METIS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.0. // Technical report, University of Minnesota, Department of Computer Science and Engineering. – 2011. URL: [http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf]
26. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM J. on Scientific Computing. – 1999. – Vol. 20, No. 1. – P. 359–392.
27. Karypis G., Kumar V. A Parallel algorithm for multilevel graph partitioning and sparse matrix ordering // J. of Parallel and Distributed Computing. – 1998. – Vol. 48. – P. 71–85.
28. Karypis G., Kumar V. Multilevel k-way partitioning scheme for irregular graphs // J. of Parallel and Distributed Computing. – 1998. – Vol. 48, No. 1. – P. 96–129.
29. Karypis G., Kumar V. Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs // SIAM Review. – 1999. – Vol. 41, No. 2. – P. 278–300.
30. Karypis G., Kumar V. ParMETIS: Parallel graph partitioning and sparse matrix ordering library // Technical Report TR 97-060, Department of Computer Science, University of Minnesota. – 1997.
31. Kernighan B.W., Lin S. An efficient heuristic procedure for partitioning graphs // The Bell System Technical Journal. – 1970. – Vol. 29. – P. 291–307.
32. Lipton R.J., Rose D.J., Tarjan R.E. Generalized nested dissection // SIAM J. on Numerical Analysis. – 1979. – Vol. 16, No. 2. – P. 346–358.
33. Lipton R.J., Tarjan R.E. Applications of a planar separator theorem // SIAM J. on Computing. – 1980. – Vol. 9, No. 3. – P. 615–627.
34. Liu J.W.H. Modification of the minimum-degree algorithm by multiple elimination // ACM Trans. Math. Software. – 1985. – Vol. 11, No. 2. – P. 141–153.
35. Markowitz H.M. The elimination form of the inverse and its application to linear programming // Management Science. – 1957. – Vol. 3. – P. 255– 269.
36. Miller G.L., Teng Sh.-H., Vavasis S.A. A unified geometric approach to graph separators // Proceedings of 31st Annual Symposium on Foundations of Computer Science. – 1991. – P. 538–547.

37. MUltifrontal Massively Parallel Solver (MUMPS 4.10.0) User's guide // Technical report ENSEEINT-IRIT. – 2011. URL: [http://mumps.enseeiht.fr/doc/userguide_4.10.0.pdf].
38. Pellegrini F., Roman J., Amestoy P. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering // Concurrency: Practice and Experience. – 2000. – Vol. 12. – P. 68–84.
39. Pellegrini F. Scotch and libScotch 5.1 User's Guide // Technical Report LaBRI, Universit´e Bordeaux I. – 2010. – [http://hal.archives-ouvertes.fr/docs/00/41/03/27/PDF/scotch_user5.1.pdf].
40. Pothen A. Graph partitioning algorithms with applications for scientific computing // Technical Report Old Dominion University Norfolk, VA, USA. – 1997.
41. Pothen A., Simon H.D., Liou K.-P. Partitioning sparse matrices with eigenvectors of graphs // SIAM J. of Matrix Analysis and Applications. – 1990. – Vol. 11, No. 3. – P. 430–452.
42. Pothen A., Simon H.D., Wang L. Spectral nested dissection // Technical Report 92-01, Computer Science Department, Pennsylvania State University. – 1992.
43. Rose D.J. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations // Graph Theory and Computing. Read R.C., ed. New York: Academic Press, 1973. – P. 183–217.
44. Rose D.J. Symmetric elimination on sparse positive definite systems and the potential flow network problem // PhD thesis, Applied Math., Harvard Univ. – 1970.
45. Schloegel K., Karypis G., Kumar V. Parallel multilevel algorithms for multi-constraint graph partitioning // Proceedings of Euro-Par – 2000. – P. 296–310.
46. Schulze J. Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods // Bit Numerical Mathematics. – 2001. – Vol. 41, No. 4. – P. 800–841.
47. Tinney W., Walker J. Direct solutions of sparse network equations by optimally ordered triangular factorization // Proceedings of the IEEE. – 1967. – Vol. 55, No. 11. – P. 1801–1809.
48. Walshaw C., Cross M. JOSTLE: parallel multilevel graph-partitioning software – an overview // In F. Magoules, editor, Mesh Partitioning Techniques and Domain Decomposition Techniques, Civil-Comp Ltd. – 2007. – P. 27–58.
49. Walshaw C., Cross M. Mesh partitioning: a multilevel balancing and refinement algorithm // SIAM J. on Scientific Computing. – 2000. – Vol. 22, No. 1. – P. 63–80.
50. Walshaw C., Cross M. Parallel optimization algorithms for multilevel mesh partitioning // Parallel Computing. – 2000. – Vol. 26, No. 12. – P. 1635–1660.
51. Yannakakis M. Computing the minimum fill-in is NP-complete // SIAM J. on Algebraic and Discrete Methods. – 1981. – Vol. 2, No. 1. – P. 77–79.
52. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. – М.: Мир, 1984. – 333 с.
53. Козинов Е.А., Лебедев И.Г., Лебедев С.А., Малова А.Ю. и др. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей // Вестник Нижегородского государственного университета. Н. Новгород: Изд-во ННГУ. 2012. Вып. 5(1). (В печати)
54. Козинов Е.А., Лебедев И.Г., Лебедев С.А. и др. Разработка прямого решателя для разреженных систем линейных уравнений с симметричной положительно определенной матрицей // Параллельные вычислительные технологии (ПаВТ'2012): Труды Международной научной конференции (Новосибирск, 26 – 30 марта 2012 г.) – Челябинск: Издательский центр ЮУрГУ, 2012. – С. 525–530. – [http://pavt.susu.ru/2012/short/043.pdf].

АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНОГО ЯДРА ДЛЯ СПЕЦВЫЧИСЛИТЕЛЕЙ НА БАЗЕ FPGA

Ю.А. Милюкин

Юго-Западный госуниверситет, Курск

Рассматриваются методы адаптивных вычислений. Вычисления, при которых аппаратная архитектура вычислителя подстраивается под структуру заданного вычислительного процесса, может быть основана на математической модели вычислительного процесса и её математического образа, задающего двумерную топологию вычислительного ядра. Приведен алгоритм распараллеливания вычислений, позволяющий распознавать по маске математические формулы.

Введение

Характеристики современных ПЛИС и структурных ASIC, включающих встроенные процессоры и блоки памяти, позволяют реализовать значительное число приложений в виде системы на программируемом кристалле.

FPGA представляет собой микросхему высокого уровня интеграции, содержащую во внутренней области матрицу функциональных блоков и систему их межсоединений, размещенную между столбцами матрицы, а в периферийной области – блоки ввода-вывода.

Современные FPGA предоставляют встроенную память и встроенные логические блоки для арифметических операций. Преимущество встроенных логических блоков – лучшая скорость и пространство. Кроме того, встроенная память проще, чем внешний интерфейс памяти. FPGA часто связаны с микропроцессорами. Также основное из преимуществ встроенных микропроцессоров – уменьшение задержки связи между микропроцессором и FPGA.

Основным достоинством классических FPGA является то, что при проведении разработки и в процессе отладки у разработчика имеется возможность многократно загружать проект в микросхему и проверять результаты своей работы непосредственно на «живом» изделии. При этом стоимость отладки любого этапа проекта невелика. Недостатком работы классических FPGA является сама система загрузки данных в микросхему. Во-первых, для работы таких FPGA необходимо внешнее устройство, в котором хранится файл инициализации. Во-вторых, информация, загружаемая в FPGA, может быть скопирована и использована для получения пиратских копий разрабатываемого изделия. Защититься от такого копирования принципиально невозможно, поскольку информация о конфигурации передается от одной микросхемы к другой через внешние выводы микросхем по линиям связи, расположенным на печатной плате. Есть и еще один существенный недостаток у классических FPGA, правда, он не так часто бросается в глаза. Во время работы, особенно в такой аппаратуре, которая работает в долговременном непрерывном режиме без перезагрузки, могут происходить частичные сбои конфигурации. Определить такой сбой не всегда возможно. Исправить частичные сбои конфигурации можно только полной перезагрузкой конфигурации, так как частичная загрузка конфигурации в классических FPGA невозможна.

Задача заключается в том, чтобы разработать технологию автоматического конструирования вычислительного ядра для заданной в символьном виде вычислительной задачи.

FPGAs непосредственно не программируются. Синтез инструментов преобразует код в поток битов, который загружается в память FPGA. Как правило, для настройки устройства используются языки описания оборудования (VHDL и Verilog). Основная проблема программирования FPGA состоит в том, что и VHDL, и Verylog (то есть языки программирования аппаратуры) предполагают побитовую обработку информации. Но также есть возможность использовать языки высокого уровня (C++). Реализация на языке C++ позволяет обработку информации с заданной разрядностью (8, 16, 24, 32 и т.д.), кратной байту, что является более предпочтительным для автоматического конструирования вычислительного ядра.

На сегодняшний день задача автоматического конструирования топологии вычислительного ядра частично решена только для аналоговых интегральных схем Anadigm Designer® 2 на аппаратной платформе FPAA.

1. Конструирование ядра через кортеж и маску

Для автоматизации конструирования спецвычислительных систем на программируемом кристалле необходимо решить две задачи:

1) синтез топологии многопроцессорного вычислительного ядра (вычислителя), в котором осуществлено распараллеливание вычислений под ту или иную вычислительную задачу;

2) создание программного кода, доставляющего решение вычислительной задаче на синтезированном вычислителе.

Такие задачи были поставлены по причине проблем, возникающих при практическом использование этих платформ:

1) производители «бортовых» вычислителей и/или их компонентов выпускают эти изделия с заведомо избыточной функциональной структурой. При этом разработчикам, как правило, известен математический функционал, подлежащий технической реализации, и, как следствие, известна функциональная структура вычислителя, которая может быть заведомо задана производителю под заказную СБИС, что влечёт за собой дальнейшую миниатюризацию воплощения её в «железе»;

2) не меньшую трудность у разработчиков вызывает собственно программирование выбранной для технической реализации встраиваемой системы, поскольку, например, электрическая принципиальная схема современного аналогового вычислителя может иметь размеры футбольного стадиона, аналогично цифровой вычислитель требует сотен (и даже тысяч) страниц программного кода.

Для решения возникающих проблем был предложен автоматический синтез топологии вычислительного ядра встраиваемой системы по заданному вычислительному процессу на заданной аппаратной платформе.

Физическая постановка задачи адаптивных вычислений:

Дано:

1) математическая модель заданного вычислительного процесса

$$u(a_1, a_2, \dots, a_n) = \sum_{i=1}^{k} \pm c_i \prod_{j=1}^{n} \varphi_{i,j,\alpha}(a_j^{m_j}), \quad (1)$$

где u – выход вычислительного процесса; a_i $(i = \overline{1,n})$ – входы вычислительного процесса (сигналы с датчиков в аналоговой и/или цифровой форме); c_i – коэффициенты усиления (числа); $\varphi_{i,j,\alpha}$ – обозначение операции взятия функции, где α – обозначение класса функции ($\alpha = 0$ – алгебраические функции, $\alpha = 1$ – тригонометрические функции, $\alpha = 2$ – показательные функции и т.д.); m_j – обозначение математической операции ($m_j = 1$ – деление; $m_j = p$ – возведение в степень $p > 0$; $m_j = {}^{1}/_{p}$ – извлечение арифметического корня степени $p > 0$; $m_j = (p)$ – операция взятия производной степени p; $m_j = \{p\}$ – операция взятия интеграла кратности p.

2) заданная аппаратная платформа, на которой процесс (1) нужно реализовать.

Требуется автоматически сконструировать топологию вычислительного ядра, реализующую заданный вычислительный процесс (1) на заданной аппаратной платформе.

Для решения задачи неравенство меняется на его математический образ, формализующийся двумя понятиями: кортежем и маской. Тем самым предложенный образ вычислительного процесса задаёт двумерную топологию многопроцессорного вычислительного ядра организации параллельных вычислений слагаемых заданной алгебраической суммы, каждое из которых вычисляется в результате последовательного перемножения результатов выполнения математических операций, формализуемых соответствующей строкой маски.

Можно провести процесс распараллеливания вычисления, распознав содержащиеся в маске математические формулы (если они в ней есть) (рис. 1).

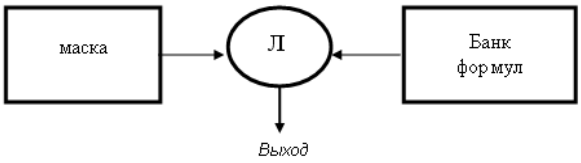


Рис. 1. Функциональная схема алгоритма автоматического распознавания математических формул

Здесь Л – логический блок для работы с банком формул, представляющим собой набор масок распознаваемых формул.

В случае графического представления технической реализации математических операций необходимо рассматривать вычислительное ядро с распараллеливанием вычислений, которое будет реализовывать вычислительный процесс (1). Такая топология является отправной точкой для разработки интерпретатора (конвертера) для преобразования элементов маски в эквивалентные им фрагменты технических устройств в интегральном исполнении, реализующих ту или иную математическую операцию (сложение, вычитание, умножение, деление, возведение в степень, извлечение арифметического корня заданной степени, дифференцирование, интегрирование, а также взятие действительной функции одной переменной).

Для синтезирования электрической принципиальной схемы бортового вычислителя с использованием обозначений пакета по программированию аналоговых интегральных схем Anadigm Designer® 2 на аппаратной платформе FPAA, позволяющего «прошивать» FPAA с использованием алфавита команд разработчиков Anadigm, будем применять рефлексивную семантику преобразования модели математической операции в модель технической реализации. Другими словами интерпретировать один формализованный язык (язык математических операций) в категории другого языка (языка технической реализации математических операций в аналоговой и/или цифровой форме). Техническая реализация математической операции в аналоговой форме, как правило, представляется посредством электрической принципиальной схемы.

Некоторые из электрических принципиальных схем, автоматически сформированных описанной в [6] программой-конвертером языка кортежа и маски в обозначения пакета Anadigm Designer® 2, то есть в файл с расширением *.cpp, который затем компилируется внешним C++-компилятором (платный пакет производства компании Anadigm), предназначенным для «физической прошивки» FPAA, приведены на рис. 2-8.

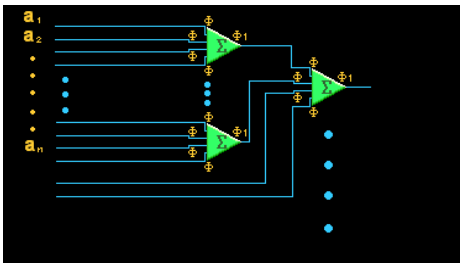


Рис. 2. Электрическая принципиальная схема реализации маски (сумматора)

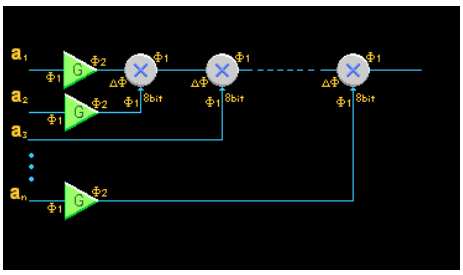


Рис. 3. Электрическая принципиальная схема реализации строки маски
(умножителя в общем случае *n*-сигналов с коэффициентом усиления, отличным от 1)

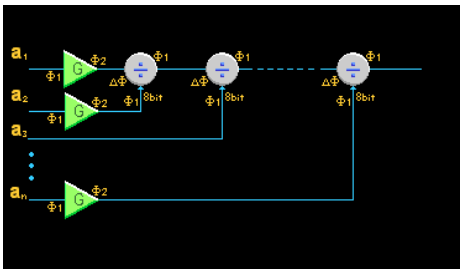


Рис. 4. Электрическая принципиальная схема реализации строки маски
(делителя в общем случае *n*-сигналов с коэффициентом усиления, отличным от 1)

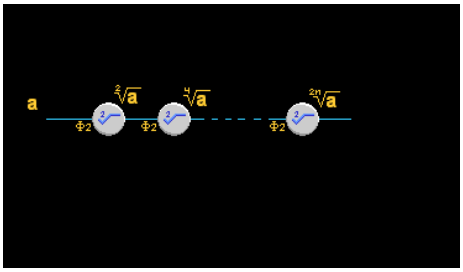


Рис. 5. Электрическая принципиальная схема взятия арифметического корня

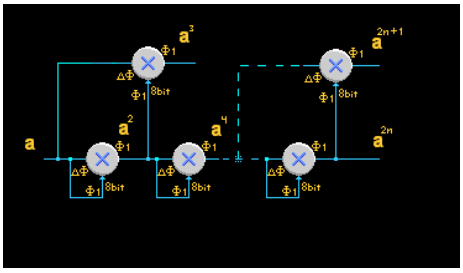


Рис. 6. Электрическая принципиальная схема возведения в степень

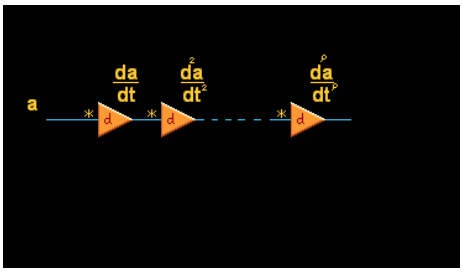


Рис. 7. Электрическая принципиальная схема взятия производной степени ρ

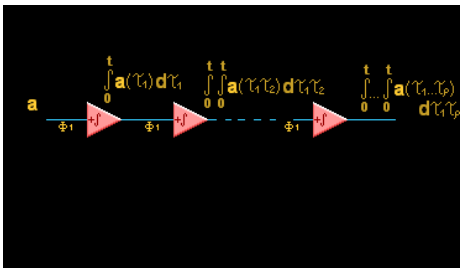


Рис. 8. Электрическая принципиальная схема взятия ρ-кратного интеграла

Литература

1. Подчукаев В.А., Милюкин Ю.А., Филонович А.В. Методы адаптивных вычислений и техническая реализация современных систем автоматического управления // Сборник научных статей по материалам III Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях». Курск, 2011. С. 6.
2. Подчукаев В.А., Милюкин Ю.А., Филонович А.В., Кутуев А.Н. Автоматическое проектирование интегральных микросхем // Сборник научных статей по материалам II Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях».Курск, 2011. С. 137.
3. Подчукаев В.А., Милюкин Ю.А., Филонович А.В., Кутуев А.Н. Вычислительный кластер как аналог его функциональной схемы // Сборник научных статей по материалам II Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях». Курск, 2011. С. 142.
4. Применение программируемых логических интегральных схем в электронной аппаратуре // Электроника и системы управления, 2009. №4.
5. Подчукаев В.А., Кулаков К.А. Аналитическое проектирование топологии интегральных микросхем // Матер. Междунар. науч.-техн. конф. «Мехатроника, автоматизация, управление – 2007». Таганрог: Изд-во ТТИ ЮФУ, 2007. С. 260-264.
6. Подчукаев В.А., Шевченко Д.С. Автоматическое проектирование технической реализации законов управления на платформе программируемых аналоговых интегральных схем производства компании Anadigm // Цифровые системы управления и обработки информации: приложение к журналу «Мехатроника, автоматизация, управление». 2008. № 7. С. 7-11.

ДИНАМИЧЕСКИЙ ХАОС КАК БЕСКОНЕЧНЫЙ РАЗРЫВ ВТОРОГО РОДА В КООРДИНАТАХ ЦЕНТРА СМЕЩЕНИЯ

Ю.А. Милюкин, С.В. Суворов, В.А.Подчукаев

Юго-Западный госуниверситет, Курск

Поставлена и решена задача аналитического конструирования по заданной математической модели динамической системы в пространстве состояний соответствующей ей математической модели в фазовом пространстве. Показано, что изображающая точка всякого решения динамической системы общего вида в пространстве состояний принадлежит гиперсфере со смещённым центром в фазовом пространстве. Полученный результат, вскрывающий причины динамического хаоса, требует динамической компьютерной визуализации, связывающей якобы случайные эффекты фазовых портретов с точками разрыва в компонентах вектора центра смещения, порождающих эти эффекты.

Для простоты изложения ограничимся случаем динамических систем, описываемых обыкновенными однородными дифференциальными уравнениями в форме Коши:

$$\dot{x} = P(x,t)x,\ \ x \in R^n,\ \ t \in [0,\infty),\ \ t_0 \geq 0,\ \ x(t_0) = x_0, \qquad (1)$$

где x – вектор состояний; x_0 – вектор начальных условий; t – время; $P(x, t)$ – заданная функциональная матрица, элементы которой определены и непрерывны вместе со своими частными производными по x и t в пространстве состояний $R^n \times [0,\infty)$, что гарантирует существование единственного решения $x(t, t_0, x_0)$, проходящего через точку (t_0, x_0), не являющуюся состоянием равновесия.

Будем полагать, что решение $x(t, t_0, x_0)$ детерминированной системы, описываемой уравнением (1) (подстановка которого в (1) обращает его в тождество), известно. Проблема хаоса была идентифицирована в процессе построения (по известным решениям в пространстве состояний $R^n \times [0,\infty)$) фазовых портретов этих решений (в фазовом пространстве R^n). Собственно фазовое пространство не содержит оси времени t, в то время как изображающая точка самого фазового портрета однозначно определяется текущими значениями компонент соответствующего решения $x(t, t_0, x_0)$. Классические примеры динамического хаоса позволяют говорить о «странном» (случайном, непредсказуемом и т.д.) поведении фазовых портретов детерминированных систем. Ясно, что «странный» характер такого поведения проистекает из перехода от пространства размерности $(n+1)$, каковым является пространство состояний $R^n \times [0,\infty)$, к фазовому пространству R^n размерности n, в явном виде не содержащему времени, а включающему его лишь опосредованно посредством изображающих точек решения, зависящих от времени.

В дальнейшем уравнение (1) будем называть математической моделью детерминированной системы в пространстве состояний $R^n \times [0,\infty)$. Отметим, что общепринятой математической модели той же системы, но уже в фазовом пространстве R^n, на единицу меньшей размерности, до сих пор нет.

Сконструируем такую модель, исходя из математической модели в пространстве состояний.

Первый шаг в этом направлении сделан в монографии [2], с. 100 (лемма). Полученный там результат сформулируем в виде следующего утверждения: изображающая точка всякого решения $x(t,t_0,x_0)$ уравнения (1) с кососимметрической матрицей $P(x,t)$ принадлежит гиперсфере

$$x^T(t,t_0,x_0)x(t,t_0,x_0)=x_0^T x_0 \tag{2}$$

в фазовом пространстве R^n.

Предложенный в [2] алгоритм перехода от математической модели (1) в пространстве состояний $R^n\times[0,\infty)$ к математической модели (2) в фазовом пространстве R^n включает следующие операции:

1) умножение обеих частей уравнения (1) слева на транспонированный вектор состояний x^T;

2) применение к полученному в п.1 результату следующего тождества для кососимметрических матриц [3], свойство 10.61:

$$x^T P(x,t)x\equiv 0, \tag{3}$$

которое позволяет представить этот результат в виде

$$x^T\frac{dx}{dt}=0; \tag{4}$$

3) умножение обеих частей уравнения (4) на dt;

4) интегрирование полученного в п.3 результата в пределах от t_0 до t;

5) замены в полученном равенстве $x(t)=x(t,t_0,x_0)$, $x(t_0)=x_0$ и представлении полученного результата в каноническом виде гиперсферы (2) [4].

Следующий шаг по переходу от математических моделей динамических систем в пространстве состояний $R^n\times[0,\infty)$ (с матрицами $P(x,t)$ общего вида) к математическим моделям этих систем в фазовом пространстве R^n осуществлён в [5], §2 (теорема).

В указанной статье отправной точкой послужило каноническое представление функциональной матрицы $P(x,t)$ общего вида в виде суммы симметрической и кососимметрической матриц

$$P(x,t)\equiv\frac{P(x,t)+P^T(x,t)}{2}+\frac{P(x,t)-P^T(x,t)}{2},$$

с учётом которого уравнение (1) можно записать следующим образом:

$$\dot{x}=\left[\frac{P(x,t)+P^T(x,t)}{2}+\frac{P(x,t)-P^T(x,t)}{2}\right]x.$$

Применение к полученной таким образом математической модели динамической системы (в пространстве состояний $R^n\times[0,\infty)$) описанного выше алгоритма перехода к математической модели в фазовом пространстве R^n позволяет сформулировать результат в виде следующего утверждения: изображающая точка всякого решения уравнения (1) $x(t,t_0,x_0)$ общего вида принадлежит гиперсфере со смещённым центром

$$x^T(t,t_0,x_0)x(t,t_0,x_0)+2G^T[x(t,t_0,x_0),t]x(t,t_0,x_0)=x_0^T x_0, \tag{5}$$

которую можно представить в виде уравнения центральной гиперсферы переменного радиуса

$$\begin{aligned}&\{x(t,t_0,x_0)+G[x(t,t_0,x_0),t]\}^T\{x(t,t_0,x_0)+G[x(t,t_0,x_0),t]\}=\\&\qquad=x_0^T x_0+G^T[x(t,t_0,x_0),t]G[x(t,t_0,x_0),t].\end{aligned} \tag{6}$$

Следует отметить, что в отличие от случая кососимметрической функциональной матрицы, использование описанного выше алгоритма при операции интегрирования (п. 4) в пределах от t_0 до t применяется к равенству

$$x^T dx = x^T \frac{P(x,t)+P^T(x,t)}{2} xdt .$$

При этом результат интегрирования правой части этого равенства в соответствии с канонической формой представления гиперквадрик общего вида [4, с. 340] должен быть представлен посредством следующей билинейной формы:

$$-2G^T[x(t,t_0,x_0),t]x(t,t_0,x_0) = \int_{t_0}^{t} x^T(\tau)\left[P(x,\tau)+P^T(x,\tau)\right]x(\tau)d\tau. \qquad (7)$$

Из сравнения (5) с (6) видно, что вектор $G[x(t,t_0,x_0),t]$ определяет координаты центра смещения.

В дальнейшем (5) будем называть математической моделью динамической системы общего вида (1) в фазовом пространстве R^n.

Таким образом, в обоих случаях геометрическим местом изображающих точек решения динамической системы в фазовом пространстве является гиперсфера. Для кососимметрических матриц – это гиперсфера постоянного радиуса $\sqrt{x_0^T x_0}$, а в случае функциональных матриц $P(x,\ t)$ общего вида – гиперсфера переменного радиуса $\sqrt{x_0^T x_0 + G^T[x(t,t_0,x_0),t]G[x(t,t_0,x_0),t]}$.

Будем рассматривать (7) как уравнение, в котором вектор состояний и интеграл правой части известны, а вектор $G[x(t,t_0,x_0),t]$ подлежит определению, то есть с позиций линейной алгебры задача определения этого вектора не доопределена, и, как следствие, её решение неединственно. Поэтому какого-либо конструктивизма к рассматриваемой проблеме результат, зафиксированный в виде (7), не прибавляет, хотя сам вектор $G[x(t,t_0,x_0),t]$ может рассматриваться в качестве нового инструмента исследований.

Наполним этот вектор конструктивным содержанием.

Полученный результат

Принимая во внимание исходное уравнение (1), запишем правую часть (7) в виде

$$\int_{t_0}^{t} x^T(\tau)[P(x,\tau)+P^T(x,\tau)]x(\tau)d\tau = \int_{t_0}^{t}\left[x^T(\tau)\frac{dx(\tau)}{d\tau}+\frac{dx^T(\tau)}{d\tau}x(\tau)\right]d\tau =$$

$$= \int_{t_0}^{t} 2x^T(\tau)dx(\tau) = x^T(\tau)x(\tau)\Big|_{t_0}^{t} =$$

$$= x^T(t,t_0,x_0)x(t,t_0,x_0) - x_0^T x_0 = \sum_{i=1}^{n}\left[x_i^2(t,t_0,x_0)-x_{0,i}^2\right] = \sum_{i=1}^{n}\frac{x_i^2(t,t_0,x_0)-x_{0,i}^2}{x_i(t,t_{0,}x_0)}x_i(t,t_0,x_0) .$$

Отметим, что полученное выражение сконструировано исходя из условия, что для всех

$$x_i(t,t_0,x_0) \neq 0 \ (i=\overline{1,n})$$

справедливо тождество

$$\frac{x_i(t,t_{0,}x_0)}{x_i(t,t_{0,}x_0)} \equiv 1 .$$

Подобное представление позволяет единственным образом выделить искомый вектор из билинейной формы левой части (7) в виде

$$G[x(t,t_0,x_0),t] = -\frac{1}{2}col\left[x_1(t,t_0,x_0) - \frac{x_{0,1}^2}{x_1(t,t_0,x_0)},...,x_n(t,t_0,x_0) - \frac{x_{0,n}^2}{x_n(t,t_0,x_0)}\right]. \qquad (8)$$

Факт получения (8) достаточно красноречив, позволяя связать динамический хаос с бесконечными разрывами второго рода [6, с. 221] в отдельных компонентах вектора G при переходе через ноль знаменателей соответствующих компонент вектора решения уравнения (1). Другими словами, управление хаосом осуществляется посредством вектора центра смещения G, не бравшимся в рассмотрение в многочисленных исследованиях динамического хаоса ранее. Таким образом, указанные точки разрыва являются своего рода «выколотыми» точками центра смещения, влекущими за собой «случайный» характер поведения фазового портрета после прохождения этих точек. Отметим, что «выколотые» точки вряд ли могут быть зафиксированы соответствующим выбором шага интегрирования при численных методах построения фазовых портретов.

Полученный результат, вскрывающий причины динамического хаоса, требует динамической компьютерной визуализации, связывающей якобы случайные эффекты фазовых портретов с точками разрыва в компонентах вектора центра смещения, порождающих эти эффекты.

Литература

1. Кузнецов С.П. Динамический хаос. – М: Физматлит, 2006. – 294 с.
2. Подчукаев В.А. Аналитические методы теории автоматического управления. – М.: Физматлит, 2002. – 256 с.
3. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. – М.: Наука, 1984. – 320 с.
4. Пензов Ю.Е. Аналитическая геометрия. – Саратов: Изд-во Сарат. ун-та, 1972. – 364 с.
5. Подчукаев В.А., Звягина А.С. Новое доказательство гипотезы Ж.А. Пуанкаре // Доклады Академии военных наук. 2009. № 5(40). С. 115–123.
6. Бронштейн И.Н., Семендяев К.А. Справочник по математике для инженеров и учащихся втузов. – М.: Наука, 1986. – 544 с.

АРХИТЕКТУРА ПАРАЛЛЕЛЬНОЙ СУБД НА ПЛАТФОРМЕ GPU-КЛАСТЕРА

Р.Ш. Минязев

Казанский национальный исследовательский технический университет им. А.Н. Туполева

E-mail: r.minyazev@gmail.com

Предлагается архитектура параллельной СУБД, функционирующей на платформе GPU-кластера, каждый вычислительный узел утяжелен двумя GPU-акселераторами. На GPU выполняются операции SELECT, PROJECT, JOIN. База данных делится горизонтально с использованием операции хеширования по ключевому полю каждого отношения. Каждый GPU-акселератор хранит в своей памяти свой фрагмент базы данных. Вычислительный узел хранит все фрагменты исходной базы данных в своей оперативной памяти. Рассматривается случай работы с консервативными данными, объем которых соизмерим с объемом оперативной памяти вычислительного узла.

Постановка задачи

Серьезной проблемой для высокопроизводительных параллельных реляционных СУБД является размещение данных между узлами и повышение скорости обработки сложных запросов с большим числом операции соединения [1, 2]. Один из перспективных подходов к повышению производительности (как уменьшения времени обработки запросов) СУБД – использование GPU-акселераторов, для исполнения наиболее трудоемких реляционных операций [3]. Рассматривается случай консервативных СУБД (преобладают операции чтения данных без модификации), ориентированных на аналитическую обработку накопленных данных. Для таких систем характерно исполнение множества сложных запросов с большим числом операций соединения [1].

Предлагаемая архитектура параллельной СУБД, получившая название Clusterix-G, представлена на рис. 1. Система функционирует на платформе GPU-кластера. База данных (БД) делится горизонтально путем хеширования по ключевому полю на 2n частей. При этом каждый вычислительный узел хранит всю расхешированную БД целиком. Рассматривается случай работы с базами данных, объем которых сравним с объемом оперативной памяти, установленной в вычислительном узле.

В системе выделяется два типа логических узлов: узел управления MGM и рабочий узел NODE. На узле управления функционируют следующие программные модули:

- MONITOR – получение запросов от пользователей и передача обратно результатов обработки, управление всеми другими модулями системы. Все поступающие запросы собираются во внешней очереди, откуда передаются модулю TRANS;
- TRANS – транслирует запрос в пакет подзапросов для исполнения на рабочем узле NODE, сообщает о готовности очередного пакета модулю MONITOR;
- SORT – получает частичные результаты от модулей IO-JOIN, формирует из них конечный результат в виде отношения, в котором устраняет дубликаты и над которым при необходимости выполняет операции сортировки и агрегации. Результат передается модулю MONITOR для отправки пользователю.

На рабочем узле NODE функционирует модуль IO-JOIN. Он получает оттранслированный пакет запросов от MONITOR (одинаковый для каждого модуля IO-JOIN),

помещает его в свою внутреннюю очередь в виде набора подзапросов, выбирает очередной подзапрос из внутренней очереди и запускает два одинаковых ядра (KERNEL_A, KERNEL_B) на GPU-акселераторах. Каждое ядро (функция) выполняет подзапрос над своей частью исходной БД.

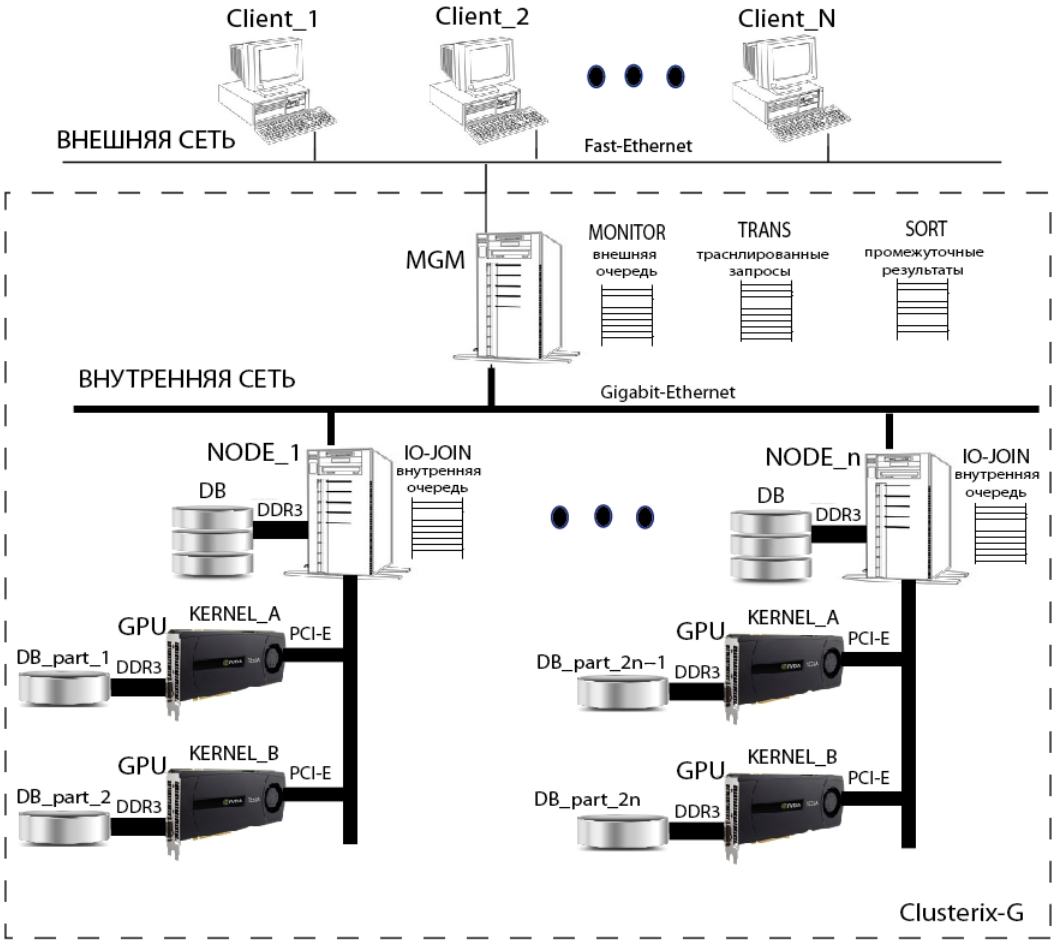


Рис.1. Архитектура параллельной СУБД Clusterix-G

Организация работы GPU

Полученный SQL запрос пользователя транслируется модулем TRANS в пакет простых SQL подзапросов согласно регулярному плану обработки (рис.2) [4]. Правомерность его использования показана, в частности, в работе [5].

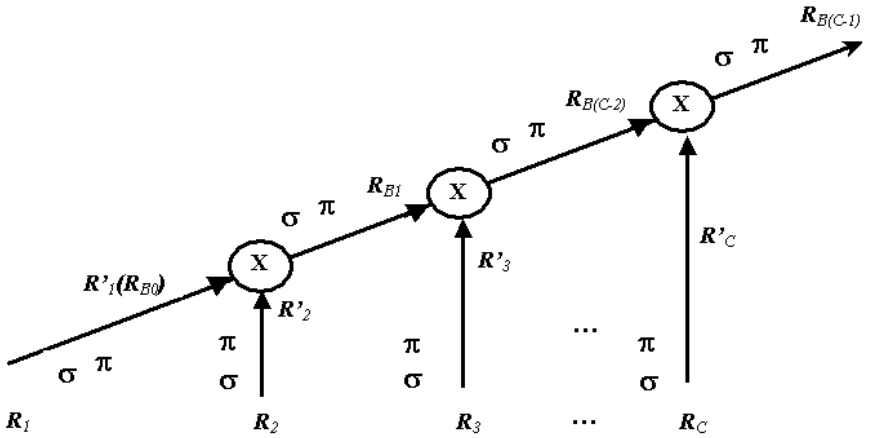


Рис. 2. Регулярный план обработки запроса

Промежуточное отношение R_i` является результатом проведения операций селекции ($\sigma$) и проекции ($\pi$) над исходным отношением $R_i$; C – количество отношений БД, задействованных в текущем запросе. Временное отношение $R_{Bj}$ получается при соединении (x) временного $R_{B(j-1)}$ и промежуточного $R_{(j+1)}$` отношений с последующим применением к ним операций проекции; R_{Bj} формируется в результате исполнения подзапроса (ядра) и сохраняется в памяти GPU, после исполнения следующего подзапроса оно может быть удалено. Результат последовательного выполнения всех ядер (функции) на GPU в виде временного отношения $R_{B(C-1)}$ передается из модуля IO-JOIN в SORT.

При выполнении операции соединения ($R_{B(j-1)}$ x $R_{(j+1)}$`) внутри ядра в память GPU последовательно подкачиваются части расхешированного отношения $R_{j+1}$ из БД, хранящейся в оперативной памяти узла, при этом подкачиваются только те поля, которые определены в $R_{(j+1)}$`. Каждый кортеж отношения хранится в памяти GPU в виде структуры, поля структур соответствуют атрибутам отношения, отношение в целом состоит из массива таких структур.

Каждый вычислительный узел кластера имеет два GPU-акселератора (см. рис. 1). В модуле IO-JOIN запускается два потока, каждый из которых контролирует работу своего GPU. Ядра KERNEL_A и KERNEL_B работают асинхронно и независимо друг от друга, каждое обрабатывает свою часть исходной БД. Суммарное число потоков, на котором запускается i-е ядро KERNEL ($i \in \{\overline{1,C}\}$), определяется числом кортежей в отношении Ri, каждый поток обрабатывает свой кортеж.

Обсуждение

Использование GPU-акселераторов обладает большим потенциалом повышения производительности (как уменьшения времени обработки сложных запросов) параллельных СУБД кластерного типа. Реализация этого потенциала требует развития теории таких систем для выработки практических рекомендаций к их построению. Это возможно путем детального анализа процессов, в них протекающих. Только знание особенностей динамики параллельных СУБД с утяжеленными GPU-акселераторами узлами позволит дать объективные рекомендации по их построению. Проведение подобных исследований возможно с использованием специализированной системы моделирования рассматриваемых СУБД. Детальный анализ динамики на ее основе поможет развитию элементов теории и выработки практических рекомендаций. Архитектура системы, предложенная в данной работе, может быть принята за основу построения такой специализированной системы моделирования.

Литература

1. Xu Y., Kostamaa P., Zhou X., Chen L. Handling data skew in parallel joins in shared-nothing systems // ACM SIGMOD international Conference on Management 9of Data Canada, 2008, proceedings. ACM. 2008. P. 1043–1052.
2. Лепихов А.В. Параллельная обработка запросов в СУБД для кластерных вычислительных систем // Отчет в рамках гранта МК-3535.2009.9
3. Saecker M., Leischner N. GPU processing in database systems. Technical report // AMD Fusion Developer Summit, 2011.
4. Райхлин В.А. Моделирование машин баз данных распределенной архитектуры // Программирование. 1996. №2. С.7–16.
5. Райхлин В.А., Минязев Р.Ш. Мультикластеризация распределенных СУБД консервативного типа // Нелинейный мир. 2011. №8. С. 473–481.

ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ДЛЯ ПОИСКА ГРАНИЧНЫХ КОНТУРОВ КАРЬЕРОВ РУДНЫХ МЕСТОРОЖДЕНИЙ

В.М. Михелев, Д.В. Петров

Белгородский государственный национальный исследовательский университет

Рассматриваются основные принципы применения высокопроизводительных вычислительных грид-систем для расчета предельных границ рудных месторождений с использованием параллельного генетического алгоритма. Описывается алгоритм решения задачи и приводятся результаты вычислительных экспериментов.

Введение

Grid-система считается одной из разновидностей кластерных систем, но имеет свои особенности, обусловленные масштабом. Грид-системы в первую очередь характеризуются географической распределенностью вычисляемых ресурсов системы. Предоставляемая ими возможность объединять в рамках одной системы огромные вычислительные мощности делает интересным их использование для высокопроизводительных вычислений.

Вычислительные узлы Grid всегда расположены далеко друг от друга, слабо связаны между собой через интернет-каналы, и доступность того или иного из них в произвольный момент времени не гарантирована. Это накладывает дополнительные требования на управление ресурсами. Существует класс промежуточного программного обеспечения, который обеспечивает прозрачную работу приложений в неоднородной сетевой среде. Это так называемое межплатформенное связующее программное обеспечение (middleware).

Среди свободно распространяемых проектов связующего ПО можно выделить трех лидеров – Globus Toolkit, UNICORE и gLite. Всё это сервис-ориентированные системы, во многом реализующие общие стандарты и технологии работы с кластерами.

Globus Toolkit представляет собой набор модулей для построения виртуальной организации распределенных вычислений. Каждый модуль определяет интерфейс, используемый высокоуровневыми компонентами, и имеет реализацию для различных сред выполнения. Вкупе они образуют виртуальную машину Globus. Существуют следующие группы модулей:

- поиска и выделения ресурсов;
- коммуникаций;
- аутентификации;
- информационные;
- доступа к данным;
- создания процессов.

На основе этих низкоуровневых компонентов строятся сервисы более высокого уровня.

Цель данной статьи – продемонстрировать применение grid-системы для поиска предельных границ рудных месторождений.

Задача нахождения предельных границ карьеров рудных месторождений является одним из важнейших этапов планирования разработки полезных ископаемых открытым способом. Ее решение, во-первых, позволяет оценить объем получаемой прибыли, а во-вторых, является фундаментом для следующих этапов проектирования, таких как нахождение оптимальной сети карьерных транспортных путей и выбор мест расположения отвалов и перерабатывающих фабрик.

При нахождении границ карьера необходимо учитывать пространственное распределение компонентов полезных ископаемых и принятых устойчивых или технологически допустимых углов откосов бортов карьера. С вычислительной точки зрения данная задача является крайне сложной, т.к. для моделирования месторождений даже среднего размера приходится обрабатывать большие массивы данных.

Для решения задачи оптимизации с применением ЭВМ используют упрощенную блочную математическую модель месторождения полезных ископаемых. Каждый блок данной модели характеризуется числом (весом), показывающим чистую прибыль, получаемую в ходе его добычи, с учетом процентного содержания полезных элементов, себестоимости его выработки и рыночной стоимости полезных элементов. В этом случае задача оптимизации сводится к нахождению конечного набора соседних блоков, сумма весов которых будет максимальна. Предлагается применить для этого генетический алгоритм.

Генетические алгоритмы оптимизации основаны на моделировании процессов мутации, скрещивания и естественного отбора. Поиск в них начинается со случайной популяции индивидуумов, однозначно характеризующихся неким набором данных – хромосомой. Качество каждого индивидуума оценивают посредством расчета функции пригодности, аргументом которой является его хромосома. Путем применения генетических операторов (отбора, мутации, скрещивания) данная популяция улучшается до тех пор, пока либо функция пригодности одного из индивидуумов не достигнет приемлемого значения, либо не пройдет допустимое количество циклов воспроизводства.

Формат представления хромосом

Для реализации генетического алгоритма в первую очередь необходимо разработать формат представления хромосом. В контексте задачи нахождения границ карьера можно предложить следующее решение: форма любого допустимого (с учетом углов наклона) карьера представляется с помощью массива целых чисел. Каждый элемент такого массива показывает глубину карьера в текущем столбце трехмерной модели месторождения.

Пусть имеется трехмерная блочная модель месторождения P_{IxJxK}, каждый элемент которой характеризуется числом (весом)

$$p_{ijk},\ i\in[0, I],\ j\in[0, J],\ k\in[0, K], \qquad (1)$$

показывающим чистую прибыль, получаемую в ходе его добычи, с учетом процентного содержания полезных элементов, себестоимости его выработки и рыночной стоимости полезных компонентов. Тогда ее можно охарактеризовать вектором $X = =\{x_1,\ldots,x_n\}$, где $n = I * J$, в котором значение глубины столбца с координатами (i, j) помещается в позицию

$$x_q,\ q = i * I + j. \qquad (2)$$

Этот массив является хромосомой, т.к. он полностью характеризует один индивид – одну конкретную форму карьера. Путем итеративного применения генетических операторов к набору таких индивидов (популяции) находится оптимальная форма поверхности карьера.

Для расширения области поиска, уменьшения вероятности преждевременного схождения и уменьшения времени вычислений был разработан параллельный вариант

данного алгоритма – иерархический параллельный генетический алгоритм с двумя уровнями параллелизма.

Первый уровень параллелизма организуется за счет применения островной модели многопопуляционного параллельного генетического алгоритма. Здесь ускорение достигается за счет выделения нескольких начальных популяций, развивающихся независимо, и периодически обменивающихся наиболее хорошим генетическим материалом. Данный обмен осуществляется посредством механизма миграции особей между популяциями. Такой подход обеспечивает снижение вероятности преждевременного вырождения популяций, увеличению их разнообразия и ускорению схождения алгоритма поиска.

Второй уровень иерархии организуется за счет применения для каждой подпопуляции однопопуляционной модели параллельного генетического алгоритма типа «хозяин–подчиненный». Она заключается в том, что в рамках одной популяции функция приспособленности каждого индивидуума вычисляется в отельном потоке, что в итоге приводит к ускорению работы алгоритма. При этом один поток является главным, «хранителем» популяции и отвечает за работу генетических операторов, а ряд потоков-подчиненных только вычисляют функцию приспособленности.

Кроме того, внимательно проанализировав структуру последовательной версии алгоритма, можно выделить еще несколько вычислительно независимых участков, которые можно выполнять параллельно в рамках развития популяции. Так, после разбиения хромосом по парам каждую пару можно скрещивать параллельно. Также и процедуру мутации можно выполнять для каждой хромосомы независимо.

Вычислительные эксперименты

В качестве технической платформы для проведения вычислительных экспериментов использовался сегмент грид-системы под управлением платформы Globus Toolkit. В состав данного сегмента входит координирующий сервер и вычислительные узлы на базе процессоров Intel Xeon. Структурная схема взаимодействия всех этих компонентов представлена на рис. 1.

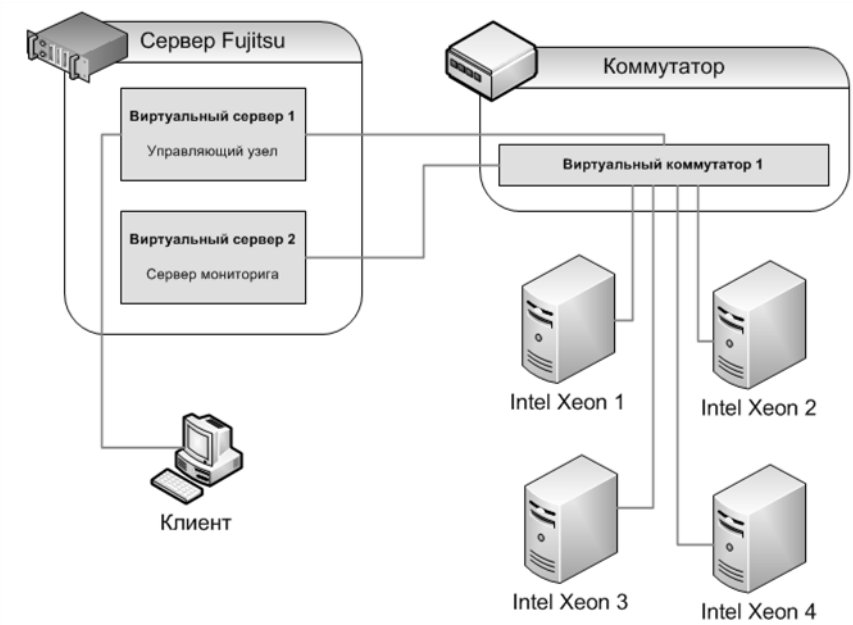


Рис. 1. Схема оборудования

Суммарные технические характеристики кластерной системы приведены в табл. 1.

Таблица 1. Суммарные технические характеристики кластерной системы

Характеристика	**Значение**
Семейство процессора	QuadCore Intel Xeon
Частота процессора	1.6 ГГц
Количество процессоров	12
Количество ядер	32
Объем ОЗУ	20 Гб
Объем HDD	1.2 Тб
Сеть	Gigabit Ethernet (1000 Mbps)

Алгоритм тестировался на нескольких моделях пространственного распределения полезных компонентов в земной поверхности: наклонное послойное залегание, вертикальное залегание, равномерное случайное распределение. На рис. 2 приведен пример визуального представления граничной формы карьера размером 100 × 100 × 100 метров с разрешением 1 метр, рассчитанного генетическим алгоритмом.

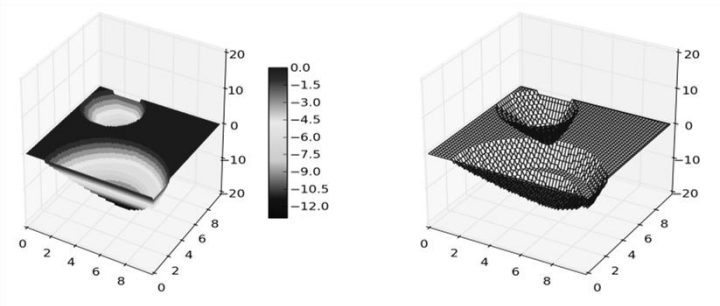


Рис. 2. Трехмерное изображение формы карьера, масштаб 1:10000

В рамках данного эксперимента проводилась проверка работы алгоритма на нескольких узлах грид-системы, при этом было использовано два уровня параллелизма. Цель данного эксперимента – выяснить, как меняется время расчетов в зависимости от количества вычислительных узлов.

В качестве тестовых данных использовалась модель карьера со случайным пространственным распределением полезных компонентов размером 100 × 100 × 100 блоков. После проведения вычислительных экспериментов, был получен результат, представленный на рис. 3.

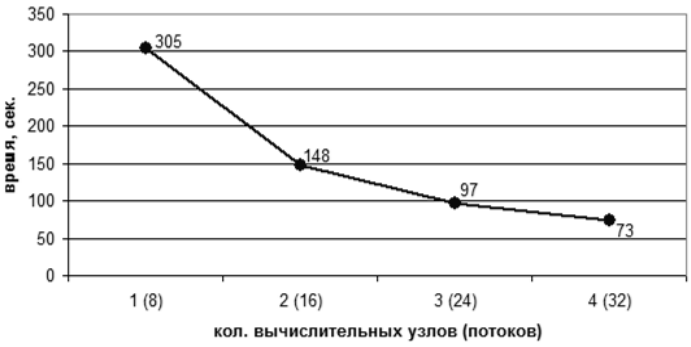


Рис. 3. Зависимость времени выполнения от количества задействованных узлов грид-системы

По результатам эксперимента можно сделать вывод, что алгоритм хорошо масштабируется на распределенную вычислительную систему. И, хотя за счет задержек на об-

мен данными между вычислительными узлами наблюдается некоторое замедление вычислительного процесса, все же целесообразно применять подобные системы для снижения времени вычислений при обработке больших моделей месторождений полезных ископаемых.

Выводы

Результаты вычислительных экспериментов показали высокую перспективность предложенного метода для выполнения расчетов на регулярных блочных моделях месторождений твердых полезных ископаемых, разрабатываемых открытым способом. Основные преимущества предложенного метода заключаются в предоставлении нового принципа решения задачи оптимизации карьеров, позволяющего работать напрямую с трехмерной моделью месторождения, что значительно повышает адекватность получаемой модели. Кроме того, возможности гибкого масштабирования вычислительного процесса позволяют сокращать время обсчета модели почти линейно с увеличением количества вычислительных узлов.

Литература

1. Васильев П.В. Ускорение моделирования и оптимизации извлечения запасов рудных месторождений на основе параллельных вычислений // Горный информационно-аналитический бюллетень. – М.: МГГУ , 2012. – №3. – С. 205-211.
2. Гергель В.П. Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н. Новгород: Изд-во Нижегородского государственного университета, 2003. – 325 с.
3. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем: учебник для студентов вузов. – М.: МГУ, 2010. – 544 с.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. Горячая Линия Телеком, 2007.
5. Шпаковский Г.И. Реализация параллельных вычислений: MPI, OpenMP, кластеры, грид, многоядерные процессоры, графические процессоры, квантовые компьютеры. – Минск: БГУ, 2011.

ЧИСЛЕННОЕ РЕШЕНИЕ И ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ МЕЖДИСЦИПЛЕНАРНЫХ ЗАДАЧ

В.Я. Модорский

Пермский национальный исследовательский политехнический университет

В процессе работы сформулированы физическая и математическая модели газогидродинамического процесса. Произведены численные эксперименты по моделированию потока в канале, проведены вычислительные эксперименты по оценке напряженно-деформированного состояния силовой конструкции. Разработана экспериментальная установка по анализу газоупругих процессов в энергетических установках. Выявлены закономерности усиления колебаний в модельных каналах при варьировании параметров потока и конструкции. В результате работы накоплен и структурирован банк данных численных экспериментов по исследованию газогидродинамических процессов в каналах наукоемких изделий, сформулированы рекомендации по уменьшению интенсивности непрогнозируемых эффектов.

Введение

Воздействие на динамические системы может приводить к тому, что их выходные характеристики отклоняются от расчетных. При определенных условиях может оказаться, что параметры движения системы начинают колебаться с увеличивающейся со временем амплитудой, что недопустимо [1].

Неустойчивость существенно снижает надежность конструкции, ухудшает ее рабочие характеристики и может привести к ее разрушению. Поэтому выявлению причин неустойчивости рабочих процессов, ликвидации колебаний или снижению их амплитуды до допустимого уровня уделяется большое внимание.

Неустойчивые рабочие процессы реализуются при наличии возмущений, образующих волны давления [1]. Источником неустойчивости может являться резонансное взаимодействие системы «поток газа – конструкция» [2]. При резонансе, когда одна из собственных частот колебаний конструкции близка или кратна частоте газодинамических колебаний в этой же конструкции, могут проявляться неустойчивые эффекты типа «флаттер». Частота и форма наблюдающихся при этом волн зависят от геометрических и физико-механических характеристик конструкции и от параметров нагрузки.

1. Исследования взаимодействия в динамической системе «газ–конструкция»

Моделирование течения газа для принятой модели проточной части экспериментальной установки выполнялось в программном комплексе *FlowVision.*

Программный комплекс *FlowVision* предназначен для моделирования трехмерных течений жидкости и газа в технических и природных объектах, а также визуализации этих течений методами компьютерной графики.

Моделируемые течения включают в себя стационарные и нестационарные, сжимаемые, слабосжимаемые и несжимаемые потоки жидкости и газа. Использование различных моделей турбулентности и адаптивной расчетной сетки позволяет моделировать сложные движения жидкости, включая течения с сильной закруткой, горением, течения со свободной поверхностью.

FlowVision основан на конечно-объемном методе решения уравнений газодинамики и использует прямоугольную адаптивную сетку с локальным измельчением. Для ап-

проксимации криволинейной геометрии с повышенной точностью *FlowVision* использует технологию подсеточного разрешения геометрии. Эта технология позволяет импортировать геометрию из систем САПР и обмениваться информацией с системами конечно-элементного анализа. Использование этой технологии позволило решить проблему автоматической генерации сетки – чтобы сгенерировать сетку, достаточно задать всего лишь несколько параметров, после чего сетка автоматически генерируется для расчетной области, имеющей геометрию любой степени сложности.

Процесс расчета течения рабочего тела включает в себя следующие шаги, выполняемые пользователем:

- создание области расчета («геометрии» устройства) в САПР и импортирование ее через форматы *VRML, STL, DEFORM, ABAQUS, ANSYS* или *NASTRAN* в *FlowVision* (*IGES, PARASOLID, VDAFS* с помощью *Flow3DVision*);
- задание физических параметров;
- задание математической модели;
- задание граничных условий;
- задание исходной расчетной сетки и критериев её адаптации по решению и по граничным условиям;
- задание параметров методов расчета;
- проведение расчета;
- просмотр результатов расчета в графической форме («визуализация» результатов расчетов) и сохранение данных в файлы;
- оценка точности расчетов методом сходимости по сетке [4].

В работе рассмотрено два возможных варианта возбуждений колебаний воздуха в экспериментальной установке: при резком открытии штуцера для отвода рабочего тела и при работе динамика с частотой 330 Гц.

Расчет собственных частот модельной камеры экспериментальной установки выполнялся в инженерном пакете *ANSYS Workbench*.

ANSYS – многоцелевой конечно-элементный пакет для проведения анализа в широком спектре инженерных дисциплин. Он занимает одно из ведущих мест среди универсальных *CAE*-систем и предметно ориентированных (или методо-ориентированных) САПР на основе метода конечных или граничных элементов (*Cosmos, Pro-Engineering, ABAQUS, NASTRAN* и другие).

Workbench – это новое поколение программных продуктов, использующих возможности решателей *ANSYS*. Эта среда инженерного анализа предоставляет уникальные возможности по интеграции с CAD системами (в том числе двунаправленную ассоциативную связь). Можно сочетать процесс проектирования в CAD пакете с получением достоверных данных расчетов и проведением оптимизации конструкции. *Workbench Products* состоят из модулей: *Design Simulation* (использует лицензии *DesignSpace, Professional, Structural, Mechanical*), *Design Modeler, Design Xplorer, FE Modeler* [5].

Последовательность решения задач в *ANSYS Workbench* следующая:

- загрузить геометрическую модель;
- задать параметры сетки и создать сеточную модель;
- задать свойства материалов;
- закрепить конструкцию;
- выбрать опции расчета;
- настроить решатель;
- провести расчет;
- провести анализ результатов.

2. Разработка конструктивной схемы экспериментальной установки

В рамках этого этапа сформулированы требования к рабочим параметрам и конструкции и в соответствии с ними разработана экспериментальная установка.

При проектировании экспериментальной установки вопрос обеспечения необходимой резонансной частоты колебаний в системе «поток газа – конструкция» является очень важным.

Известно, что резонанс наблюдается при близости частот продольных колебаний в газе и в конструкции.

$$f_{\text{г}} = f_{\text{к}}. \qquad (1)$$

Из этого следует, что геометрические и физико-механические характеристики модельной камеры должны быть подобраны так, чтобы выполнялось условие (1).

Подберем геометрические характеристики модельной камеры.

В качестве математической модели продольных колебаний корпуса модельной камеры рассмотрим упругий консольный стержень длиной l, на свободном конце которого находится сосредоточенная масса. В данной задаче инерционностью стержня можно пренебречь и учитывать только его упругость.

Тогда частоты продольных колебаний, возникающие в корпусе камеры, можно вычислить по формуле [3]:

$$f_{к} = \frac{1}{2\pi}\sqrt{\frac{c}{M}}, \qquad (2)$$

где c – жесткость конструкции, M – масса груза.

Частота продольных колебаний газа в замкнутой трубе:

$$f_{г} = \frac{a}{2l}, \qquad (3)$$

где $a = \sqrt{k\frac{P}{\rho}}$ – скорость распространения возмущений в газе, k – показатель адиабаты, P – давление, ρ – плотность.

Параметры установки (рис. 1) выбраны так, что частота продольных колебаний в газе $f_{\text{г}} = 330$ Гц.

Из условия резонансного взаимодействия (1) следует $f_{к} = 330$ Гц.

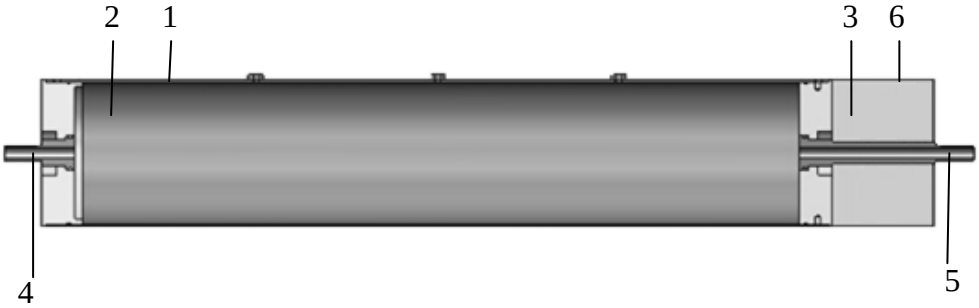


Рис. 1. Конструктивная схема модельной камеры: 1– корпус; 2, 3 – крышки; 4, 5 – штуцеры; 6 – модельные элементы

В ряде случаев для системы в целом значения $f_{\text{г}}$ и $f_{\text{к}}$ могут отличаться от парциальных [4]. Необходимые значения частот мы можем получать при варьировании (подборе) характеристик конструкции и параметров нагрузки. Таким образом, потребуется дополнительная настройка модельной камеры на резонансный режим.

Сформулированы следующие требования, предъявляемые к экспериментальной установке:

- основание должно обладать необходимым запасом прочности и жесткости, обеспечивать необходимую ориентацию и надежное крепление модельной камеры экспериментальной установки;
- конструкция модельной камеры должна обеспечивать настройку резонансного взаимодействия в системе «поток газа – конструкция»; возможность оснащения корпуса модельной камеры необходимыми средствами измерения, с помощью которых можно получить информацию о параметрах потока рабочего тела и уровне деформаций конструкции в контрольных точках; возможность установки сменных корпусов, отличающихся материалом и геометрическими характеристиками; модельная камера не должна является сосудом, работающим под давлением;
- система регистрации, измерения и анализа должна обеспечивать регистрацию на ПК временных зависимостей давления, виброускорений, виброскоростей и виброперемещений в нескольких контрольных точках конструкции; точность измерений; проведение комплексного анализа сигналов, включая спектральный анализ; обработку файлов измерения в *MS Excel*;
- система возбуждения колебаний должна обеспечивать безопасную работу на экспериментальной установке.

Исходя из этого, можно предложить схему экспериментальной установки, которая представлена на рис. 2.

Модельная камера вертикально устанавливается на основание. Крепление ограничивает возможность смещения модельной камеры под действием волн давления.

Для моделирования потока газа предусмотрена система подачи и отвода рабочего тела (газа). На первом этапе в качестве рабочего тела взят воздух.

Рис. 2. Конструктивная схема экспериментальной установки

Такая конструкция модельной камеры позволяет реализовать несколько вариантов возбуждений колебаний воздуха в экспериментальной установке.

В результате проведения вычислительных экспериментов получены временные зависимости изменения параметров и кинограммы процессов. В ходе проведения физических экспериментов получены показания вибродатчиков и микрофонов. Проведен спектральный анализ этих сигналов и подтверждены результаты вычислительных экспериментов.

Литература

1. Присняков В.Ф. Динамика ракетных двигателей твердого топлива. – М.: Машиностроение, 1984. – 248 с.
2. Модорский В.Я. Нелинейное деформирование стержневой конструкции при наддуве // Авиационная техника. 2003. №3. С. 63–64.
3. Вибрации в технике: Справочник. В 6 т. / Ред. В.Н. Челомей. – Т.1 Колебаний линейных систем / Под ред. В.В. Болотина. – М.: Машиностроение, 1978. 352 с.
4. Модорский В.Я., Соколкин Ю.В. Газоупругие процессы в энергетических установках / Под ред. Ю.В. Соколкина. – М.: Физматлит, 2007. – 176 с.
5. Котов А.Г. САПР изделий из композиционных материалов. Моделирование процессов деформирования и разрушения в среде ANSYS. Пермь: Изд-во Перм. гос. техн. ун-та, 2008. – 351 с.

РАЗРАБОТКА РАСЧЕТНО-ЭКСПЕРИМЕНТАЛЬНОГО КОМПЛЕКСА ПО ИССЛЕДОВАНИЮ АЭРОУПРУГИХ КОЛЕБАНИЙ В ЭНЕРГЕТИЧЕСКИХ УСТАНОВКАХ

В.Я. Модорский, Л.Н. Бутымова

Пермский национальный исследовательский политехнический университет

Сформулированы физическая и математическая модели газогидродинамического процесса. Произведены численные эксперименты по моделированию потока в канале. Проведены вычислительные эксперименты по оценке напряженно-деформированного состояния силовой конструкции. Разработана экспериментальная установка по анализу газоупругих процессов в энергетических установках.

Совершенствование рабочих процессов в энергетических установках сопряжено с решением ряда проблем. В частности, необходима разработка расчетных моделей и методик, позволяющих прогнозировать опасные аэроупругие колебания в конструкции. В соответствии со схемой Коллара изучение этой проблемы предлагается проводить по двум направлениям: исследование задач, связанных с волновыми процессами в газе, и исследование процессов динамического напряженно-деформированного состояния в конструкции. В связи со сложностью задачи необходимо сочетание аналитического, численного и экспериментального подходов.

Важную роль при изучении колебательных режимов, в связи со значительными техническими трудностями и материальными затратами при проведении натурных и модельных испытаний, играет теоретическое направление исследований. В данном случае аналитическое исследование реализует «таровский» подход. Разработана простейшая аналитическая модель. Схема, реализующая автоколебания, состоит из источника колебаний, колебательного звена и отрицательной обратной связи.

Для численных расчетов применялся *ANSYS* – конечно-элементный программный комплекс для проведения анализа в широком спектре инженерных дисциплин.

Для численного моделирования работы экспериментальной установки сформулирована следующая физическая модель: процессы рассматриваются в трехмерной динамической постановке, в качестве рабочего тела выбран воздух, стенки модельной камеры непроницаемые, нетеплопроводные, в начальный момент времени модельная камера заполнена воздухом, значение давления для первого варианта равно 0,3 МПа, для второго варианта – 0,1 МПа, температура воздуха 293 К [1].

Численно определены параметры устройства для возбуждения колебаний в экспериментальной модельной установке – акустического трансформатора. Получены максимальные коэффициенты усиления (K_y) для заданных физико-механических характеристик и геометрии волновода [2].

Экспериментальное исследование аэроупругости предлагается проводить на модельной установке. Предлагается рассмотреть возможность использования акустического трансформатора для возбуждения колебаний.

Данная установка включает в себя модельную камеру и систему измерения и регистрации.

Литература

1. Горбушина Н.В., Бутымова Л.Н., Модорский В.Я., Соколкин Ю.В. Решение междисциплинарных задач на ВВК ПНИПУ // Материалы XXXXI Всероссийского симпозиума «Механика и процессы управления». Т. 2. – М.: РАН, 2011. С. 24–29.
2. Бульбович Р.В., Бутымова Л.Н., Модорский В.Я., Писарев П.В., Зимин Д.В. Численное моделирование напряженно-деформированного состояния активной заглушки резонатора // Научно-технический вестник Поволжья. 2012. № 1. С. 95-101.

ЗАДАЧА ОПТИМИЗАЦИИ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРИЛОЖЕНИЯ В DESKTOP GRID

Н.Н. Никитина, А.С. Румянцев

Институт прикладных математических исследований
Карельского научного центра РАН, Петрозаводск

В настоящее время для ускорения вычислений широко применяются системы типа Desktop Grid. Данные системы распределенных вычислений представляют собой совокупность неспециализированных вычислительных единиц – в частности, персональных компьютеров, принадлежащих организации или частным лицам. При этом сравнительно небольшая мощность отдельных вычислительных узлов компенсируется их большим количеством и наличием высокоскоростных каналов связи между ними, таких как локальные сети организации и Интернет. Эффективность функционирования таких систем зависит от методов распределения заданий по вычислителям, который, определяет алгоритм работы менеджера очереди. Отметим работы [1–4], в которых рассматриваются различные подходы к построению эффективных расписаний для Desktop Grid систем.

Системы Desktop Grid предназначены для выполнения приложений, которые требуют большой вычислительной мощности (хранения больших объемов данных) и при этом могут быть разделены на большое количество независимых задач. Одной из ключевых особенностей систем Desktop Grid является непостоянство множества вычислительных узлов. Вычислительные узлы доступны лишь в то время, которое в одностороннем порядке устанавливают их владельцы (производя так называемые добровольные вычисления). При составлении расписания задач для Desktop Grid необходимо учитывать отказы узлов, связанные с их временной или постоянной недоступностью. Другой важной особенностью, связанной с архитектурой системы, является возможность получения в результате вычислений недостоверного либо искаженного значения, а также отсутствие результата в заданный промежуток времени (называемый deadline). Указанные особенности приводят к необходимости дублирования задач с одинаковыми исходными данными для повышения надежности вычислений, что, однако, полностью не исключает появления ошибок.

В данной работе приводится постановка задачи оптимизации расписания для системы Desktop Grid с учетом указанных особенностей. Предположим, что в состав системы Desktop Grid входит множество вычислительных узлов G мощностью $|G|=K$. Положим для простоты, что процесс вычислений можно представить в виде условных тактов времени (соответствующих значению deadline). Каждому узлу $g \in G$ поставлено в соответствие значение вероятности отказа за один вычислительный такт $P(g)$, $0 \le P(g) \le 1$.

Рассмотрим набор значений $\{p_i\}_{i=1}^{\infty}$, называемых порогами отказов узлов: $p_0 = 0 < p_1 < p_2 < \ldots < p_i < \ldots$ В соответствии с данным набором значений узлы могут быть разбиты на непересекающиеся группы $G_i \subseteq G, \bigcup_{i=1}^{\infty} G_i = G$:

$$G_i = \{g \in G : P(g) \in [p_{i-1}, p_i)\}, \quad 1 \le i \le \infty.$$

Обозначим мощности групп узлов, полученных данным разбиением, через $|G_i| = K_i$, при этом $\sum_{i=1}^{\infty} K_i = K$. Вероятность отказа на любом из узлов группы G_i, по построению, не превосходит p_i.

Предположим, что в системе нужно выполнить приложение, состоящее из n идентичных независимых задач, каждая из которых выполняется в течение одного вычислительного такта. В целях ускорения процесса вычислений разобьем множество задач на группы $\{n_i\}_{i=1}^{\infty}, n_i \geq 0, \sum_{i=1}^{\infty} n_i = n$, и назначим каждую группу из n_i задач на выполнение соответствующей группе узлов G_i. При этом задача, назначенная группе узлов G_i, дублируется на i узлов этой группы для повышения надежности. Таким образом, в группе узлов G_i может одновременно выполняться не более $\left\lfloor \frac{K_i}{i} \right\rfloor$ уникальных задач. С учетом ненадежности узлов, задача может не выполниться за один вычислительный такт. В таком случае будет сгенерирован новый экземпляр данной задачи, который будет отправлен на выполнение той же группе узлов в том же количестве экземпляров. Вероятность того, что ни один экземпляр некоторой задачи в группе G_i не выполнится успешно за один вычислительный такт, не превосходит p_i^i, за j вычислительных тактов – p_i^{ji}, что соответствует геометрическому распределению с вероятностью успеха p_i^i. Тогда математическое ожидание общего количества задач (включая исходную), порожденных одной задачей, назначенной группе узлов G_i, составляет $\frac{1}{1-p_i^i}$.

При полной надежности вычислительных узлов общая продолжительность вычислений в группе G_i составила бы

$$\left\lceil \frac{n_i}{\left\lfloor \frac{K_i}{i} \right\rfloor} \right\rceil$$

тактов. С учетом задач, порожденных в результате отказов, ожидаемая общая продолжительность вычислений в группе G_i составит

$$t_i = \frac{\left\lceil \frac{n_i}{\left\lfloor \frac{K_i}{i} \right\rfloor} \right\rceil}{1-p_i^i}.$$

Задача оптимизации расписания может быть сформулирована следующим образом: $\max_i t_i \longrightarrow \min_{\{p_i\},\{n_i\}}$.

Критерием оптимальности является минимизация времени выполнения приложения, то есть наибольшей продолжительности вычислений в группах узлов. Предполагается решать задачу с использованием методов теории игр.

Работа выполнена при поддержке РФФИ, проекты 10-07-00017, 12-07-31147.

Литература

1. Benoit A. et al. Scheduling concurrent bag-of-tasks applications on heterogeneous platforms // IEEE Transactions on Computers, 2010. Vol. 59, Iss. 2. P. 202–217.
2. Celaya J., Marchal L. A fair decentralized scheduler for bag-of-tasks applications on desktop grids // Cluster Computing and the Grid, IEEE International Symposium on, 2010. P. 538–541.
3. Kondo D., Casanova H. computing the optimal makespan for jobs with identical and independent tasks scheduled on volatile hosts // Technical Report CS2004-0796, Dept. of Computer Science and Engineering, University of California at San Deigo, July, 2004.
4. Kondo D., Casanova H., Chien A. Scheduling task parallel applications for rapid application turnaround on enterprise desktop grids // Journal of Grid Computing. 2007. 5(4). P. 379–405.

ОБЛАКО ДАННЫХ ДЛЯ АКТИВНОГО ХРАНИЛИЩА МНОГОМЕРНЫХ МАССИВОВ

А.М. Новиков

НИЦ «Курчатовский институт», НБИКС-Центр, Москва

Представлены результаты тестирования производительности созданной базы многомерных массивов для хранения геофизических данных общепринятых форматов с использованием облачной системы масштабируемого, отказоустойчивого хранилища OpenStack Swift. Решение названо «Активное Хранилище», поскольку оно реализует функции хранения совместно с базовой или комплексной обработкой извлекаемых по запросу данных. Сделаны выводы об оптимальности созданного решения для использовании в облачных инфраструктурах.

Развитие научных исследований в областях нанофизики, физики высоких энергий, биологии, наук о Земле требует разработки приложений, основывающихся на работе с большими объемами данных – результатов экспериментов и наблюдений и их обработки.

Высокопроизводительные вычисления для обработки постоянно растущих объемов научных данных, а также средства высокоскоростного, а иногда и реального времени доступа и средства визуализации данных требуют от научного сообщества создания, поддержания и обновления аппаратно-программных комплексов, при том что внедрение этих структур напрямую не касается научной работы и требует вложений экономических и людских ресурсов.

Использование современного оборудования и технологий виртуализации привело ко всё более расширяющемуся использованию так называемых облачных вычислений (cloud-computing). В общем виде облако включает в себя ресурсы вычислительные, хранения и доступа (т.е. разграничения по пользователям и проектам). Приложения для обработки научных данных зачастую узкоспециализированы, разработаны давно для устаревших фреймворков или языков программирования либо же доступны в виде пакета для одной из свободно распространяемых ОС (класса Linux) или архива открытых кодов. Всё это ведёт к необходимости использования для разворачивания таких приложений самого широкого класса «Инфраструктура как сервис» (IaaS), т.е. с возможностью управления полным процессом от установки ОС до конфигурации зависимостей и приложения, которое затем можно выставлять и использовать в качестве веб-сервиса (в том числе и для работы через удалённый рабочий стол). Тем не менее, пока реализации подобных пакетов или комплексов приложений в одном «образе» виртуальной машины только начинают своё развитие, прежняя проблема множественности и иногда несовместимости форматов данных для разных утилит остаётся актуальной.

В науках об окружающей среде широко используются многомерные массивы данных для хранения результатов моделирования, спутниковых наблюдений, растровых карт и т.д. Эти наборы данных могут быть довольно велики, вплоть до нескольких терабайт и более. Многомерность следует из наличия координат пространственной сетки, времени наблюдений и некоторых дополнительных параметров (например, высоты). Большая часть данных об окружающей среде хранится и тиражируется в виде коллекций файлов общепринятых открытых форматов, таких как netCDF, GRIB, HDF5 или же проприетарных. При работе с таким представлением данных могут возникать затруд-

нения из-за необходимости наличия базовых навыков в обращении с теми или иными форматами данных или утилитами для работы с ними (а иногда для разных форматов и утилит одновременно). Кроме того, зачастую важны «сквозные» или общие исследования, для которых файлы нужно объединить по времени или пространству (исключая приведение к общей основе), хотя может оказаться и так, что часть данных находится в совершенно другом формате и у другого научного сообщества. И наконец, т.к., например, геофизические или климатические данные имеют геопривязку и отсчёты во времени, то сравнительный анализ удобно проводить в наглядных представлениях, посредством графиков, временных рядов или пространственных полей (с изолиниями или градиентных). При этом окно пространственной выборки может быть большим, что удобно для использования климатических и спутниковых данных в системах мониторинга.

Таким образом, возникает необходимость в создании инфраструктур или сред, в частности веб-порталов, где большинство действий будет осуществляться без непосредственного участия пользователя, предоставляя последнему гибкие средства представления и анализа данных, желательно в режиме быстрого отклика (хотя асинхронные запросы или длительные задачи анализа также бывают актуальными). Данные для таких сред должны быть представлены единообразно и в полном или регулируемом объёме доступа. В аппаратно-программной подложке, обеспечивающей внешнюю инфраструктуру подобных сред, можно выделить несколько составляющих. Это файловые системы или системы баз данных, системы обработки данных и, для некоторых задач, среды для рендеринга или отображения данных. Каждая из составляющих может быть распределённой, параллельной или масштабируемой, работать в облаке, грид-среде или выделенном вычислительным кластере.

В работе представляется тестирование производительности созданной базы многомерных массивов для хранения климатических данных с использованием облачной системы масштабируемого, отказоустойчивого хранилища OpenStack Swift, развёрнутых на вычислительном кластере. Решение названо «Активное Хранилище» (АХ), поскольку оно реализует функции хранения совместно с базовой или комплексной обработкой извлекаемых по запросу данных.

Структура используемого решения включает в себя OpenDAP веб-сервер с подключенной библиотекой серверной части АХ, реляционную SQL СУБД для хранения метаданных и бинарные данные в виде файлов-объектов системы Swift. В таком виде преобразованиями данных или проведением операций с ними занимается приложение веб-сервера, а библиотека АХ реализует работу с метаданными и объектам данных. Оригинальная разработка включает в себя встроенные процедуры на языке .Net для MS SQL Server, кроссплатформенная – на языке Java для PostgreSQL соответственно. В качестве основы для схемы хранения была использована Общая модель данных (CDM), созданная на основе таких форматов научных данных, как NetCDF, HDF5, OpenDAP. Данные организуются в иерархическую структуру с метаданными и хранятся в виде специализированных бинарных объектов, содержащих определённую форму по измерениям хранимой величины, таким образом исходный многомерный массив разделяется на отрезки или чанки. Меняя размеры «чанков», можно варьировать скорость извлечения данных в разных видах запросов, например с преобладанием временной или пространственных координат (по всей Земле или по всем годам измерений). Для ускорения работы предусмотрена масштабируемость структуры по кластеру баз данных.

В качестве системы хранения использовалась облачная система OpenStack Swift с использованием одной или нескольких точек входа, различной степени репликации данных, протоколов HTTP или HTTPS. «Активное Хранилище» в виде веб-сервиса размещалось на виртуальных машинах с балансировщиком нагрузки, а также в виде отдельной библиотеки совмещённых клиента и сервера для приложения тестировщика.

Для тестирования использованы реальные данные реанализа погоды NCEP в виде значений геофизических величин на узлах пространственно-временной сетки. Общий объём данных составлял ~2 Гб.

Показано, что использование «чанков» оптимизирует запросы средних геометрий выборок из всего многомерного пространства величины. Протестированы различные объёмы геометрий с изменением выборок от максимальных по одному измерению (отсчётов времени) и минимальных по другим (точкам пространственной сетки) до минимальных по первому и максимальных по другим. Приводятся сравнительные результаты запросов чтения различных объёмов и чтения множественными одновременными клиентами. Показаны гибкая масштабируемость совмещённой системы «Активное Хранилище» и Openstack Swift и устойчивость к высоким нагрузкам. Сделаны выводы об оптимальности использования именно такой технологии хранения многомерных данных для работы в облачных инфраструктурах.

АЛГОРИТМ ПОИСКА ПАРАЛЛЕЛЬНОСТИ С СИНХРОНИЗАЦИЯМИ ВО ВЛОЖЕННОСТЯХ ЦИКЛОВ

А.А. Новокрещенов

Нижегородский государственный технический университет им. Р.Е. Алексеева

Предлагается алгоритм поиска параллельности с синхронизациями во вложенностях циклов. Алгоритм может быть использован для получения параллельных программ, предназначенных для выполнения как на системах с общей памятью, так и на гетерогенных системах, в состав которых входит графический процессор. По результатам работы алгоритма параллельный код может быть сгенерирован существующими генераторами кода.

Введение

Актуальной, на сегодняшний день, является задача распараллеливающей трансляции. Это комплексная задача, состоящая из фаз синтаксического анализа, анализа зависимостей, поиска параллельности, генерации кода. Ядром процедуры распараллеливающей трансляции является фаза поиска параллельности, т.к. именно с ее помощью происходит преобразование последовательной программы в ее параллельную форму. В данной работе предлагается алгоритм поиска параллельности с синхронизациями во вложенностях циклов. На ряде тестовых примеров было подтверждено, что параллельные программы, полученные с помощью данного алгоритма, могут быть эффективно выполнены, как на системах с общей памятью, так и на гетерогенных системах, в состав которых входят графические процессоры.

1. Основные термины и обозначения

Пусть дана неидеальная вложенность циклов глубины d, с индексными переменными $x_1, x_2, ..., x_d$ и инструкциями присваивания $S_1, S_2, ..., S_n$. При этом границы изменения каждой индексной переменной x_i представляют собой аффинные выражения от индексных переменных $x_1, ..., x_{i-1}$ и/или символьных констант, а шаг изменения каждой индексной переменной равен +1. Условия в операторах ветвления, присутствующих в теле вложенности, и индексные выражения обращения к массивам в инструкциях $S_1, S_2, ..., S_n$, так же представляют собой аффинные выражения от индексных переменных охватывающих циклов и/или символьных констант [1]. Существенная часть вычислений, выполняемых в научных и инженерных приложениях, протекает во вложенностях циклов, удовлетворяющих данным условиям [2].

Срабатывание инструкции S_i при определенной комбинации индексных переменных называют экземпляром инструкции S_i [3]. Все множество комбинаций значений индексных переменных, при которых происходит срабатывание инструкции S_i, называют ее доменом – D_{S_i}. Если инструкцию S_i охватывают m циклов, то D_{S_i} может быть представлен в виде целочисленного параметризованного многогранника в Z^m от

$\vec{x}_{S_i} = \{x_1, x_2, ..., x_m\}$ и $\vec{n}_{S_i} = \{n_1, n_2, ..., n_k\}$. Вектор $\vec{n}_{S_i}$ называют вектором глобальных параметров циклов, охватывающих S_i [2].

Обращение к массиву, заключенное в инструкции S_i, может быть представлено в виде кортежа:

$$\left\langle A, f(\vec{x}_{S_i}), \delta \right\rangle, \tag{1}$$

где A – имя массива, к которому выполняется обращение, $f(\vec{x}_{S_i})$ – аффинная функция, устанавливающая соответствие между комбинацией значений индексных переменных $\vec{x}_{S_i}$ и элементом массива A, δ - переменная, принимающая значения истина или ложь, если происходит обращение на запись или чтение соответственно [4].

В качестве модели инструкции S_i может быть использован следующий кортеж:

$$\left\langle D_{S_i}, M_{S_i} \right\rangle, \tag{2}$$

где D_{S_i} – домен инструкции S_i, M_{S_i} – множество обращений, заключенных в теле S_i представленных в форме (1).

Модель входной вложенности циклов представляет собой множество кортежей вида (2), определяющее все инструкции данной вложенности.

Инструкция $\left\langle D_{S_j}, M_{S_j} \right\rangle$ зависит по данным от инструкции $\left\langle D_{S_i}, M_{S_i} \right\rangle$ в том случае, если существует пара обращений $\left\langle A, f_1(\vec{x}_{S_i}), \delta_1 \right\rangle \in M_{S_i}$ и $\left\langle A, f_2(\vec{x}_{S_j}), \delta_2 \right\rangle \in M_{S_j}$, для которой справедливо:

$$(\delta_1 \vee \delta_2) \wedge (\exists \vec{x}^*_{S_i} \in D_{S_i}, \exists \vec{x}^*_{S_j} \in D_{S_j} \mid \vec{x}^*_{S_i} \prec \vec{x}^*_{S_j}) \wedge f_1(\vec{x}^*_{S_i}) - f_2(\vec{x}^*_{S_j}) = \vec{0}\ , \tag{3}$$

где $\vec{x}^*_{S_i}$ и $\vec{x}^*_{S_j}$ – комбинации конкретных значений индексных переменных, символ «$\prec$» обозначает отношение «меньше» в лексикографическом смысле [4]. В данной работе в качестве аппроксимации множества пар используются вектора расстояний. Вектором расстояния $\vec{v}_{i,j}$ называется вектор, для которого справедливо условие:

$$\vec{x}^*_{S_j} = \vec{x}^*_{S_i} + \vec{v}_{i,j} \tag{4}$$

для пар $\left\{\vec{x}^*_{S_i}, \vec{x}^*_{S_j}\right\}$, удовлетворяющих (3) [5].

Многогранным сокращенным графом зависимостей (МСГЗ) называют граф, каждая вершина которого представляет собой инструкцию S_i вложенности циклов, дуга из S_i присутствует S_j в том случае, если инструкция S_j зависит по данным от S_i. При этом каждая такая дуга помечается многогранником D_{S_i,S_j}, аппроксимирующим множество векторов расстояния для данной зависимости [5].

2. Алгоритм поиска параллельности

Ключевая идея алгоритма заключается в отображении экземпляров инструкций исходной вложенности в Z^1. В данном случае каждую точки Z^1 следует интерпретировать, как логическую дату выполнения некоторого множества экземпляров инструкций. В одну точку Z^1 (на один момент времени) должны отображаться экземпляры инструкций, между которыми отсутствуют зависимости по данным [2].

Результатом работы алгоритма поиска параллельности является аффинное отображение Θ_{S_i}, построенное для каждой инструкции S_i исходной вложенности [2]:

$$\Theta_{S_i} = T\begin{pmatrix} \vec{x}_{S_i} \\ \vec{n}_{S_i} \\ 1 \end{pmatrix}. \qquad (5)$$

Функция Θ_{S_i} определяет момент времени выполнения для каждого экземпляра инструкции S_i.

Ниже представлен алгоритм поиска функций вида (5).

Вход: МСГЗ исходной вложенности циклов.

Выход: набор функций Θ_{S_i} для каждой инструкции S_i исходной вложенности.

Алгоритм: для каждой дуги МСГЗ, проведенной из вершины S_i в вершину S_j, выполнить.

1. Составить ограничение, не позволяющее зависимым экземплярам S_j выполняться раньше экземпляров S_i [2]:
   $$\Theta_{S_j}(\vec{x}_{S_j}) - \Theta_{S_i}(\vec{x}_{S_i}) - 1 \geq 0. \qquad (6)$$
2. Составить прообраз аффинной функции, положительной на всем многограннике зависимостей D_{S_i,S_j}, используя для этого аффинную формулировку леммы Фаркаша [6].
3. Установить соответствие между коэффициентами полученного прообраза и коэффициентами неравенства (6). Результатом данного шага является система линейных уравнений от искомых коэффициентов функций отображения Θ_{S_i} и Θ_{S_j}, а так же коэффициентов прообраза аффинной функции, полученного на шаге 2.
4. Получить систему уравнений и неравенств, объединив уравнения предыдущего шага и неравенства, требующие неотрицательности искомых коэффициентов и коэффициентов прообраза, полученного на шаге 2 (данные неравенства являются следствием использования аффинной формулировки леммы Фаркаша).
5. Найти лексикографически минимальное решение полученной системы уравнений и неравенств, установив предварительно порядок следования коэффициентов искомых уравнений Θ_{S_i} и Θ_{S_j}.

Следует отметить, что изменения порядка искомых коэффициентов может привести к различным формам параллелизма.

3. Пример

Предлагается рассмотреть результаты работы алгоритма для следующей исходной вложенности циклов:

```
for (i = 0; i <= N - 1; i++)
  for (j = 0; j <= N - 1; j++)
  {
     X[i][j] = X[i][j] * sqrt(Y[i - 1][j]);          /* S1 */
    Y[i][j] = Y[i][j] * sqrt(X[i][j - 1]);           /* S2 */
  }
```

Между инструкциями данной вложенности присутствует перекрестная зависимость.

Используя различный порядок следования искомых коэффициентов (шаг 5), были получены следующие функции отображения (табл. 1).

Таблица 1. Функции отображения

Функции отображения	**Выполняемые преобразования**
$\Theta_{S_1}(\vec{x}_{S_1}) = 2j-1, \Theta_{S_2}(\vec{x}_{S_2}) = 2j-2$	Перестановка циклов, разрезание цикла по *i*
$\Theta_{S_1}(\vec{x}_{S_1}) = i+j-2, \Theta_{S_2}(\vec{x}_{S_2}) = i+j-2$	Волновое распараллеливание
$\Theta_{S_1}(\vec{x}_{S_1}) = 2i, \Theta_{S_2}(\vec{x}_{S_2}) = 2i+1$	Разрезание цикла по *j*

На основе данных функций были получены три параллельные версии исходной вложенности соответственно.

Вариант для CPU (перестановка циклов, разрезание цикла по *i*):

```
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    y[p2][1] = y[p2][1] * sqrt (x[p2][1-1]);
#pragma omp parallel for shared (x, y) private (p1, p2)
  for (p1=1;p1<=2*N-2;p1++)
    for (p2=1;p2<=N;p2++)
    {
      if ((p1+1)%2 == 0)
        x[p2][(p1+1)/2] = x[p2][(p1+1)/2] * sqrt (y[p2-1][(p1+1)/2]);
      if (p1%2 == 0)
        y[p2][(p1+2)/2] = y[p2][(p1+2)/2] * sqrt (x[p2][((p1+2)/2)-1]);
    }
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    x[p2][N] = x[p2][N] * sqrt (y[p2-1][N]);
```

Вариант для CPU (волновое распараллеливание):

```
for (p1=0;p1<=2*N-2;p1++)
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=MAX(1,p1-N+2);p2<=MIN(N,p1+1);p2++)
  {
    x[p2][p1-p2+2] = x[p2][p1-p2+2] * sqrt (y[p2-1][p1-p2+2]);
    y[p2][p1-p2+2] = y[p2][p1-p2+2] * sqrt (x[p2][p1-p2+2-1]);
  }
```

Вариант для CPU (разрезание цикла по *j*):

```
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    x[1][p2] = x[1][p2] * sqrt(y[1-1][p2]);
#pragma omp parallel for shared (x, y) private (p1, p2)
  for (p1=3;p1<=2*N;p1++)
    for (p2=1;p2<=N;p2++)
    {
      if ((p1+1)%2 == 0)
        y[(p1-1)/2][p2] = y[(p1-1)/2][p2] * sqrt (x[(p1-1)/2][p2-1]);
      if (p1%2 == 0)
        x[p1/2][p2] = x[p1/2][p2] * sqrt(y[(p1/2)-1][p2]);
    }
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    y[N][p2] = y[N][p2] * sqrt(x[N][p2-1]);
```

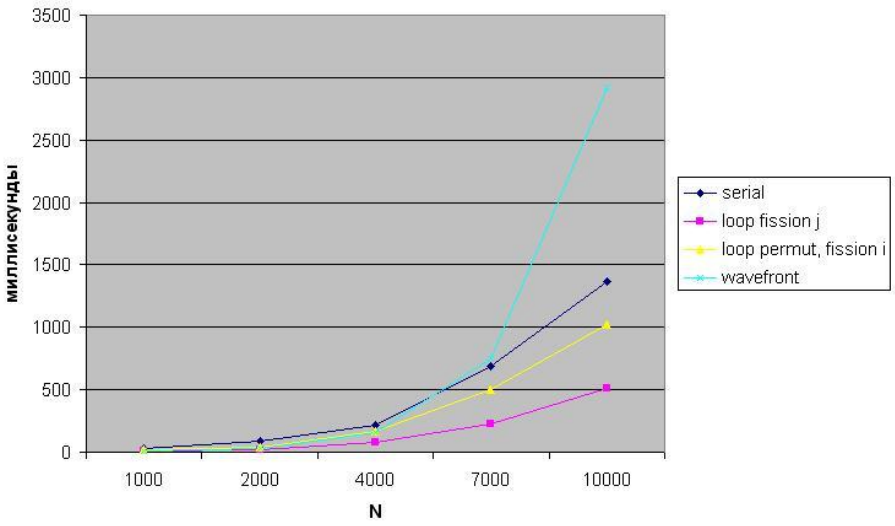


Рис. 1. Результаты временных замеров

Для оценки быстродействия программ выполнялись замеры времени их выполнения (рис. 1) для входных массивов различной длины (различных значений *N*). Для вычислений в данной работе использовалось следующее программное обеспечение и оборудование. Операционная система Debian Linux 6.0.4, CPU: Intel Pentium Dual-Core 2,6 ГГц E5300, RAM – 2048 Мб. Сборка программ выполнялась с помощью компилятора icc, с опциями –fast, –openmp, –xSSSE3.

Литература

1. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. Ростов-на-Дону: Издательство Ростовского университета, 2004. 192 с.
2. Pouchet L., Bastoul C. Iterative optimization in the polyhedral model: part 1, one-dimensional time // Proceedings of the International Symposium on Code Generation and Optimization, CGO’07. Washington: IEEE Computer Society, 2007. P. 144-156.
3. Ахо А., Лам М., Сети Р. Компиляторы. Принципы, технологии и инструментарий, 2-е изд. М.: Вильямс, 2008. 1184 с.
4. Lim A., Lam M. Maximizing parallelism and minimizing synchronization with affine transform // Proceedings of the 24th ACM SIGPLAN-SIGACT. N.Y.: ACM, 1997. P. 215–227.
5. Касьянов В.Н., Мирзуитова И.Л. Реструктурирующие преобразования: алгоритмы распараллеливания циклов // Программные средства и математические основы информатики. Новосибирск: Институт систем информатики им. А.П. Ершова СО РАН, 2004. С. 142–188.
6. Feautrier P. Some efficient solutions to the affine scheduling problem. One-dimensional time // International Journal of Parallel Programming. Norwell: Kluwer Academic Publishers, 1992. V. 21. P. 313-348.

СИСТЕМА WBS КАК РАСШИРЕННЫЙ ИНСТРУМЕНТ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ СРЕДОЙ

Д.С. Одякова, Г.В. Тарасов, Д.И. Харитонов

Институт автоматики и процессов управления ДВО РАН, Владивосток

Рассмотрены принципы функционирования системы WBS (Web Batch System), разработанной в ИАПУ ДВО РАН. Основной функцией системы WBS является организация прохождения заданий в многопроцессорной вычислительной среде. Благодаря архитектурным особенностям и формальному подходу к формированию заданий, наряду с заданиями пользователей система позволяет выполнять административные задания по тестированию частных характеристик вычислительных узлов, а также по проверке работоспособности вычислительной среды.

Введение

К настоящему времени сформировался образ среды для научных расчётов, удовлетворяющий следующим требованиям: во-первых, это многопроцессорная система, реализованная в большинстве случаев в виде кластера или GRID-среды; во-вторых, это длительный цикл эксплуатации и отказоустойчивость; и в-третьих, предоставление специфического сервиса доступа пользователей к ресурсам и разделения ресурсов, включая запуск задач, требующих специального программного обеспечения. Со временем каждая вычислительная среда для научных расчётов обрастает своими пользователями. Пользователи формируют перечень прикладного программного обеспечения (ПО), которое должно поддерживаться вычислительной средой. Однако существует ряд факторов, препятствующих сохранению работоспособности ПО в вычислительной среде в течении длительного интервала времени. В современном мире согласно закону Мура, как минимум раз в пять лет приходится заниматься обновлением оборудования, что влечёт за собой как минимум обновление низкоуровневых драйверов операционной системы. Новым пользователям требуется установка нового ПО. В последнее время актуальной является установка критических обновлений системы, связанных с безопасностью. Любая из трёх вышеперечисленных причин может привести к необходимости обновления всего программного стека, после которого естественным образом возникает проблема проверки работоспособности как системы в целом, так и всего перечня прикладного программного обеспечения пользователей. К сожалению, невозможно проверить свойства системы, требуемые для различного ПО, исходя из списка установленных файлов. Альтернативным путем является использование множества тестов, проверяющих частные требования. Тестирование в распределённой гетерогенной среде является трудоёмким процессом, поэтому предлагается автоматизировать процесс запуска тестов и сбора информации о результатах тестирования, необходимых для проверки характеристик вычислительной среды.

В рамках системы WBS (Web Batch System), разрабатываемой в ИАПУ ДВО РАН, цикл задач, связанных с обслуживанием вычислительной среды, может быть представлен образом, идентичным вычислительным задачам пользователей. Первичный запуск административной задачи в системе сохраняется в виде шаблона в базе данных, шаблон запуска задачи параметризуется, ему назначаются специфические свойства приоритетности и прав доступа. Готовый шаблон может быть использован в процессе автомати-

зированного управления вычислительной средой. Далее рассматривается общая архитектура системы WBS, основной аппарат формирования заданий в системе, а также применение системы для управления различными заданиями в распределённой вычислительной среде.

1. Архитектура WBS

Архитектура WBS состоит из 6 компонент, представленных на рис. 1. Компоненты делятся по зонам доступа: на внешний интерфейс пользователя, распределённая система управления на основе SQL-сервера и NFS-сети, а также внутренний интерфейс доступа к вычислительным ресурсам.

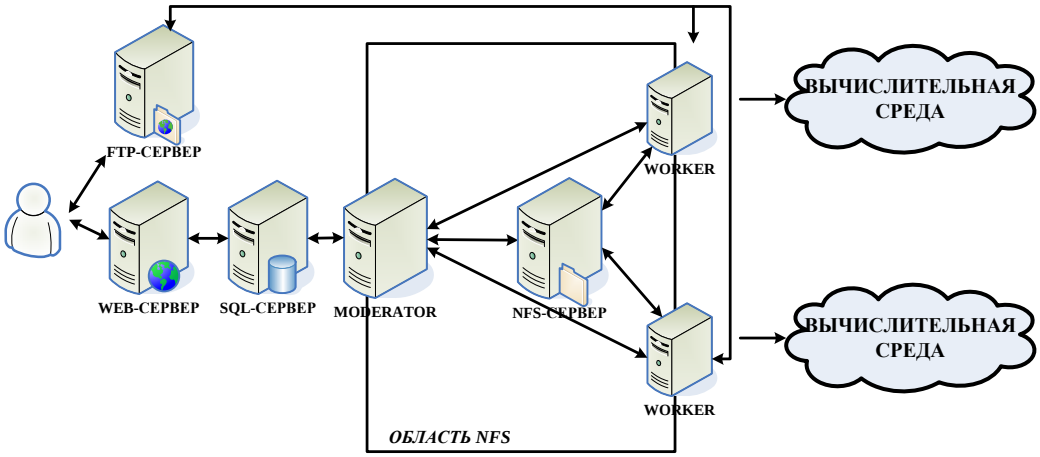


Рис. 1. Архитектура WBS

Web-сервер предоставляет основной интрефейс пользователям к системе WBS через интернет-браузер. Он реализован на основе www-сервера Apache с использованием языков PHP и JavaScript, а также библиотеки jQuery. Позволяет пользователям и администраторам системы управлять заданиями (создавать, удалять, редактировать, проверять статус) через интернет с помощью веб-браузера.

SQL-сервер (EnterpriseDB или PostgreSQL) является ключевым компонентом системы управления, который хранит информацию о пользователях, задачах, тестах, конфигурациях вычислительной системы и формирует файлы заданий. Все задания, выполняемые системой WBS, начинаются и заканчиваются на SQL-сервере, который при необходимости обращается к MODERATOR'у с требованием обновить список заданий, нуждающихся в выполнении.

MODERATOR – это часть системы управления, он выполняет связующую функцию между SQL и управляющими узлами вычислительной системы, является программой-демоном, работающей в многопоточном режиме. MODERATOR выполняет следующие основные функции: забирает задания пользователей из SQL, кладёт файлы заданий в соответствующие директории на NFS-сервере, посылает сообщения, сигнализирующие о поступлении нового задания, соответсвующим WORKER'ам, дожидается сообщений о выполнении заданий от WORKER'ов, вносит на SQL-сервер информацию о завершении задач.

WORKER'ы отвечают за взаимодействие с вычислительными ресурсами, являются программами-демонами, работающими в многопоточном режиме. В силу своей специфики компоненты WORKER должны располагаться во внутренней сети вычислительных ресурсов, причем, количество WORKER'ов может варьироваться в зависимости от способа запуска заданий (GRAM, OpenPBS, SLURM и т.д.). После того как WORKER получил задание, всё, что ему необходимо выполнить, описано специальным образом в файле-инструкции. Описанные инструкции WORKER выполняет последовательно, по-

сле того как задание выполнено, отправляет сообщение MODERATOR'у. При необходимости WORKER имеет возможность обращаться к FTP-серверу, чтобы класть или забирать файлы с данными.

NFS-сервер является буфером обмена данными между MODERATOR'ом и WORKER'ами.

FTP-сервер используется в качестве хранилища больших входных и выходных файлов с данными задач пользователей.

2. Генерация скриптов (шаблонов)

Основная идея подхода, использованного при создании WBS, состоит в том, что любые операции пользователей и администраторов вычислительной среды можно представить в виде задачи. Задача проходит определенный жизненный цикл: создание (CREATING), редактирование (EDITING), ожидающая запуска (PENDING), выполнение (RUNNING), окончание (DONE). До начала запуска каждая задача в системе WBS представляется посредством шаблонов и набора параметров к ним. Шаблон описан в виде последовательности параметризуемых действий, необходимых для выполнения задачи. На основе шаблона генерируются файлы, необходимые для выполнения задания на указанном вычислительном ресурсе. Файлы прикладных задач могут быть нескольких типов.

Файл-инструкция для WORKER'а, представляет собой xml-подобный файл, в котором последовательно перечислены инструкции. В настоящее время используется четыре типа инструкций. Во-первых, инструкции обращения к командному интерпретатору (command). Во-вторых, указание WORKER'у временно приостановить выполнение инструкций (wait) и ожидать сигнала продолжения. В-третьих, указание разобрать заданный текстовый файл (parse) и заменить переменные, маркированные специальным образом, на значения, определённые для конкретного WORKER'а. В четвёртых, указание для копирования файлов данных задачи из и на FTP-сервер.

Файл паспорта задачи, относящийся непосредственно к запуску задачи на выполнение, может отличаться в зависимости от типа системы, на которой выполняется задание – OpenPBS, SLURM, GRAM и т.д.

Третий тип – это специфические файлы настроек задачи, представленные в текстовом виде и расположенные в SQL.

Четвёртый тип – это большие файлы задачи (файлы с выходными или входными данными), которые хранятся на FTP-сервере. Входные файлы к моменту запуска задачи пользователь уже должен разместить на FTP-сервере в директорию задачи и указать список файлов в настройках запуска.

Файлы первых трёх типов генерируются на SQL-сервере с использованием следующей универсальной процедуры. Шаблон файлов задания представляется в виде графа, каждый узел которого имеет собственные параметры и множество инцидентных узлов. При заведении нового задания пользователь, редактируя параметры задачи в интерактивных страницах браузера, задаёт структурное дерево параметров файла, каждому узлу которого ставится в соответсвие узел шаблонного графа параметров файла. Обход этого дерева происходит сверху вниз (вглубь) и слева на право (в ширину). При заходе и выходе из каждого узла дерева вызываются скрипты формирования файла, определенные администратором для соответствующего узла шаблонного графа параметров файла. Эти скрипты сохраняют в файле задания текстовое представление структуры узла шаблона. Для создания нового шаблона можно либо скопировать, либо наследовать свойства уже существующего (родительского) шаблона. Создание шаблонов задач осуществляется администраторами системы WBS.

3. Область применения WBS

Благодаря тому, что задачи описаны в SQL с помощью шаблонов, а остальные компоненты системы WBS являются исполнителями с чётко определённой функциональностью, область применения WBS может быть достаточно широкой. В настоящее время в суперкомпьютерном центре ИАПУ ДВО РАН WBS используется в качестве 1) системы управления заданиями пользователей на вычислительных ресурсах центра; 2) инструмента тестирования частных характеристик вычислительной среды и её отдельных компонент; 3) расширенного инструмента для администрирования гетерогенной и проверки целостности вычислительной среды.

При управлении заданиями пользователей на SQL-сервере формируется множество шаблонов заданий. Используя эти шаблоны, пользователь имеет возможность создавать, редактировать, удалять, а также просматривать состояние своих задач. Манипулирование шаблонами осуществляется администраторами системы. Пользователи только применяют готовые шаблоны для создания своих задач, задавая значения параметров через веб-браузер. Процесс генерации файлов заданий, копирования их на вычислительный узел, запуска и получения результатов является полностью автоматическим. Пользователь может лишь наблюдать состояния задач и прерывать их исполнение при необходимости.

При тестировании характеристик вычислительной среды на SQL-сервере формируется множество шаблонов тестов заданий. Для совместимости теста с различным аппаратным и программным обеспечением вычислительных узлов исходные тексты тестов располагаются в сетевой директории, доступной с каждого из узлов вычислительной системы. Каждый тест располагается в собственной директории и имеет единственный исполняемый скриптовый файл, совпадающий с названием директории. При запуске тестового задания исходные тексты копируются на тестовый узел. Тест компилируется и запускается. В процессе выполнения теста вывод сохраняется на SQL-сервере, что позволяет администраторам собирать статистическую информацию о системе. В частности можно получить информацию о последнем тестировании, о тестировании на заданное число, график изменения тестированных характеристик (что является полезным при настройке операционной системы и прикладных библиотек вычислительных узлов).

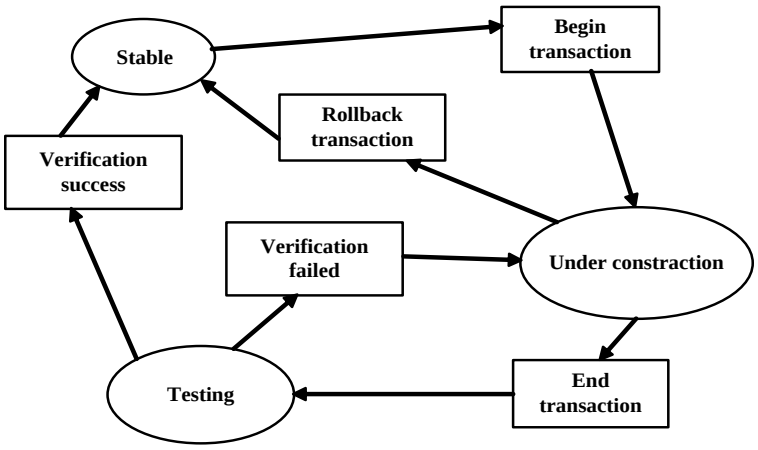


Рис. 2. Диаграмма состояний вычислительных узлов.

При администрировании вычислительной среды необходимо: контролировать установку программного обеспечения на каждом узле программной среды; предотвращать несанкционированную установку программного обеспечения; проверять работоспособность программного обеспечения; хранить историю изменений системы и отоб-

ражать текущее состояние системы. Для выполнения поставленных задач в ИАПУ ДВО РАН разрабатывается программная система OSeye, использующая возможности WBS для запуска задач администрирования и тестирования вычислительных узлов.

При администрировании вычислительной системы каждому узлу назначается одно из трёх состояний (рис. 2): STABLE – в этом состоянии узел может использоваться в обычном штатном цикле, установка нового ПО заблокирована; UNDER CONSTRUCTION – в этом состоянии на узле могут проводиться манипуляции с установленным ПО, а также установка нового ПО; TESTING – в этом состоянии в автоматическом режиме запускаются тесты из предписанного набора. В состоянии UNDER CONSTRUCTION администратор производит изменения в системе, после которых необходимо убедиться, что система находится в работоспособном состоянии, под которым понимается возможность компиляции и запуска распределённых задач пользвателей. Основная задача OSeye сводится к переводу из состояния UNDER CONSTRUCTION в STABLE, условием которого является успешная проверка работоспособности системы.

Для обеспечения работоспособности распределённой среды необходимо контролировать большое количество существенно отличных друг от друга параметров, таких как: функционирование Ethernet, Myrinet, Infiniband сетей, наличие и корректность настройки SSH протокола, наличие языков программирования и библиотек, наличие и доступность прикладных программных пакетов и библиотек, корректность функционирования системы управления прохождением заданий. Для контроля работоспособности программного обеспечения используется процесс тестирования, который позволяет абстрагироваться от понятий различных предметных областей в пользу унифицированного интерфейса тестирования системы.

Заключение

В настоящей статье рассмотрена архитектрура системы WBS, неформально описан способ генерации файлов заданий на основе шаблонов в SQL, а также описаны примеры использования WBS в качестве расширенного инструмента управления вычислительной средой. Управление вычислительной средой реализовано через выполнение заданий WBS, при помощи которых может контролироваться работоспособность узлов системы и тестирование частных характеристик её компонентов.

Литература

1. Голенков Е.А., Левин В.А., Харитонов Д.И., Шиян Д.С. Организация поддержки циклических расчетов в GRID-сегменте ДВО РАН. - Горный информационно-аналитический бюллетень (научно-технический журнал) Mining informational and analytical bulletin (scientific and technical journal). 2009. Т. 18. № 12. С. 225–229.
2. Бабяк П.В., Тарасов Г.В. Организация потоковой обработки спутниковых данных на основе GRID-технологий // Горный информационно-аналитический бюллетень (научно-технический журнал) Mining informational and analytical bulletin (scientific and technical journal). 2009. Т. 18. № 12. С. 170–175.

ИДЕНТИФИКАЦИЯ МОДЕЛЕЙ ЭКОНОМИКИ СО СТРУКТУРНЫМИ ИЗМЕНЕНИЯМИ

Н.Н. Оленёв

Вычислительный центр им. А.А. Дородницына РАН, Москва

Модели экономики, учитывающие структурные изменения, трудно использовать из-за сложностей в их идентификации. В настоящей работе предложен способ идентификации моделей со структурными изменениями, основанный на использовании параллельных вычислений. Дополнительные сложности связаны с тем, что часть параметров может резко менять значения при изменении условий функционирования системы. Здесь применяют не только формальные методы, но и эвристические.

Введение

Алгоритмы параллельных вычислений при идентификации моделей экономики, учитывающих структурные изменения, меняются. Даже при идентификации нормативных балансовых динамических моделей [1] учет структурных изменений потребует дополнительного внимания, поскольку нормативы здесь будут кусочно-постоянными на разных интервалах времени, границы которых надо также идентифицировать.

Идентификация моделей экономики с производственными фондами, дифференцированными по моментам создания (vintage capital model) [2], является более сложной задачей. Но задача идентификации таких моделей чрезвычайно важна, поскольку учет структурных изменений в производстве дает возможность оценить возможные пути развития современных экономических систем, в частности, перспективы инновационного развития России и ее регионов.

Модель экономики с учетом поколений капитала

Полвека назад модели экономического роста с капиталом, дифференцированным по моментам создания были сердцевиной теории роста [3]. Западные исследования в этом направлении резко прекратились в конце 60-х вместе с обнаружением так называемого противоречия воплощения и техническими сложностями в исследовании моделей. В России исследования моделей экономики с производственными мошностями, распределениями по возрасту продолжались до начала 90-х годов [2], когда они практически прекратились вместе с резким сокращением финансирования фундаментальных исследований. В то же самое время эти исследования вновь возродились на Западе. На это, в частности, повлияли три прорыва в методологии исследований [3], объясняющие этот всплеск: (1) революция в статистическом учете роста, которая использует в своих интересах доступность новых временных рядов, (2) революция в теории оптимального управления, позволяющая благополучно изучить модели роста с распределением капитала по возрасту, (3) достижения в экономической демографии, учитывающей возрастную структуру человеческого капитала наряду с возрастной структурой физического капитала. Исследования западных ученых подошли к уровню, на котором они у нас были заброшены, а в исследованиях человеческого капитала, распределенного по возрасту, уже превзошли наши исследования, значит нам стоит вновь присоединиться к этому, теперь уже совместному, направлению исследований.

Рассмотрим односекторную модель экономики с производствнными мощностями, распределенными по моментам создания. В этой модели мощности с течением времени экспонециально убывают, а число рабочих мест сохраняется с момента создания производственной мощности до момента ее уничтожения. Тогда динамика производственных мощностей на микроуровне описывается уравнениями:

$$m(t,\tau) = m(\tau,\tau)\exp\left(-\mu(t-\tau)\right), \quad \lambda(t,\tau) = \lambda(\tau,\tau)\exp\left(\mu(t-\tau)\right). \qquad (1)$$

Здесь $m(t,\tau)$ – величина производственной мощности, созданной в момент времени τ, к моменту времени t, параметр μ – темп уменьшения мощностей и темп возрастания трудоемкости. Агрегирование такой микромодели с целью получения макроописания можно осуществить численно, если у нас имеется достаточно вычислительных ресурсов. В частных случаях на основе этой модели можно получить аналитические выражения для производственной функции [4]. В общем случае агрегирование можно всегда осуществить численно на основе описания модели на микроуровне.

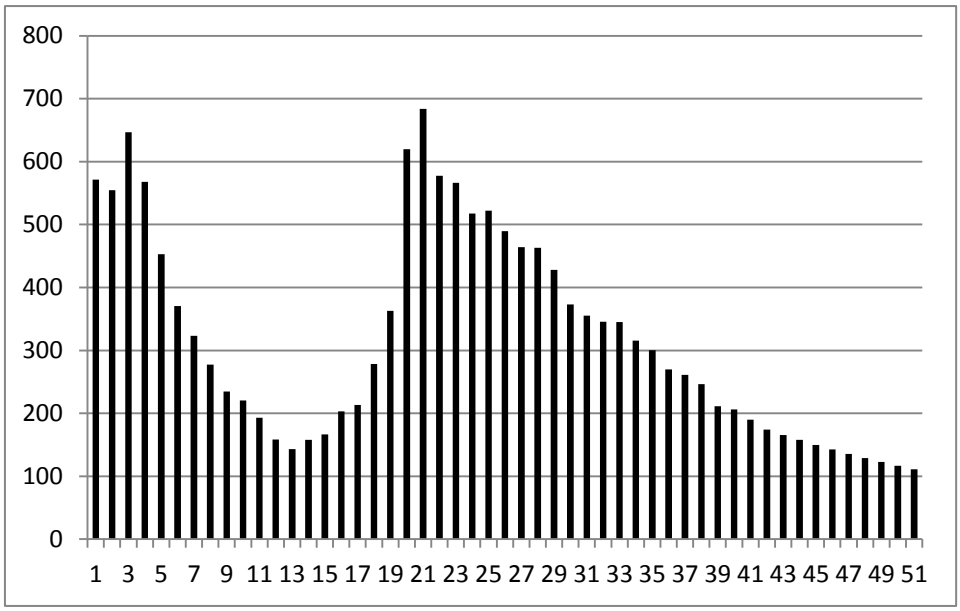


Рис. 1. Распределение производственных мощностей РФ по возрасту на начало 2011 г.

Производственная мощность – это максимально возможный выпуск. Производственная мощность определенного возраста определяется объемом инвестиций в соответствующем году создания мощностей, коэффициентом фондоемкости – линейным коэффициентом пересчета фондообразующего продукта в производственную мощность, а также темпом деградации мощностей. Суммарная производственная мощность задает потенциал выпуска – максимально возможный объем ВВП. Такая модель [2] учитывает производственные возможности и задает границы изменения выпуска.

Параллельные вычисления для определения параметров такой модели проводились на кластерном суперкомпьютере МСЦ РАН. Параметры определялись косвенным образом при сравнении результатов расчета макроэкономических показателей с их статистическими аналогами.

Работа выполнена при поддержке РФФИ (проекты №№ 12-01-00916, 11-07-97017-р_поволжье_а), ПФИ Президиума РАН № 14, ПФИ ОМН РАН №3.

Литература

1. Гергель В.П., Горбачев В.А., Оленев Н.Н., Рябов В.В., Сидоров С.В. Параллельные методы глобальной оптимизации в идентификации динамической балансовой нор-

мативной модели региональной экономики // Вестник ЮУрГУ, №25 (242), 2011. С. 4–15. (Сер. "Математическое моделирование и программирование", вып.9).
2. Оленев Н.Н., Поспелов И.Г. Исследование инвестиционной политики фирм в экономической системе рыночного типа // Математическое моделирование: Методы описания и исследования сложных систем. – М.: Наука, 1989. С. 175–200.
3. Boucekkine R., de la Croix D., Licandro O. Vintage capital growth theory: Three breakthroughs // Barcelona GSE Working Paper Series. Working Paper No. 565. June 2011. 30 p.
4. Оленев Н.Н., Петров А.А., Поспелов И.Г. Модель процесса изменения мощности и производственная функция отрасли хозяйства // Математическое моделирование: Процессы в сложных экономических и экологических системах. М.: Наука, 1986. С. 46–60.

ГЛОБАЛЬНЫЙ ПОИСК В ЗАДАЧАХ СВЕРХВЫСОКОЙ РАЗМЕРНОСТИ НА ПРИМЕРЕ ИДЕНТИФИКАЦИИ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ЭКОНОМИКИ НИЖЕГОРОДСКОЙ ОБЛАСТИ

Н.Н. Оленёв[1], В.В. Рябов[2]

[1]Вычислительный центр РАН, Москва

[2]Нижегородский госуниверситет им. Н.И. Лобачевского

Данная работа продолжает совместные исследования ВЦ РАН и ННГУ по построению и идентификации математической модели Нижегородской области, завершая второй этап: идентификации параметров уже построенной и уточнённой модели с неулучшаемым на данный момент числом независимых параметров (N=57). Для решения задачи такой размерности применяются эффективные алгоритмы, развиваемые нижегородской научной школой глобальной оптимизации.

1. Задача идентификации модели региональной экономики

Оценка экономического развития региона особенно важна во время переходных процессов от одной стратегии развития к другой. Для адекватности используемой модели требуется, чтобы она была должным образом идентифицирована по исторической статистике. Другими словами, требуется определить параметры модели, при которых ее поведение в прошлом соответствует историческим данным, то есть должная идентификация модели подразумевает ее верификацию по временным рядам макропоказателей изучаемой региональной экономики. Идентифицированная имитационная модель региональной экономики дает возможность получить количественную оценку динамики макропоказателей экономики региона, включая оценку экономического потенциала, показатели структуры человеческого капитала и его динамики.

Задача идентификации многосекторных моделей региональной экономики заключается в поиске значений неизвестных параметров модели, при которых результаты расчетов по модели временных рядов макропоказателей экономики близки к историческим статистическим данным для этих временных рядов. Критерием близости расчетных и исторических статистических временных рядов может служить свертка критериев Тейла для каждого макропоказателя [1]. Здесь исследование сталкивается с «проклятием размерности». Число параметров, которые нельзя определить напрямую из статистики, оказывается для таких моделей настолько велико, что прямое применение параллельных вычислений на суперкомпьютерах (перебор параметров по равномерной сетке) оказывается бессильным. Требуются новые пути решения для задачи идентификации многосекторных моделей экономики с помощью параллельных вычислений.

Применение эффективных параллельных алгоритмов глобальной оптимизации, развиваемых в Нижегородском государственном университете им. Н.И. Лобачевского, позволяет вплотную приблизиться к решению подобных задач. Огромное количество независимых параметров модели (изначально N=60, на данный момент N=57) требует применения всех возможных способов ускорения сходимости, а также построения дополнительных локальных оценок оптимума с использованием метода Хука-Дживса. Применение алгоритмов с высокой эффективностью распараллеливания на кластере ННГУ позволяет достичь предела вычислительных ресурсов.

2. Трехсекторная модель экономики Нижегородской области

Полное описание модели обширно, оно представлено в [1]. Здесь укажем только основные положения. При построении модели экономики региона выделено восемь экономических агентов. Во-первых, три производителя, которые в модели представлены тремя секторами экономики Нижегородской области: первый сектор включает добывающие и инфраструктурные отрасли, (2) второй – обрабатывающие отрасли региональной экономики, (3) третий – отрасли услуг, включая финансовые услуги. Во-вторых, три основных потребителя конечной продукции: (4) домашние хозяйства, (5) Правительство региона, (6) внешние для региона потребители. Замыкает модель описание поведения еще двух экономических агентов: (7) банковской системы региона, (8) торгового посредника. Модель экономики региона описывает динамику материальных и финансовых балансов через изменение запасов денег, продуктов и факторов производства. Модель учитывает налогообложение производителей и домашних хозяйств, а также теневой оборот в сфере производства и потребления. Производители поставляют продукцию конечного потребления на внутренние и внешние рынки, а также промежуточную продукцию в смежные сектора экономики. Домашние хозяйства потребляют конечную продукцию и предлагают труд производственным секторам экономики. Банковская система выпускает денежные средства и выдает кредиты производителям. Правительство региона формирует консолидированный региональный бюджет, собирая налоги, которые в модели представлены семью ставками: 1) налог на прибыль, 2) налог на добавленную стоимость, 3) акцизы, 4) единый социальный налог, 5) пошлины на вывоз продукции, 6) пошлины на ввоз продукции, 7) подоходный налог. Кроме того, Правительство региона распределяет полученные средства по основным экономическим агентам. Модель учитывает теневой оборот, который не облагается налогами.

Банковская система Нижегородской области не является замкнутой, большую роль в инвестиционных решениях играют филиалы Российских банков других регионов. В качестве первого приближения при описании банковской системы Нижегородской области предполагалось, что часть золотовалютных резервов Российской федерации обеспечивает резервирование активов Нижегородской области. Считаем, что банковские активы областной банковской системы состоят из золото-валютных резервов и суммарной задолженности производственных секторов экономики региона, а пассивы – из суммарных запасов денег у контрагентов банковской системы, которые подчиняются финансовому балансу банковской системы.

Полная модель содержит более ста соотношений. Число внешних по отношению к модели параметров, которые невозможно определить напрямую из данных статистики, удалось сократить до 63 за счет использования полученных нами соотношений между параметрами на равновесном начале. Казалось бы, подобного типа нормативные модели совершенно бессмысленны, поскольку для проведения осмысленных качественных и количественных расчетов на них необходимо провести идентификацию их внешних параметров, а большая часть параметров не может быть оценена напрямую из данных экономической статистики. Нами предложена новая технология идентификации внешних параметров модели, основанная на использовании высокоскоростных параллельных вычислений на многопроцессорной системе. Для идентификации параметров сравниваются полученные при расчетах на модели временные ряды макропоказателей экономики региона с соответствующими статистическими временными рядами. В качестве критериев близости временных рядов использован индекс несовпадения Тейла. Критерием качества идентификации параметров модели является количественное соответствие основных макроэкономических показателей статистическим показателям экономики региона за период с 2000-2010 гг.

3. Эффективные алгоритмы многоэкстремальной оптимизации

Алгоритмы, развиваемые Нижегородской научной школой многоэкстремальной оптимизации, предполагают следующую постановку задачи:

$$\varphi^*=\varphi(y^*)=\min\{\varphi(y):\ y\in D\},\qquad D=\{y\in R^N:\ a_i\le y_i\le b_i,\ 1\le i\le N\},\tag{1}$$

где целевая функция $\varphi(y)$ удовлетворяет условию Липшица с соответствующей константой L, а именно

$$|\varphi(y_1)-\varphi(y_2)|\le L\|y_1-y_2\|,\ y_1,y_2\in D.$$

Используя кривые типа развертки Пеано $y(x)$, однозначно отображающие отрезок [0,1] на N-мерный гиперкуб P

$$P=\{y\in R^N:\ -2^{-1}\le y_i\le 2^{-1},\ 1\le i\le N\}=\{y(x):\ 0\le x\le 1\},$$

исходную задачу можно редуцировать к следующей одномерной задаче:

$$\varphi(y_D(x^*))=\min\{\varphi(y_D(x)):\ x\in[0,1]\}.$$

Рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией одномерную задачу, в которой целевая функция удовлетворяет равномерному условию Гельдера (см. [5]), т.е.

$$|\varphi(y_D(x'))-\varphi(y_D(x''))|\le K|x'-x''|^{1/N},\ x',x''\in[0,1],$$

где N есть размерность исходной многомерной задачи, а коэффициент K связан с константой Липшица L исходной задачи соотношением $K\le 4L\sqrt{N}$.

Различные варианты индексного алгоритма для решения одномерных задач и соответствующая теория сходимости представлены в работах [4, 5].

Параллельная версия индексного метода основана на построении множественных отображений Пеано, получаемых путём сдвига или вращения гиперкубов друг относительно друга (сдвиговые и вращаемые развёртки), что позволяет заодно и улучшать сходимость алгоритма за счёт более точной адаптивной оценки неизвестной константы Гёльдера K в процессе вычислений. Целым рядом преимуществ обладает схема построения вращаемых развёрток, предложенная в работе [7] и позволяющая использовать до N (N –1) + 1 вычислительных ядер.

Поскольку поиск глобального оптимума занимает длительное время, значительный интерес представляет также оценка локального оптимума, в области притяжения которого найден последний рекорд. Для получения такой оценки используется метод Хука-Дживса [6]. Он относится к прямым методам (или методам нулевого порядка), т.е. использует в ходе работы только значения целевой функции, не требуя ее дифференцируемости. Это хорошо согласуется с требованием липшицевости целевой функции в задаче (1).

В дополнение к этому, точки траектории спуска отображаются с использованием кривой Пеано на одномерный отрезок и используются для адаптивной оценки константы Гельдера K в ходе решения задачи (1). Таким образом, локальный спуск и алгоритм построения глобального покрытия области поиска используют поисковую информацию, полученную друг у друга.

Ещё одна модификация состоит в использовании последовательности приближений кривых Пеано к кривой Гильберта (развёртки растущего уровня детализации).

4. Развертки растущего уровня детализации

Изложенные ранее подходы, сохраняющие часть информации о близости точек в многомерном гиперкубе при отображении на отрезок, используют множественные отображения, основанные на аффинных преобразованиях (смещениях, поворотах) исходного отображения. Однако следует отметить, что потеря информации о близости точек усугубляется и с ростом детализации развёртки.

Поэтому в рамках данной работы предлагается подход, основанный на построении отображений различной степени детализации. Причём предлагаемая модификация способна дополнять подходы на основе поворотов и вращений.

Модификация состоит в том, чтобы выполнять поиск сначала на «грубой» сетке и постепенно увеличивать детализацию развертки, тем самым усложняя получаемую одномерную целевую функцию $\varphi(y_m(x)) \to \varphi(y_{m+1}(x))$ и увеличивая предельную точность поиска. При этом необходимо определить правило перехода на следующий уровень детализации. Если рассматривать данный подход как решение последовательности задач с различным m, то очевидными условиями перехода служат:

1) достижение такой точности поиска, что при текущем m две различные точки гиперкуба сливаются в одну на отрезке;
2) остановка по общей заданной точности ε для всех подзадач;
3) остановка по максимальному числу испытаний для каждой подзадачи.

Процедура перехода от задачи с порядком развёртки m к задаче с порядком развёртки m+1 требует также пересчёта координат каждой точки испытания на отрезке и вычисления оценки константы Гёльдера для новой целевой функции $\varphi(y_{m+1}(x))$. Из способа построения фрактального преобразования на основе кривой Пеано вытекает такое его свойство, что порядок точек испытаний на отрезке не меняется, поэтому переупорядочивание их в памяти не требуется, что существенно упрощает процедуру перехода на новый уровень детализации.

В рамках исследования данного подхода реализована версия алгоритма, которая одновременно учитывает первые два условия перехода.

Большая размерность исследуемой задачи идентификации (N=57) требует применения чисел расширенной точности, т.к. для представления числа 2^N (которое используется внутри алгоритма построения кривой Пеано) недостаточно 32-битного целого числа. Граница применимости алгоритмов глобальной оптимизации на данный момент составляет $m \cdot N < 1024$, где m обычно изменяется от 10 до 15.

Результаты параллельного решения этой задачи на одном из кластеров ННГУ (~200 процессоров) приводятся в докладе.

5. Содержательные результаты идентификации модели

Найден допустимый вариант значений параметров, при которых результаты расчетов на историческом интервале 2000-2009 гг. соответствуют статистическим данным Нижегородской области, а прогноз экономического развития дает экономически осмысленные результаты. В этом решении задачи идентификации доли теневого оборота в производственных секторах экономики области (инфраструктурном, обрабатывающем и сектора услуг) составляют 0.22, 0.28 и 0.49 соответственно.

Для каждого сектора найдены четыре параметра степенной производственной функции типа Кобба-Дугласа от четырех производственных факторов: труда, капитала, используемой в производстве продукции двух смежных секторов.

Коэффициент прироста зарплаты при нехватке труда составляет 3.0, коэффициент прироста цены при нехватке продукта 0.06, инфляционная составляющая в росте зарплаты 0.5. Описание соответствующих параметров можно найти в [2]. Штраф за отмывание денег в каждом секторе составляет 0.01 от их количества. Доля теневых денег, которая была отмыта, в каждом секторе составляет 0.1.

Доли теневой продукции инфраструктурного сектора, идущие населению, обрабатывающему сектору и сектору услуг, равны, соответственно, 0.6, 0.3, 0.1. Доли теневой продукции обрабатывающего сектора, идущие населению, инфраструктурному сектору и сектору услуг, равны 0.6, 0.2, 0.2 соответственно. Доли теневой продукции сектора

услуг, идущие населению, инфраструктурному сектору и обрабатывающему сектору, равны 0.5, 0.2, 0.3 соответственно.

Более подробное описание содержательных результатов приводится в докладе. Пробные вычислительные эксперименты с идентифицированной моделью показали работоспособность модели.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (номер проекта 11-07-97017-р_поволжье_а).

Литература

1. Оленев Н.Н., Стародубцева В.С. Исследование влияния теневого оборота на социально-экономическое положение в Республике Алтай // Региональная экономика: теория и практика. 2008. № 11 (68). С. 32–37.
2. Гергель В.П., Горбачев В.А., Оленев Н.Н., Рябов В.В., Сидоров С.В. Параллельные методы глобальной оптимизации в идентификации динамической балансовой нормативной модели региональной экономики // Вестник ЮУрГУ, №25 (242), 2011. С. 4–15.
3. Стронгин Р.Г. Поиск глобального оптимума. М.: Знание, 1990.
4. Стронгин Р.Г. Параллельная многоэкстремальная оптимизация с использованием множества разверток // Ж. вычисл. матем. и матем. физ. 1991. Т.31. №8. С. 1173–1185.
5. Strongin R.G., Sergeyev Ya.D.. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000.
6. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. Нижний Новгород: Изд-во Нижегородского гос. ун-та. – 2007. – С. 406-407.
7. Баркалов К.А., Рябов В.В., Сидоров С.В. Параллельные вычисления в задачах многоэкстремальной оптимизации // Вестник ННГУ. – 2009. – №6. – С. 171–177.

МОДУЛЬ ИСПОЛНЕНИЯ ЗАПРОСОВ ДЛЯ БАЗЫ ДАННЫХ, РАЗМЕЩЕННОЙ В ПАМЯТИ GPU

Д.А. Павлов

Казанский национальный исследовательский технический университет им. А.Н. Туполева

Вычислительные мощности видеокарт превосходят производительность центральных процессоров. Они обладают распределенной структурой с большим числом однотипных вычислительных блоков, способных работать параллельно, т.е. ядра мультипроцессора в GPU могут исполнять одни и те же инструкции одновременно.

Именно эта способность может быть использована для обработки одной процедурой сразу нескольких блоков данных (рис. 1).

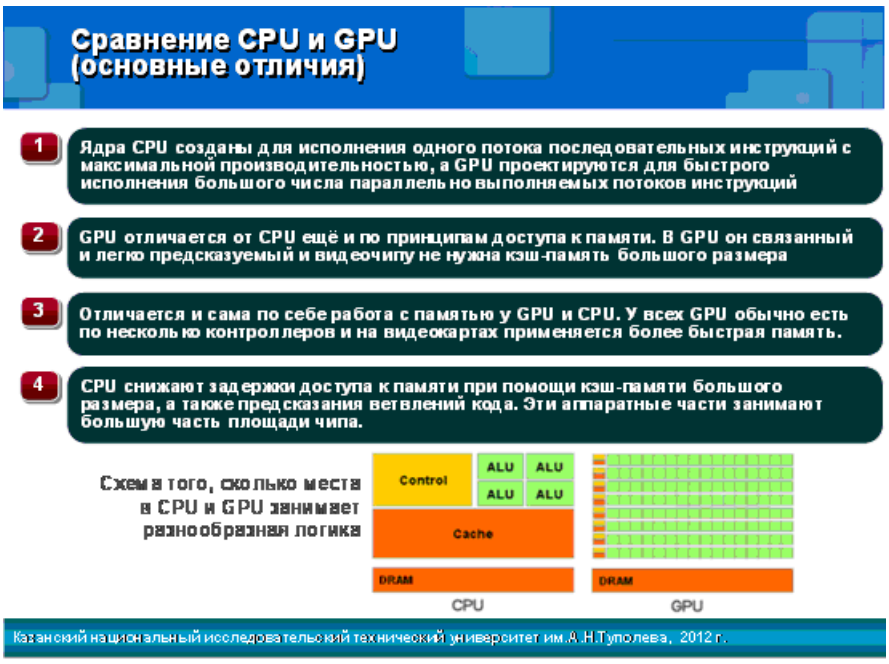


Рис. 1

Поскольку характер запросов к БД предполагает выполнение однообразных действий над большим массивом данных, представляется целесообразным разбить эти данные на отдельные блоки и обрабатывать их параллельно. И такую возможность нам дает архитектура CUDA, использующая вычислительные мощности графических процессоров (рис. 2).

Размещение же в памяти GPU самой базы данных ускорит доступ к данным и значительно упростит разработку программного модуля. Разумеется, для реализации такой схемы потребуются ресурсы целого вычислительного кластера с суммарной видеопамятью порядка десятков гигабайт. В проекте предполагается задействовать кластер КНИТУ-КАИ.

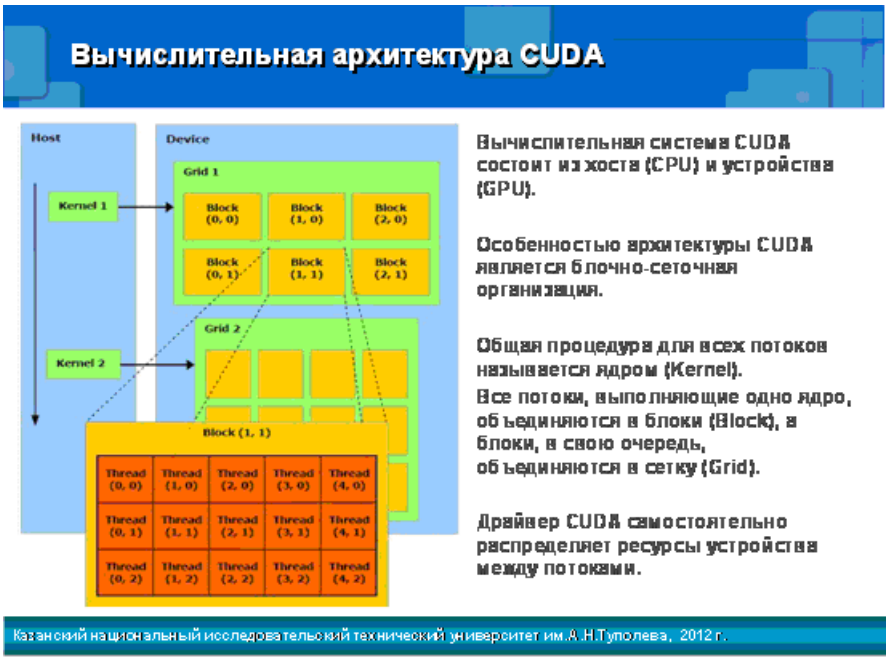

Вычислительная архитектура CUDA
Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)
Вычислительная система CUDA состоит из хоста (CPU) и устройства (GPU).
Особенностью архитектуры CUDA является блочно-сеточная организация.
Общая процедура для всех потоков называется ядром (Kernel).
Все потоки, выполняющие одно ядро, объединяются в блоки (Block), а блоки, в свою очередь, объединяются в сетку (Grid).
Драйвер CUDA самостоятельно распределяет ресурсы устройства между потоками.
Казанский национальный исследовательский технический университет им.А.Н.Туполева, 2012 г.

Рис. 2

ЛОГИКО-МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ

С.В. Панков

Южный федеральный университет, Ростов-на-Дону

Теоретические основы нейронных сетей Хопфилда [1] используют функцию *энергии*, которая определяет метод построения сети и позволяет понять её функционирование. Основное свойство энергетической функции (определённой на состояниях сети) заключается в том, что в процессе эволюции состояний сети энергия уменьшается и достигает локального минимума (*аттрактора*), при этом сеть стабилизируется в состоянии этого локального минимума. Используемый Хопфилдом формализм прояснил принцип хранения информации как динамически устойчивых аттракторов, что открыло путь к использованию рекуррентных сетей для организации ассоциативной памяти и для решения оптимизационных задач. Однако в сетях Хопфилда остаются проблемы, в частности, связанные с ограничениями на объём ассоциативной памяти, с распознаванием ложных образцов и различием похожих образцов. Актуальным также является поиск новых классов практически значимых нейронных сетей, определение их структуры, подбор параметров, понимание их функционирования.

Здесь на примере нейронных сетей Хопфилда, демонстрируется логико-математический подход к моделированию и исследованию поведения рекуррентных нейронных сетей. Этот подход базируется на формализме *L*-программ, включающим в себя логико-математическую модель параллельных вычислений – *L*-программы и логические средства анализа *L*-программ [2]. Подход направлен на решение некоторых из перечисленных проблем. Он обеспечивает компактность модели, независимость её вида от числа нейронов, что влечёт независимость анализа нейронной сети от её размера.

1. Нейронные сети Хопфилда

Сеть Хопфилда состоит из одного слоя нейронов. Число нейронов в слое обозначим через n. Число входов и выходов сети также равно n. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, на который подаётся входной сигнал сети. Выходные сигналы сети получаются на аксонах нейронов. *Состояние сети* определяется набором $\boldsymbol{S}=\langle S_1,..,S_n\rangle$, где $S_i\in\{-1,1\}$ – степень возбуждения i-го нейрона (для всех $1\le i\le n$).

Задачу реализации ассоциативной памяти сетью Хопфилда можно сформулировать следующим образом. Пусть сеть запомнила некоторый набор двоичных сигналов (представляющих изображения, звук, характеристики некоторых процессов, и т.п.), которые считаются образцами. Требуется, чтобы сеть умела по зашумленному сигналу (сигналу несущему только частичную информацию) подаваемому на её входы, находить (воспроизводить) соответствующий образец, если такой есть.

Образец двоичного сигнала представляется вектором вида $\boldsymbol{P}=\langle P_1,..,P_n\rangle$, (где $P_i\in\{-1,1\}$ для всех $1\le i\le n$). (Вектор входных сигналов $\langle \boldsymbol{x}_1,..,\boldsymbol{x}_n\rangle$ представляется аналогичным образом.) Пусть m – число образцов, тогда через $\boldsymbol{P}^k$ обозначим вектор, описывающий k-тый образец (для всех $1\le k\le m$), а через P_i^k – его i-й элемент. Если сеть успешно распознает некоторый образец $\boldsymbol{P}^k$, на основе предъявленных ей данных, её выходы

<$\mathbf{y}_1$,..,$\mathbf{y}_n$> будут содержать именно этот образец, то есть $\mathbf{y}_1=P_1^k,\ldots,\mathbf{y}_n=P_n^k$. В противном случае, выходной вектор не совпадет ни с одним образцовым.

Состояние нейронной сети $\boldsymbol{S}$ в некоторый момент времени t обозначим через $\boldsymbol{S}(t)$. Силу связи от i-го к j-му нейрону обозначим через W_{ij}. На стадии *инициализации* сети (в момент времени t=0) состояние сети $\boldsymbol{S}(0)$ определяется входными сигналами сети $S_1(0)=\boldsymbol{x}_1,\ldots,S_n(0)=\boldsymbol{x}_n$. Весовые коэффициенты синапсов устанавливаются следующим образом:

$$W_{ij} = \begin{cases} \sum_{k=1}^{m} P_i^k * P_j^k, & i \neq j, \\ 0, & i = j. \end{cases} \quad (1)$$

Заданием матрицы W осуществляется одномоментное обучение нейронной сети (запоминание образцов). Определим *срабатывание* некоторого j-го нейрона в момент времени t+1. Сначала j-й нейрон вычисляет взвешенную сумму $U_j(t+1)$ пришедших на его входы сигналов, вычисленных другими нейронами в момент времени t

$$U_j(t+1) = \sum_{i=1, i \neq j}^{n} W_{ij} * S_i(t). \quad (2)$$

Затем нейрон изменяет состояние своего возбуждения в зависимости от уровня сигнала $U_j(t+1)$ и индивидуального порога нейрона – H_j

$$S_j(t+1) = \begin{cases} -1, & \text{если } U_j(t+1) < H_j, \\ +1, & \text{если } Uj(t+1) > H_j, \\ S_j(t), & \text{если } U_j(t+1) = H_j. \end{cases} \quad (3)$$

Рассматривается два варианта динамического функционирования сети: синхронное и асинхронное. В первом случае срабатывают все нейроны сети одновременно на каждом временном шаге. Во втором случае в каждый момент времени срабатывает произвольное непустое подмножество нейронов, выбираемое случайным образом. Установлено, что любое изменение состояния сети, в результате срабатывания нейронов, приводит к уменьшению значения функции *энергии* нейронной сети – $E(\boldsymbol{S})$

$$E(\boldsymbol{S}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1, j \neq i}^{n} W_{ij} * S_i * S_j + \sum_{i=1}^{n} S_i * H_i.$$

Поверхность энергии $E(\boldsymbol{S})$ в пространстве состояний имеет сложную форму с большим количеством локальных минимумов. Сеть *завершает* свою работу в *стационарном* состоянии, в котором срабатывание любого нейрона не изменяет этого состояния. Оно также является одним из локальных минимумов функции $E(\boldsymbol{S})$, который соответствует некоторому образцу $\boldsymbol{P}^k$, в случае успешного распознавания входного вектора сигналов.

2. Моделирование нейронных сетей Хопфилда

Моделирование параллельных систем на основе L-программ предполагает разработку языка предметной области – L, для исследуемого класса систем. При этом L является проблемно-ориентированным языком логики предикатов первого порядка. Определим язык L для рассматриваемого здесь класса рекуррентных нейронных сетей Хопфилда.

2.1. Язык предметной области – *L*

Сорта:

NEURON – множество номеров нейронов {1,2,...,n};

PATTERN – множество номеров запоминаемых образцов {1,2,...,m};

INT – множество целых чисел;

BIN – множество {-1,1}.

Функции:

s: $NEURON \to BIN$ – задаёт состояние нейрона;

p: $PATTERN \times NEURON \to BIN$ – множество образцовых векторов $\{P^1,\ldots, P^m\}$, $p(j,i)$ определяет i-тый элемент j-того вектора-образца;

w: $NEURON \times NEURON \to INT$ – вычисляет элементы матрицы весовых коэффициентов связей – W_{ij} (в соответствии с формулой (1), с точностью до замены P_i^k на $p(k,i)$, P_j^k на $p(k,i)$, и с использованием арифметических операций +, *);

u: $NEURON \to INT$ – вычисляет для текущего состояния сети взвешенную сумму сигналов U_j, пришедших на входы произвольного j-того нейрона, от других нейронов (в соответствии с формулой (2), с точностью до замены W_{ij} на $w(i,j)$, $S_i(t)$ на $s(i)$, и с использованием арифметических операций +, *);

h: $NEURON \to INT$ – задаёт индивидуальные пороговые значения, используемые при активации нейрона (при его переходе в следующее состояние).

Язык L также содержит обычные арифметические функции и отношения. Знаками $\wedge$, $\vee$, $\neg$, $\to$, $\forall$ и $\exists$ будем обозначать обычные логические связки и кванторы.

Переменные:

x: $NEURON$; y:: BIN.

2.2. Формальная модель *L*-программ

Произвольная L-программа задаётся конечной совокупностью правил вида *условие* $\Rightarrow$ *действие*, где *условие* – произвольная L-формула, *действие* – L-формула специального вида. Она определяет систему переходов $S''=(Q,T)$, где Q – множество L-структур (т.е. алгебраическая система, определяемая носителем и интерпретацией функциональных и предикатных символов языка L на этом носителе), а $T \subseteq Q \times Q$ – отношение перехода (или отношение непосредственной достижимости). Каждая L-структур q из Q описывает некоторое состояние моделируемой системы, в данном случае она будет представлять состояние нейронной сети.

Переход из состояния q в другое состояние заключается в переопределении функций и отношений L-структуры q посредством асинхронного исполнения действий правил. При этом условия правил определяют, какое переопределение возможно. Такая организация L-программ привносит в логику первого порядка динамизм, необходимый для описания поведения параллельных вычислений. L-программа завершает свою работу в L-структуре, на которой становятся ложными условия всех правил. Неподвижной точкой L-программы называется такая L-структура q, из которой если и возможен переход, то только в саму себя. Множество всех неподвижных точек L-программы обозначим через $fin[S']$. Так локальные минимумы энергетической функции $E(\boldsymbol{S})$ будут соответствовать неподвижным точкам L-программы, моделирующей нейронную сеть Хопфилда.

2.3. Моделирующие *L*-программы

Начальное состояние удовлетворяет следующему ниже предусловию ***In***, которое утверждает, что состояние сети определено произвольным двоичным вектором входных сигналов с элементами -1 и 1

In: $\forall x(\exists y\ s(x)=y)$

Синхронная L-программа:

$\$\ x,y\quad (\ u(x)<h(x) \wedge y=-1 \vee$

$\quad u(x)>h(x) \wedge y=1 \vee$

$\quad u(x)=h(x) \wedge y=s(x)\ ) \Rightarrow s(x)=y$

Эта L-программа состоит из одного правила, моделирующего срабатывание нейрона x и изменение его состояния, в соответствии с формулой (3). Благодаря стоящему

перед правилом синхронизатору $ x,y, это правило моделирует одновременное срабатывание всех нейронов. Оно исполняется для всех наборов значений переменных x и y, для которых истинно условие правила.

Асинхронная L-программа:

$$(u(x)<h(x)\wedge y= -1 \vee$$
$$u(x)>h(x)\wedge y=1 \vee$$
$$u(x)=h(x)\wedge y=s(x)) \Rightarrow s(x)=y$$

Отличие этой L-программы от предыдущей состоит только в том, что в ней отсутствует синхронизатор. В этом случае правило моделирует срабатывание произвольного непустого множества нейронов. Оно исполняется для произвольного непустого множества наборов значений переменных x и y, для которых истинно условие правила. Отметим, что вид синхронной и асинхронной модели не зависит от числа нейронов в сети.

3. Анализ нейронных сетей

Пусть $In\subseteq Q$ – множество начальных состояний (допустимых векторов входных сигналов), а $Out\subseteq Q$ – это множество состояний, соответствующих образцовым векторам. Требуется установить, что для каждого набора входных сигналов из In будет достижима в системе переходов S' неподвижная точка из Out (образцовое состояние). Установить это свойство можно путём проверки выполнения следующих достаточных условий: Существует такое множество состояний $INV\subseteq Q$, что справедливо:

1) $In\subseteq INV$;
2) $\forall q',q\in Q\ ((q',q)\in T\ \&\ q'\in INV)\Rightarrow q\in INV)$. т.е. INV – инвариант;
3) $INV\cap fin[S']\subseteq Out$.

Алгоритмы из [2] позволяют получить логические формулы, выражающие в языке L, отношение перехода T, множество $fin[S']$ для моделирующей L-программы. Если множества In, INV и Out также выражаются L-формулами, то анализ такого свойства моделируемой нейронной сети сводится к доказательству истинности формул логики предикатов первого порядка. Здесь приведём только L-форму FIN, выражающую множество неподвижных точек $fin[S']$

$$FIN\colon \forall x,y\ ((u(x)<h(x)\wedge y= -1 \vee$$
$$u(x)>h(x)\wedge y=1 \vee$$
$$u(x)=h(x)\wedge y=s(x)) \rightarrow s(x)=y).$$

Формулы, выражающие множество неподвижных точек для синхронной и асинхронной моделирующей L-программы, будут совпадать, а формулы для отношения перехода T будут различны. Формулы, выражающие отношения перехода T, могут быть использованы, в частности, для обоснования невозрастания энергетической функции $E(S)$ на множестве достижимых состояний сети.

Для систем с регулярными схемами взаимодействия (как в случае нейронной сети Хопфилда) удаётся находить компактные формулы – инварианты, независящие от числа её элементов (в данном случае нейронов). Такой эффект, в частности, демонстрировался в [3]. Формулы, выражающие отношение перехода, множество неподвижных точек (и другие) также не зависят от количества элементов, что обеспечивает независимость анализа от размера системы.

Литература

1. Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities // Proceedings of National Academy of Sciences. 1982. V. 79. No. 8. P. 2554–2558.

2. Крицкий С.П., Панков С.В. О верификации асинхронных программ продукционного типа // Программирование. 1994. № 5. С. 40–52.

3. Панков С.В. Логический подход к спецификации и анализу поведения клеточных автоматов // Материалы Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ, многоядерный компьютерный мир». Новороссийск: Изд-во МГУ, 2007. С. 85–88.

ПАРАЛЛЕЛЬНОЕ РЕШЕНИЕ ЗАДАЧИ КЛАССИФИКАЦИИ ЛИЦ МЕТОДОМ ГОЛОСОВАНИЯ С НЕЙРОСЕТЕВЫМ АРБИТРОМ

В.В. Пекунов

ООО «Спецлаборатория», Иваново
E-mail: pekunov@mail.ru

Задача классификации лиц, то есть идентификация личности по ее изображению, полученному, в частности, с камеры видеонаблюдения, весьма актуальна для современных систем безопасности. Среди методов идентификации выделяются: а) методы, основанные на выделении и сравнении характерных черт лица (метод эластичных графов и другие [6]), б) методы, основанные на редукции изображения всего лица к вектору проекций данного изображения на новый специфический базис (метод главных компонентов [3, 8] с последующим привлечением нейронных сетей, линейного дискриминантного анализа [3] и/или иных методов), в) методы, предполагающие какой-либо иной подход (скрытые марковские модели [2], дискретные преобразования Фурье) к анализу отдельных фрагментов изображения лица и их комплексов. Первый класс методов весьма чувствителен к точности определения координат черт лица и иногда требует наличия цветовой компоненты изображения [4], что в общем случае, когда изображение снимается с простых камер видеонаблюдения, часто черно-белых и не дающих качественного разрешения, затруднительно. Поэтому в сфере нашего особого внимания оказался второй класс методов, в основе которого лежит надежный и хорошо формализованный метод главных компонентов [8].

Задача данной работы состояла в том, чтобы построить надежный и быстродействующий классификатор лиц, представленных вектором координат в пространстве главных компонентов. Было выдвинуто требование работы всех компонентов системы в реальном масштабе времени, что определило потребность в максимально возможном применении технологий распараллеливания вычислений при решении данной задачи на отдельно взятом компьютере. Учитывая, что ресурсы центрального процессора обычно достаточно интенсивно задействованы под задачи видеонаблюдения и аналитики, весьма актуальным представляется использование вычислительных мощностей многоядерных видеокарт, многообразие которых закономерно приводит нас к идее применения кроссплатформенной технологии OpenCL.

Поступающие с камеры изображения лиц автоматически центрировались в результате работы детектора лиц (двухэтапной детекции с фильтром LBP (Local Binary Patterns, [7]) на первом этапе и отсеиванием результатов на втором этапе фильтром с примитивами Хаара [9]). Далее отсекался фон, для этого на изображении подбирался наиболее адекватный эллипс, выделяющий исключительно лицо, опираясь на его граничные точки, определенные детектором Кенни [1]. Подбор эллипса осуществлялся методом Хафа [1]. После отсечения фона выполнялась эквализация изображения [1], позволяющая девальвировать различия в условиях освещенности.

После предварительной обработки к изображению лица применялся метод главных компонент, изображение проецировалось на базис из 45 собственных лиц, сгенерированных по базе BioID. Полученная метрика лица (вектор координат в пространстве главных компонентов) подлежала идентификации по базе данных метрик лиц персон. Сразу же было принято решение отказаться от чисто нейросетевой [5] идентификации с одной сетью-классификатором по причине ее высокой сложности и, следовательно,

чрезмерно высоких затрат на обучение и переобучение сети (при вводе в базу новых лиц), хотя и известно, что такой подход иногда дает весьма высокую вероятность распознавания. Однако справедливости ради следует отметить, что такой результат достигается при идентификации тех изображений лиц, которые непосредственно входят в обучающую выборку. При попытке идентификации лиц в несколько иных ракурсах (сдвинутых, наклоненных, повернутых) процент распознавания обычно уменьшается. Также мы отказались от одного из классических методов – линейного дискриминанта Фишера [3], поскольку сразу же оказалось, что ковариационная матрица часто не имеет полного ранга. Возможно помог бы отбор переменных (координат пространства главных компонентов) с применением корреляционного анализа, но было принято более простое решение – использовать для классификации линейный персептрон.

В качестве возможных нейросетевых вариантов рассматривались лишь: а) *линейный персептрон*[1], б) *классическая двуслойная нейронная сеть прямого распространения*[2] с первым нелинейным (экспоненциальные сигмоиды) слоем из 20÷32 нейронов и одним линейным нейроном во втором слое, обучение которой ведется градиентным методом (обратного распространения ошибки). Значительные надежды возлагались на *метод группового учета аргументов* (МГУА[3], см. [3]) с простейшими квадратичными функциями-блоками, который позволяет инкрементально быстро строить не менее сложные и перспективные интерполяторы, чем нейронные сети. Кроме того, рассматривались такие элементарные методы как *наибольшей близости* [8] и *k-ближайших соседей* [3], базирующиеся на расстоянии Махалонобиса.

Было принято решение провести сравнительное экспериментальное исследование всех указанных методов. В трех первых случаях для каждой персоны строился отдельный интерполятор (сеть полиномов МГУА или нейронная сеть), который обучался выделять метрики лиц данной персоны среди других. Значение выхода интерполятора принадлежало диапазону [0; 1], величина значения определяла степень уверенности идентификации данной персоны. В таком случае при появлении в базе данных нового лица иногда можно оперативно переопределить (переобучить) требуемое количество интерполяторов за время, допустимое по требованиям решаемых нами задач.

Методы рассматривались на базе данных из 180 изображений лиц (28 персон в различных ракурсах). Задача была приближена к реальности: системе предъявлялись изображения персон в тех же ракурсах, что и в базе, а также и в несколько иных.

Таблица 1. Проценты верного распознавания персон различными методами

Метод				
Наибольшей близости	k-ближайших соседей	Линейные персептроны	Двуслойные сети	МГУА
1. Изображения лиц в произвольных ракурсах				
77,5%	72,5%	67,5%	–	60%
2. Изображения лиц с принудительным варьированием ракурса				
83%	76,6%	55,3%	57,8%	55,3%

[1] Количество входов n равняется размеру вектора метрики лица. Сеть, в данном случае состоящая из одного нейрона со смещением, обучалась методом наименьших квадратов, сводящемуся к решению системы из n+1 линейных алгебраических уравнений.

[2] Количество входов равняется размеру вектора метрики лица. Все нейроны имели смещение.

[3] В первом каскаде элементарных интерполяторов МГУА обычно осуществляется выбор из n^2 элементарных классификаторов, где n – число входов. В нашем случае для повышения скорости работы производился выбор из m^2 классификаторов, $m < n$, где m — количество наиболее «перспективных» входов, отобранных в ходе корреляционного анализа.

Примечательна *разница в качестве распознавания в первом и втором случаях*. Вероятно, именно лица в «обычных», наиболее часто встречающихся ракурсах, в предложенных условиях распознавались первыми двумя методами несколько хуже, чем лица в «существенно отклоняющихся» ракурсах. Для прочих методов ситуация обратная.

Лидируют два простых метода – наибольшей близости и *k*-ближайших соседей. Из трех интерполяционных методов наихудшие результаты показал МГУА, видимо потому, что полиномы дали достаточно маловероятное приближение, которое в сочетании с завышенной степенью таких полиномов привело к возникновению множества очагов неверной идентификации. Нейросетевые методы дали более правдоподобное приближение, но, видимо, проявился хорошо известный эффект [5] скорее чрезмерного подстраивания под элементы обучающей выборки чем выделения общей закономерности. Таким образом, для дальнейшей работы в качестве основных были выбраны оба вышеупомянутых простых метода в сочетании с принципом голосования. Персона считается идентифицированной, если за нее подали голос оба метода. Особенно же интересен случай «полемики» методов, когда они рекомендуют разные персоны.

При анализе экспериментальных данных по «полемике» было обнаружено, что помимо случаев, в которых оба простых метода ошибаются, есть ряд случаев, где хотя бы один из них дает верный ответ. Это позволило бы повысить точность предсказания, если бы удалось привлечь некоторый алгоритм-арбитр, который выберет верный ответ.

Достаточно сложной задачей оказался выбор арбитра. Было сделано предположение, что оказавшиеся более правдоподобными в сравнении с МГУА нейросетевые методы могут, если не идентифицировать одну персону среди множества, то, по меньшей мере, верно выбрать персону среди двух, предложенных методами наибольшей близости и *k*-ближайших соседей.

Были проведены дополнительные исследования, целью которых являлось определение наиболее перспективного метода-арбитра в случае «полемики» основных простых методов. Обнаружилось, что линейный персептрон дает верный ответ в 50% таких случаев, а двуслойная сеть – в 80% случаев. Таким образом, *предлагаемый в настоящей работе метод* идентификации является двухэтапным. На первом этапе проводится идентификация двумя первичными методами: а) поиск ближайшей (по Махалонобису) метрики в пространстве собственных лиц с рекомендацией соответствующей персоны; б) поиск *k*-ближайших метрик в том же пространстве с рекомендацией персоны, которой соответствует наибольшее количество этих метрик. Если рекомендации методов совпадают, то идентификация завершена. В противном случае начинается второй этап, на котором включается арбитр (либо линейные персептроны, либо двуслойные сети), выбирающий персону среди двух рекомендованных. Выбирается персона, для которой арбитр дает большее значение.

Требование быстродействия определило следующий алгоритм работы арбитра:

а) если присутствуют обученные двуслойные нейронные сети для каждого из кандидатов, то выбирается персона, сеть которой дает лучший прогноз;

б) если обученных сетей нет, то либо они оперативно обучаются (если ресурсы вычислительной системы позволяют выполнить обучение за доли секунды) и работают в соответствии с пунктом а), либо задействуются линейные персептроны (если обученные персептроны отсутствуют, то они оперативно обучаются, это весьма быстрый процесс на современных ЭВМ).

В заключение рассмотрим подробно вопрос о максимально возможном повышении скорости обучения двуслойной нейронной сети. Методы, основанные на неполном переборе (например, случайного поиска) были отвергнуты по причине весьма значительных временных затрат на переобучение даже с применением параллельных вычислений. Метод обратного распространения ошибки (как и любой иной градиентный метод)

подразумевает *последовательную* (*даже в случае какого-либо распараллеливания счета*) *подстройку коэффициентов для каждого элемента обучающей выборки размера k*. Поэтому время обучения включает существенный компонент с прямой пропорциональностью k. Очевидно, что данный факт приводит к существованию разумного ограничения k_{max} на значение данной величины.

Предлагается включать в обучающую выборку лишь некоторые, наиболее характерные элементы. В нашем случае это можно сделать особенно просто, учитывая, что нейронная сеть является арбитром двух методов. *В обучающую выборку двуслойной сети*, сопоставленной каждой персоне, должны входить: а) лица, соответствующие данной персоне; б) лица, соответствующие «конкурирующим» персонам, то есть таким, с которыми у данной наиболее часто возникает «полемика» методов наибольшей близости и k-ближайших соседей; в) лица, соответствующие некоторому количеству случайным образом выбранных персон, что должно несколько повысить потенциальную надежность арбитража.

Для ускорения расчета было принято решение задействовать локальные вычислительные ресурсы многоядерных видеокарт. Этого должно было оказаться более чем достаточно для одновременного обучения двух и более нейронных сетей. Чтобы исключить привязку к конкретному производителю видеокарты, для распараллеливания был выбран интерфейс OpenCL.

Был выбран следующий *принцип распараллеливания*: каждая сеть отдавалась одному многопоточному мультипроцессору карты, каждый нейрон первого слоя сети обрабатывался отдельным потоком. Единственный нейрон второго, выходного слоя обрабатывался в каждом из этих потоков. Такая дублирующая схема оказалась наименее затратной, поскольку практически не требует координации потоков. При расчетах максимально возможно задействовалась внутренняя регистровая память потоков, там же, где это было невозможно, применялась разделяемая (shared в терминологии производителя видеокарт, она же local в терминологии OpenCL) память мультипроцессоров. При нехватке разделяемой памяти использовалась глобальная (преимущественно для передачи на видеокарту обучающей выборки).

Проводилось *сравнение результатов*, полученных с применением видеокарты, с результатами, полученными на классическом процессоре. Прямое сравнение временных характеристик обычной однопроцессорной программы и OpenCL-программы дало результат, на первый взгляд производящий впечатление абсурдного: расчет на 64 потоковых процессорах nVidia GeForce GTX570 оказался в 217 раз быстрее, чем на одном ядре процессора Intel Core i5-2300 (2,8 ГГц). Однако, если учесть, что однопроцессорная программа не была оптимизирована (Debug-версия компилятора C++ среды Microsoft Visual Studio) и содержала значительное количество операций индексного доступа к элементам массивов, от которых была свободна программа для видеокарты (за счет перехода от индекса массива к номеру потока), а также OpenCL-программа работала преимущественно с быстродействующей регистровой памятью, становится ясно, что вышеуказанный результат возможен.

Тем не менее, для контроля было принято решение о *дополнительном сравнении* с результатами, полученными той же OpenCL-программой, но запущенной уже не на видеокарте, а на вышеуказанном многоядерном центральном процессоре (ЦП) Intel с применением среды Intel OpenCL SDK 1.5. В данном случае расчет на видеокарте занял в 2,1 раза меньше времени чем расчет на четырех ядрах ЦП, то есть примерно в 2,1×4 = 8,4 раза меньше, чем расчет на одном ядре ЦП. Данную величину (8,4) можно было бы обозначить как «физическое ускорение» P_{opt}, помимо которого нас интересует также «логическое ускорение» S_{opt} по отношению к одному ядру ЦП, определяемое масштабированием результата с учетом разницы тактовых частот. Тактовая частота f_{cpu}

используемого ЦП (2,8 ГГц) в два раза выше тактовой частоты f_{gpu} потоковых (шейдерных) процессоров (1,4 ГГц) видеокарты. Тогда «логическое ускорение» составит

$$S_{opt} = P_{opt} \frac{f_{cpu}}{f_{gpu}} = 8,4 \frac{2,8}{1,4} = 16,8.$$

Эта величина приблизительно в 4 раза меньше ожидаемой (64-х, поскольку было задействовано именно такое количество потоковых процессоров). Вероятно, такой дополнительный множитель объясняется меньшей эффективностью исполнения серии команд потоковыми процессорами, то есть меньшим количеством команд, исполняемым за такт, в сравнении с ЦП Intel, что связано, в частности, с применением векторных SSE-команд ЦП.

По той же формуле легко подсчитать «логическое ускорение» S_{nopt} по сравнению с расчетом неоптимизированной программой, работающей на одном ядре ЦП:

$$S_{nopt} = 217 \frac{2,8}{1,4} = 434.$$

Резюмируя, в данной работе предлагается новый метод идентификации лиц, включающий опрос методов наибольшей близости и *k*-ближайших соседей с последующим использованием нейронной сети (двуслойной или линейной однослойной) в качестве арбитра в случае «полемики» методов. Показано, что применение арбитра позволяет повысить точность прогнозирования до 87÷88% при сохранении высокой скорости идентификации. Это достаточно хороший показатель, особенно с учетом того, что идентифицирующей системе предъявлялись изображения лиц не только из базы данных, но схожие с ними, в том числе повернутые и наклоненные. При обучении двуслойных нейронных сетей применен перенос вычислений (с распараллеливанием) на видеокарту. На 64 потоковых процессорах nVidia GeForce GTX570 получено существенное логическое ускорение в 16,8 раз (физическое – в 8,4 раза) *по сравнению с оптимизированным расчетом* на одном ядре процессора Intel Core i5-2300 (2,8 ГГц) и в 434 раза (физическое – в 217 раз) *по сравнению с неоптимизированным расчетом* на том же процессоре.

Литература

1. Визильтер Ю.В. Обработка и анализ цифровых изображений с примерами на LabVIEW и IMAQ Vision / Ю.В. Визильтер, С.Ю. Желтов, В.А. Князь, А.Н. Ходарев, А.В.Моржин. – М.: Книга по требованию, 2011. – 464 с.
2. Гультяева Т.А., Попов А.А. Скрытые марковские модели с одномерной топологией в задаче распознавания лиц // Сборник научных трудов НГТУ. – 2006. – № 1. – С. 1–6.
3. Дюк В., Самойленко А. Data mining: учебный курс – СПб: Питер, 2001. – 368 с.
4. Лопатина А.Д. Выделение области лица с помощью комбинации методов цветовой и яркостной сегментации // Вестник УГАТУ. Управление, ВТ и И. – 2009. – Т.13, №2 (35). – С.180–187.
5. Рассел С., Норвиг П. Искусственный интеллект: современный подход – М.: «Вильямс», 2007. – 1408 с.
6. Самаль Д.И. Алгоритмы идентификации человека по фотопортрету на основе геометрических преобразований: автореф. дис. ... канд. тех. наук – Минск, 2002. – 20 с.
7. Ahonen T., Hadid A., Pietikäinen M. Face recognition with local binary patterns // Proc. Eighth European Conf. Computer Vision – Springer, 2004. – P. 469–481.
8. Turk M., Pentland A. Eigenfaces for recognition // Journal of Cognitive Neuroscience. – 1991. – Vol. 3. – №1. – P. 71–86.
9. Viola P.A., Jones M.J. Rapid Object Detection using a Boosted Cascade of Simple Features // Computer Vision Pattern Recognition. 2001. (1). P. 511–518.

ПЛАНИРОВАНИЕ ЗАДАЧ ДЛЯ ОБЛАЧНЫХ ВЫЧИСЛИТЕЛЬНЫХ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ

П.Н. Полежаев, Ю.А. Ушаков, А.Е. Шухман

Оренбургский госуниверситет

На концептуальном уровне описывается подход к планированию вычислений в облачных центрах обработки данных с использованием программно-конфигурируемых сетей. Предлагается вариант развития облачной управляющей системы OpenStack за счет эффективного планирования задач (групп виртуальных машин), путем их распределения по топологически-близким вычислительным узлам, а также за счет управления потоками данных между ними с целью их топологической локализации и снижения сетевой конкуренции.

Введение

Облачные вычислительные системы стали стандартом де-факто для многих прогрессивных областей сферы информационных технологий. Российские и иностранные компании используют их с целью размещения своих научных и бизнес-приложений, что позволяет им избежать расходов на создание и поддержание функционирования собственных центров обработки данных (ЦОД). С другой стороны, владельцы облачных вычислительных ЦОД путем консолидации вычислительных ресурсов и систем хранения данных (СХД) способны снизить совокупную стоимость владения ИТ-инфраструктурой за счет обслуживания значительного количества клиентов, а также использования эффективных технических средств планирования и балансировки нагрузки, управления пересылкой данных в сети, интеграции нескольких территориально разрозненных сегментов ЦОД.

Существующие сетевые протоколы облачных вычислительных ЦОД Fiber Channel, Infiniband, а также традиционный Ethernet, имеют ограниченные возможности по управлению трафиком и QoS. Усовершенствованные варианты протокола Ethernet – Converged Enhanced Ethernet и Cisco Data Center Ethernet включают расширения по управлению потоками на основе приоритетов, разделению пропускной способности, управлению перегрузками и логическим состоянием полос передачи данных, по обеспечению передачи без потерь, а также по одновременному использованию нескольких параллельных путей передачи данных между узлами. Основные недостатки данных решений – сложная децентрализованная схема управления потоками данных, основанная на множестве закрытых протоколов, значительная стоимость сетевого оборудования, сложность его расширения. Отдельно следует заметить, что данные решения используют реактивную схему управления потоками, когда решения по коммутации принимаются во время передачи потока.

Весьма перспективным направлением технологий вычислительных ЦОД является использование программно-конфигурируемых сетей (ПКС – Software Defined Network). Принципы ПКС впервые возникли в исследовательских лабораториях Стэнфорда и Беркли [1], и в настоящее время развиваются в рамках консорциума Open Network Foundation и европейского проекта OFELIA [2]. Известен положительный опыт компаний Google и Amazon по внедрению ПКС в облачных ЦОД.

В основе подхода ПКС лежит возможность динамически управлять пересылкой данных в сети с помощью открытого протокола OpenFlow. Все активные сетевые устройства объединяются под управлением сетевой операционной системы (СОС), которая обеспечивает приложениям доступ к управлению сетью. Сетевые контроллеры могут быть централизованными, использовать общие абстракции для пересылки пакетов.

За счет управления пересылкой данных в сети использование ПКС в облачных ЦОД позволяет реализовать схемы одновременной многопутевой передачи данных, управление потоками на основе приоритетов, виртуализацию сети, обеспечить QoS, эффективно распределить нагрузку на сеть. Открытость стандарта OpenFlow и вынесение логики управления в отдельный контроллер упрощает программное и аппаратное обеспечение активного сетевого оборудования, что позволит в будущем производителям снизить его стоимость, а следовательно уменьшатся затраты на создание ЦОД. Централизация и открытость средств управления ПКС позволяет гибко и эффективно адаптировать работу ЦОД под возникающие потребности бизнеса, что ускоряет внедрение инноваций и обеспечивает конкурентоспособность компаний.

В настоящее время отсутствуют решения на основе ПКС для облачных вычислительных ЦОД, включающих СХД, направленных на управление потоками данных задач с целью их топологической локализации и снижения сетевой конкуренции между ними, что позволит повысить эффективность работы облачного вычислительного ЦОД. Это определяет новизну настоящей работы.

Наличие потребности компаний в создании эффективных облачных вычислительных ЦОД, включающих СХД, на основе ПКС, перспективный характер технологии ПКС, а также отсутствие готовых решений в этой области являются существенными факторами, определяющими актуальность проведения данных исследований.

1. Недостатки управляющей системы OpenStack

Наиболее распространенным открытым программным обеспечением для управления облачными вычислительными ЦОД является OpenStack [3]. Его популярность, прежде всего, обусловлена открытостью, документированностью, наличием исходных кодов, а, следовательно, возможностью адаптации его компонент под произвольные архитектуры вычислительных систем, в отличие, например, от узко ориентированных закрытых решений VMware.

OpenStack позволяет создавать многоарендные архитектуры IaaS облачных вычислительных ЦОД, когда вычислительные и сетевые ресурсы разделяются между несколькими пользователями-арендаторами. Каждому арендатору выделяется виртуализованные ресурсы в виде группы виртуальных машин, связанных виртуальной сетью. Пользователь вправе сам настраивать конфигурацию виртуальных машин, включая параметры процессора, памяти, загружаемый образ с операционной системой, программами и данными. Хранение образов виртуальных машин обеспечивается компонентой Glance, а данных – облачным хранилищем Swift.

OpenStack был выбран в качестве основы для построения собственной управляющей системы облачного вычислительного ЦОД с использованием ПКС. Его наиболее существенным недостатком, требующим доработки, является алгоритм планирования задач (групп виртуальных машин).

В существующем реализованном варианте данный алгоритм подбирает подходящие по конфигурации физические узлы для размещения виртуальных машин, руководствуясь их интегральными характеристиками. Интегральная характеристика узла представляет собой линейную свертку его динамических характеристик (host capabilities), взятых с некоторыми весовыми коэффициентами. Алгоритм планирования ранжирует

узлы в порядке убывания значения интегральной характеристики и выбирает запрашиваемое число первых узлов. Данный процесс в терминологии OpenStack носит название “Host weighting”. Очевидным недостатком данного подхода является недо-груженность вычислительных ресурсов вследствие фрагментации расписания из-за нерационального распределения виртуальных машин в соответствии с их запросами к физическим ресурсам.

Еще одним недостатком OpenStack является отсутствие планирования сетевого трафика между виртуальными машинами. Действительно, когда известно расположение виртуальных машин пользователя, необходима локализация сетевого трафика между ними, направленная на снижения сетевой конкуренции. Конкуренция возникает из-за очередей пакетов в коммутаторах, отправляемых виртуальными машинами различных групп.

Другой недостаток OpenStack – отсутствие гарантий по пропускной способности виртуальной сети, соединяющей виртуальные машины. Нет возможности задания минимально-гарантированной пропускной способности для каждой сетевой связи, которая бы обеспечивалась сетевой частью облачного вычислительного ЦОД.

Авторы планируют устранить данные недостатки за счет использования адаптированных вариантов ранее разработанных алгоритмов планирования задач для кластерных вычислительных систем [4], ПКС на основе протокола OpenFlow, а также сетевых средств обеспечения QoS.

2. Суть предлагаемого решения по развитию OpenStack

На рис. 1 приведена концептуальная схема планирования задач, предлагаемая в рамках настоящей работы. Серым цветом обозначены элементы, разрабатываемые авторами.

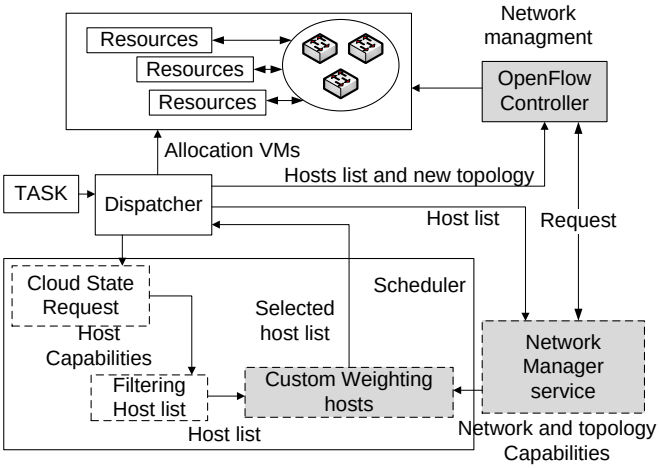


Рис. 1. Концептуальная схема планирования в улучшенном варианте системы OpenStack

Опишем основные элементы схемы.

1. Вычислительные узлы, связанные с помощью коммутаторов с поддержкой OpenFlow. Использование данных коммутаторов позволяет управлять потоками данных групп виртуальных машин с целью их топологической локализации и снижения сетевой конкуренции между ними.
2. Dispatcher представляет собой диспетчер облачной вычислительной системы, решающий задачу управления и планирования виртуальными машинами.
3. OpenFlow Controller – контроллер протокола OpenFlow, представляющий собой центральную компоненту ПКС. Сосредотачивает в себе всю программную логику управления маршрутизацией пакетов. Используется для формирования таблиц

коммутации в OpenFlow-коммутаторах на основе рассчитанных потоков данных между назначенными на физические узлы виртуальными машинами планируемых групп. Также используется для сбора сетевой статистики с коммутаторов OpenFlow и передачи ее элементу Network Manager Service.

4. Network Manager Service – служба, реализующая механизмы сбора статической и динамической информации об облачном вычислительном ЦОД. Включает обнаружение топологии сети на основе протокола LLDP, получение статистики от контроллера OpenFlow, сбор информации о загрузке вычислительных ядер и памяти узлов с помощью SNMP.
5. Scheduler – планировщик задач, который на основе запроса пользователя на выделение ресурсов для группы виртуальных машин, а также сведений о состоянии облачной вычислительной системы, подбирает оптимальные вычислительные узлы (hosts) для запуска виртуальных машин.

Более подробно схема работы планировщика изображена на рис. 2. Предполагается выполнение следующих шагов в процессе планирования:

1. Получение сведений о состоянии вычислительных узлов. В этом шаге отличие от стандартного планировщика OpenStack в том, что используется полный список всех узлов облачной вычислительной системы.
2. Получение сведений о состоянии сети, включая ее статические и динамические характеристики.
3. Объединение всей полученной информации в виде взвешенного неориентированного графа системы. Его вершинами являются узлы и коммутаторы, ребрами – сетевые связи между ними. Вся полученная информация описывается в виде весов вершин и ребер.
4. Выбор оптимального подграфа. На данном шаге предлагается использовать модифицированные варианты ранее разработанных авторами алгоритмов планирования Backfill MDM и Backfill SDM, которые учитывают топологию вычислительной системы, а также сетевую конкуренцию. Более подробно о них можно прочитать в работе [4].
5. Для совместимости с существующим кодом диспетчера OpenStack выбранные узлы записываются в список, в котором каждому элементу приписывается фиктивный интегральный вес.

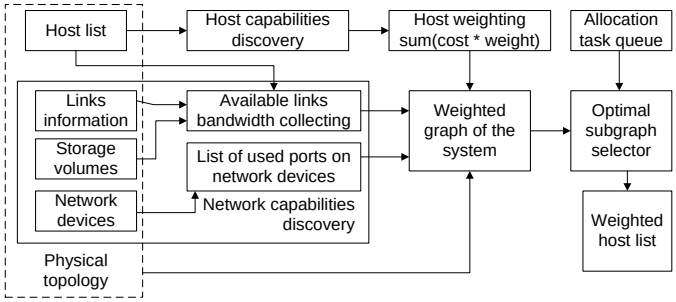


Рис. 2. Концептуальная схема работы планировщика

Также будет частично дописана компонента Dispatcher системы OpenStack, в ней планируется реализовать передачу сведений контроллеру OpenFlow о назначаемой группе виртуальных машин, чтобы тот предварительно (до запуска виртуальных машин на узлах) установил правила коммутации в таблицы соответствующих коммутаторов. Проактивный (предварительный) характер задания правил позволит исключить задержку на установление записей по каждому потоку, которая имела бы место при реактивной схеме (когда коммутатор при получении первого пакета потока в случае отсутствия правил передает его на анализ контроллеру).

Если на базе группы виртуальных машин разворачивается вычислительный кластер (HPC on cloud), то топологически-близкое размещение виртуальных машин, а также использование эффективной маршрутизации, позволит снизить время выполнения коммуникационно-интенсивных параллельных задач, решаемых на данном кластере. Тот же результат будет иметь место (снижение времени вычислений, снижение времени отклика) для прочих вычислительных приложений, выполняющихся на виртуальных машинах.

Заключение

На концептуальном уровне описан подход к планированию вычислений в облачных центрах обработки данных с использованием программно-конфигурируемых сетей. Предложен вариант развития облачной управляющей системы OpenStack за счет эффективного планирования задач (групп виртуальных машин), путем их распределения по топологически-близким вычислительным узлам, а также за счет управления потоками данных между ними с целью их топологической локализации и снижения сетевой конкуренции. В ближайшем будущем планируется реализация данного подхода в рамках облачного вычислительного ЦОД, создаваемого в Оренбургском государственном университете.

Исследования поддержаны федеральной целевой программой «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы» (госконтракт №07.514.11.4153) и РФФИ (проект №12-07-31089).

Литература

1. Смелянский Р.Л. Проблемы современных компьютерных сетей // Труды XIX Всероссийской научно-методической конференции «Телематика 2012». – СПб: СПБГУ ИТМО, 2012. – Т. 1. – С. 371–375.
2. OFELIA: OpenFlow in Europe –.[http://www.fp7-ofelia.eu/].
3. OpenStack Open Source Cloud Computing Software – [http://www.openstack.org/].
4. Полежаев П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ'2010): Труды международной конференции. – Челябинск: ЮУрГУ, 2010. – С. 287–298.

МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ С БОЛЬШИМ ЧИСЛОМ КАТЕГОРИЙ

А.Н. Половинкин

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача классификации изображений с большим числом категорий объектов. Предлагаются основанные на деревьях решений и их ансамблях модификации алгоритмов классификации, учитывающие семантическую иерархию объектов в базе данных WordNet. Приводятся результаты вычислительного эксперимента.

Введение

Важное прикладное значение имеют задачи классификации изображений с большим числом категорий объектов (проблемы подобного рода возникают в различных областях: например, поиск изображений по ключевым словам, корректное распознавание объектов для робототехники и т.п.). До последнего времени алгоритмы для решения данной задачи мало исследовались, в том числе и ввиду вычислительной сложности. Впервые проведенные вычислительные эксперименты подобного рода на базах изображений Caltech [3] и ImageNet [6] показали, что применение общепризнанных на настоящее время алгоритмов не дает удовлетворительной точности распознавания. В работах [4, 5] выдвинута и экспериментально апробирована идея о влиянии на качество алгоритмов классификации того, насколько эти алгоритмы принимают во внимание семантические зависимости между рассматриваемыми категориями. В данной работе рассматриваются основанные на деревьях решений и их ансамблях модификации алгоритмов классификации, учитывающие семантическую иерархию объектов в базе данных WordNet.

1. Алгоритм обучения с учителем «случайный лес»

Случайный лес [1] является одним из наиболее универсальных и эффективных алгоритмов обучения с учителем, применимым как для задач классификации, так и для задач восстановления регрессии. Идея метода заключается в построении ансамбля параллельно обучаемых независимых деревьев решений [2]. Итоговая классификация объекта выполняется путем голосования всех деревьев, входящих в состав ансамбля.

Среди достоинств алгоритма можно выделить высокое качество предсказания, способность эффективно обрабатывать данные с большим числом классов и признаков, внутреннюю оценку обобщающей способности модели, высокую распараллеливаемость и масштабируемость. Также метод обладает всеми преимуществами деревьев решений, в том числе отсутствием необходимости предобработки входных данных, обработкой как вещественных, так и категориальных признаков, поддержкой работы с отсутствующими значениями.

Напомним, что дерево решений [2] строит разбиение пространства признаков на непересекающиеся области путем его рекурсивного разбиения. Фактически каждому узлу дерева соответствует некоторая область пространства признаков и правило, по которому осуществляется ее разделение на две, приписанных к дочерним вершинам. Так как задача построения оптимального дерева, минимизирующего ошибку на обучающей выборке при заданном ограничении на его размеры, является NP-трудной, для обуче-

ния данной модели используется жадная стратегия максимального уменьшения функции, описывающей неоднородность данных *I(D)*, где D есть область в пространстве признаков, соответствующая некоторому узлу.

Пусть $y \in \{1, \dots, K\}$, где K – число категорий объектов, и область D содержит M объектов из обучающей выборки, обозначим относительную частоту класса k в данном узле дерева $\hat{p}_k = \frac{1}{M}\sum_{x_i \in D} I(y_i = k)$. Класс $\tilde{k}$ с наибольшей относительной частотой соответствует ответу, предсказываемому для всех объектов, относящихся к данному узлу. В качестве функции, описывающей неоднородность данных в узле *I(D)*, возможны следующие варианты [9]:

- ошибка классификации: $\frac{1}{M}\sum_{x_i \in D} I\left(y_i \neq \tilde{k}\right) = 1 - \hat{p}_{\tilde{k}}$;
- индекс Джини: $\sum_{k \neq \tilde{k}} \hat{p}_k \hat{p}_{\tilde{k}} = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k)$;
- кросс-энтропия: $-\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$.

Обучение модели случайного леса состоит из независимого построения определенного количества деревьев по следующей процедуре: генерация случайной подвыборки из обучающей выборки (размер данной подвыборки обычно составляет 50–70% от размера всей обучающей выборки) и построение дерева решений по данной подвыборке (причем, в каждом новом узле дерева переменная для разбиения выбирается не из всех признаков, а из случайно выбранного их подмножества небольшой мощности).

2. Описание базы WordNet

WordNet [7] представляет собой большую базу данных существительных, глаголов, прилагательных и наречий английского языка, сгруппированных в наборы синонимов, каждый из которых соответствует определенной сущности. Между данными наборами определены связи, которые отражают их семантические и лексические отношения. Среди типов этих связей можно выделить отношение включения, которое связывают более общие понятия (например, «мебель») с более детальными (например, «шкаф»), который, в свою очередь, может быть связан с понятием «книжный шкаф» и т.д. (очевидно, что данное отношение обладает свойством транзитивности). Общую структуру можно описать в виде направленного ациклического графа, вершинам которого соответствуют понятия, а ребрам – отношения между данными понятиями. В построенном графе можно выделить корневую вершину, которой соответствует понятие «сущность», являющуюся общим предком для всех остальных вершин.

3. Алгоритм построения случайного леса с использованием модифицированной матрицы штрафа

Одним из возможных использований семантической иерархии объектов при обучении ансамбля деревьев решений является её применение при построении каждого отдельного дерева в ансамбле путем модификации функции неоднородности (индекс Джини). Основная цель данной модификации заключается в том, чтобы увеличить «штраф» за нахождение в узле объектов, помеченных категориями, находящимися далеко друг от друга в семантической иерархии. Вычисляемый таким образом индекс Джини можно записать следующим образом: $\sum_{k \neq \tilde{k}} w_{k\tilde{k}} \hat{p}_k \hat{p}_{\tilde{k}}$, где веса w_{ij} вычисляются следующим образом:

Вход: H – граф, описывающий структуру понятий в базе WordNet, h – высота дерева; $\{C_1, \dots, C_k\}$ – классы объектов, для которых необходимо вычислить веса.

1. Для всех $i = 1, \dots, K$
   a. $w_{ii} = 0$.
   b. Для всех $j = i + 1, \dots, K$

i. Вычислить $h(i,j)$ – максимальную глубину узла, являющегося общим предком для классов C_i и C_j в графе H.

ii. $w_{ij} = h - h(i,j), w_{ji} = w_{ij}$.

2. Для всех $i = 1, \dots, K$
   a. Нормировать w_{ij} таким образом, чтобы $\sum_{j=1}^{K} w_{ij} = 1$.

4. Иерархический алгоритм классификации изображений с использованием случайного леса

Один из наиболее известных подходов к решению задачи классификации объектов с большим числом классов заключается в использовании иерархической классификации [8]. Основная идея данного подхода заключается в построении иерархии классов объектов, организованной в виде дерева таким образом, что классы, относящиеся к каждому узлу дерева, делятся на некоторое число кластеров, каждый из которых соответствует одному узлу-потомку. Процесс продолжается до тех пор, пока в каждом терминальном узле дерева не будет содержаться ровно по одному классу. В каждом узле дерева строится классификатор (обычно бинарный), который обучает модель классификации кластеров классов, относящимся к узлам-потомкам. Для классификации нового объекта применяется цепочка из классификаторов, ведущая из корня дерева к терминальному узлу, соответствующему тому или иному классу.

Для построения дерева, описывающего иерархию объектов в базе WordNet, разработан следующий алгоритм.

Вход: H – граф, описывающий структуру понятий в базе WordNet; $\{C_{i_1}, \dots, C_{i_m}\}$ – классы объектов, для которых необходимо построить разбиение.

1. Если $m = 1$, то СТОП.
2. Вычислить $h(i_1, \dots, i_m)$ – минимальную глубину узла, являющегося общим предком для всех классов $C_{i_1}, \dots, C_{i_m}$ в графе H.
3. Для всех $i = i_1, \dots, i_m$
   a. Вычислить $C(i)$ – предка узла i глубины $h(i_1, \dots, i_m) + 1$.
5. Выполнить разбиение множества $\{C_{i_1}, \dots, C_{i_m}\}$ на подмножества $CG_1 \cup \dots \cup CG_K$, где $CG_k = \{C_{j_1}, \dots, C_{j_k}\}$, $C(j_1) = C(j_2) = \dots = C(j_k)$.
6. Выполнить шаги (1)-(5) для всех $CG_1, \dots, CG_K$.

Таким образом, выходом данного алгоритма является дерево T с числом терминальных узлов, равным числу возможных категорий, описывающее иерархию категорий. В каждом узле данного дерева строится отдельный классификатор (случайный лес), объединение данных классификаторов составляет иерархический классификатор. Алгоритм построения может быть описан следующим образом.

Вход: $D = \{(x^i, y^i),\ i = 1, \dots, n\}$ – обучающая выборка; T – дерево, описывающее иерархию категорий объектов:

1. Для каждого нетерминального узла $t \in T$
   a. получить список категорий $\{C_{i_1}, \dots, C_{i_m}\}$, соответствующих данному узлу;
   b. выбрать подмножество из обучающей выборки $\widetilde{D} \subseteq D$ такое, что метки классов для всех объектов из $\widetilde{D}$ принадлежат множеству $\{C_{i_1}, \dots, C_{i_m}\}$;
   c. получить список потомков узла t: $\{t_1, \dots, t_K\}$ и соответствующих им наборов категорий $\{CG_1, \dots, CG_K\}$;
   d. обучить классификатор на наборе данных $\overline{\widetilde{D}}$, для которого вектора признаков $\overline{\widetilde{x}}^i$ соответствуют векторам признаков из $\widetilde{D}$, а категории $\overline{\widetilde{y}}^i \in \{1, \dots, K\}$ (в зависимости от того, какому набору категорий из $\{CG_1, \dots, CG_K\}$ принадлежит y^i).

5. Вычислительный эксперимент

В рамках работы была проведена первоначальная апробация предложенных алгоритмов. Алгоритм построения случайного леса с модифицированной матрицей штрафа (RF_COST_MATRIX) и иерархический алгоритм классификации изображений с использованием случайного леса (RF_HIERARCHICAL) сравнивались с алгоритмом, объединяющим в себе построение словаря кодовых слов с использованием алгоритма KMeans и линейного SVM в качестве классификатора (BOW+SVM). Сравнение проводилось на базе данных ILSVRC 2010 [10], являющейся подмножеством базы данных ImageNet, содержащим 1000 категорий. Данные делились в соотношении 50:50 на обучающую и тестовую выборки. Каждое изображение масштабировалось таким образом, чтобы максимальная размерность не превосходила 300 пикселей, затем извлекались dense SIFT признаки (перекрывающиеся патчи размером 20x20 пикселей с шагом в 10 пикселей). Из извлеченных таким образом признаков выбиралось случайным образом 10^7, на которых выполнялось построение словаря (1000 слов, соответствующих центроидам кластеров, найденным алгоритмом KMeans). При обучении машины опорных векторов использовался метод «один-против-всех», таким образом, общее число обученных классификаторов равнялось 1000 (число категорий). Обучение алгоритма случайного леса выполнялось с параметрами, выбираемыми по умолчанию. Результаты приведены в таблице 1.

Таблица 1. Ошибка классификации на тестовой выборке на базе данных ILSVRC2010

Алгоритм	Значение ошибки на тестовой выборке
RF_COST_MATRIX	0.53
RF_HIERARCHICAL	0.44
BOW+SVM	0.42

Как видно из табл. 1, иерархический алгоритм классификации на основе случайного леса дает примерно одинаковые результаты по сравнению с алгоритмом, объединяющим в себе построение словаря кодовых слов с использованием алгоритма KMeans и линейного SVM в качестве классификатора. Среди возможных подходов для повышения точности предсказания с использованием иерархического алгоритма следует выделить отдельное обучение словаря в каждом узле дерева, которое описывает иерархию объектов.

Работа выполнена в рамках программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы», государственный контракт № 11.519.11.4015.

Литература

1. Breiman L. Random Forests // Machine Learning. 2001. V. 45, №. 1, P. 5–32.
2. Breiman L., Friedman J.H., Olshen R.A., Stone C.J. Classification and Regression Trees. Wadsworth & Brooks, 1984.
3. CaltechDatasets – [http://www.vision.caltech.edu/html-files/archive.html].
4. Deng J., Berg A., Fei-Fei L. Hierarchical Semantic Indexing for Large Scale Image Retrieval // IEEE Computer Vision and Pattern Recognition (CVPR) – 2011.
5. Deng J., Berg A., Li K., Fei-Fei L. What does classifying more than 10,000 image categories tell us? // Proceedings of the 12th European Conference of Computer Vision (ECCV) – 2010.

6. Deng J., Dong W., Socher R., Li L., Li K., Fei-Fei L. ImageNet: A large-scale hierarchical image database // In CVPR09. – 2009.
7. Fellbaum C. WordNet: An Electronic Lexical Database. – MIT Press, 1998.
8. Griffin G., Perona P. Learning and using taxonomies for fast visual categorization // In CVPR, 2008.
9. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. – Springer, 2008.
10. Large Scale Visual Recognition Challenge 2010 (ILSVRC2010) [http://www.image-net.org/challenges/LSVRC/2010/index].

РАСПРЕДЕЛЕНИЕ РЕСУРСОВ В СЕТЕВЫХ СТРУКТУРАХ С АКТИВНЫМИ ЭЛЕМЕНТАМИ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

М.Х. Прилуцкий, У.С. Кулакович

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлена общая математическая модель функционирования сложных производственных систем, проводится её исследование. В рамках построенной математической модели поставлены оптимизационные задачи объёмно-календарного планирования, предложены алгоритмы приближенного решения поставленных задач с использованием параллельной вычислительной среды.

Целью данной работы является построение общей математической модели распределения ограниченных ресурсов в сетевых структурах с активными элементами, в рамках которой формализуется широкий класс задач производственного планирования.

Как и в [1], пусть $G = (V,A)$, $A \subseteq V^2$, – ориентированный граф без петель и контуров, моделирующий процесс изготовления изделия. Множество вершин графа V соответствует элементам системы, моделирующим процесс выполнения операций по изготовлению изделий, множество дуг A – моделирует взаимозависимость выполнения операций. Обозначим через $K(j) = \{i \mid (i,j) \in A,\ i \in V\}$ - множество номеров вершин графа, непосредственное предшествующих вершине с j-ым номером, $j \in V$, а через $Q(j) = \{i \mid (j,i) \in A,\ i \in V\}$ - множество номеров вершин графа, непосредственно следующих после вершины с j-ым номером, $j \in V$. Разбиением множества V является совокупность множеств $V^{вх}$, $V^{вых}$, V^a, V^p , где

- $V^{вх}$ – множество входных элементов (определяют «начальные» операции по изготовлению изделий), $V^{вх} = \{j \mid K(j) = \varnothing,\ j \in V\}$;
- $V^{вых}$ – множество выходных элементов (определяют «конечные» операции по изготовлению изделий), $V^{вых} = \{j \mid Q(j) = \varnothing,\ j \in V\}$;
- V^a – множество активных элементов системы – операций, выполнение которых возможно с различной интенсивностью;
- V^p – множество пассивных элементов системы, под которыми будем понимать операции, выполняемые с заданной интенсивностью.

Не уменьшая общности (за счёт введения фиктивных элементов) будем полагать, что система имеет один входной и один выходной элементы. Будем считать, что для любого активного элемента время выполнения соответствующей операции может принимать любое значение от минимально возможного до максимального возможного.

Пусть I – множество характеристик, соответствующих элементам системы (например, время выполнения операции, её ресурсоёмкость). Пусть U^j – множество допустимых управлений, соответствующих активному элементу j, где $j \in V^a$; w^j – вектор, определяющий значения на входе j-го элемента системы, $j \in V$. Известны минимальные и максимальные возможные значения каждой из характеристик i на входе и выходе j-го элемента системы, $i \in I$, $j \in V$. Известны также вектор-функция, преобразующая входные характеристики каждого элемента системы в его выходные характеристики под воздействием допустимых управлений; вектор-функция, определяющая входные характеристики j-го элемента системы по выходным характеристикам всех элементов, непосред-

ственно предшествующих элементу j, $j \in V$. Задан директивный срок окончания изготовления изделия, а также величины возможного отклонения времени завершения изготовления изделия от заданного директивного срока.

В отличие от [1], здесь предполагается, что функционирование системы осуществляется по тактам планирования и в каждый из тактов известен объём ресурса, которым располагает система. Будем считать, что в системе распределяется только один ресурс, и он является нескладируемым. Известна функция, зависящая от интенсивности выполнения операции j, определяющая объём ресурса, который необходим для выполнения j-ой операции, т.е. известен минимально-необходимый объём ресурса для выполнения j-ой операции (при минимальной интенсивности) и максимально-необходимый (при максимальной интенсивности), $j \in V^a$.

Решением проблемы распределения ресурсов в системах с активными элементами является график выполнения операций, определяющий для каждого элемента системы такт времени начала и окончания операции при выбранных интенсивностях потребления ресурсов. График операций позволяет однозначно определить объём ресурса, необходимый для выполнения каждой из операций, вектор входных и выходных характеристик каждого элемента системы и объём ресурса, используемого системой в каждый из тактов планирования (не должен превышать заданного объёма ресурса, которым располагает система в каждый из тактов). Предполагается, что время окончания «последней» операции, соответствующей «выходному» элементу системы, должно определенным образом соответствовать заданному директивному сроку окончания изготовления изделия.

Построенная общая математическая модель, формализующая процесс распределения ресурсов, учитывает ограничения характеристик на входах и выходах элементов, согласует характеристики элементов, задает условия баланса между входными и выходными характеристиками элементов, а так же формализует условия допустимости функционирования системы, связанные с ограничениями на наличие ресурса, потребляемого активными элементами.

В рамках построенной общей математической модели ставится задача оптимального распределения ресурсов по критерию минимизации отклонения времени изготовления изделия от директивного срока, и задача минимизации потерь распределяемого нескладируемого ресурса.

Показано, что поставленные оптимизационные задачи относятся к классу NP-трудных и для их решения предлагаются приближенные алгоритмы, относящиеся к классу фронтальных алгоритмов. В основе фронтальных алгоритмов лежат процедуры упорядочения множества работ путем введения приоритетов работ. Строятся совокупности таких процедур, объединенные в «стратегии управления». Для решения поставленных задач предлагается параллельная схема, основанная на выборе различных стратегий, которые меняются за счет изменения приоритетов активных элементов системы.

Работа выполнена при частичной финансовой поддержке Минобрнауки России, государственное соглашение о представлении гранта №14.B37.21.0878.

Литература

1. Прилуцкий М.Х., Дикарев К.И. Распределение ресурсов в иерархических системах с активными элементами. – Вестник Нижегородского университета, 2012, №4.

ОБ ОДНОМ ИЗ АЛГОРИТМОВ ПОИСКА ОБЪЕКТОВ ПО БИОМЕДИЦИНСКИМ ДАННЫМ

А.Н. Решетников

Нижегородский госуниверситет им. Н.И. Лобачевского
E-mail: ANReshetnikov@mail.ru

Постановка задачи

На вход рассматриваемому алгоритму подаются данные компьютерной томографии или электронной микроскопии (всюду далее – CT- и EM-данные соответственно), представляющие собой трёхмерный массив вокселов. Каждому из элементов данного массива приписано определённое целочисленное значение плотности $density[i, j, k] \in Z$. При этом известны размеры каждого воксела вдоль осей **X**, **Y** и **Z** $ScaleX$, $ScaleY$ и $ScaleZ$ в некоторых метрических единицах. Таким образом, объём 1 воксела исходных данных равен $ScaleX \cdot ScaleY \cdot ScaleZ$ кубических единиц. Отметим, что данный 3D-массив можно интерпретировать двумя способами: как трёхмерное изображение и как последовательность полутоновых 2D-изображений (слоёв) одинаковой ширины и высоты.

По представленным данным требуется найти всевозможные объекты, при отсутствии какой бы то ни было априорной информации о них. Данная задача имеет важное практическое значение как один из промежуточных этапов в трёхмерной реконструкции объектов по таким данным и их последующем анализе конечным пользователем.

Многие из алгоритмов, решающие поставленную задачу, включают в себя 2 этапа:

1) сегментацию исходных данных для получения деталей искомых объектов или их начальных приближений;
2) обнаружение самостоятельных объектов по найденным сегментам.

Сегментация исходных данных

Учитывая специфику исходных биомедицинских данных (воксельная модель представления и скалярное поле значений плотности), для решения задачи сегментации использовался ***метод k-средних*** (или ***центров тяжести***) [1].

Для более детального представления сегментов, относящихся к разным объектам, предлагается использовать один из возможных нисходящих методов иерархической сегментации. Его суть состоит в том, что после сегментации исходных данных по значениям плотности можно ввести в рассмотрение сегменты более высокого уровня детализации, а именно, ***сегменты по расстоянию*** между вокселами и ***связные области*** (в пространственном смысле). Указанные сегменты в общем случае образуют иерархическую структуру, изображённую на рис. 1. При этом сегментацию по расстоянию можно осуществить итеративным алгоритмом, основанным на методе *k*-средних, рассматривая в качестве объектов пары индексов $\{(i, j)\}$, описывающих положение вокселов на одном и том же слое. В качестве меры рассогласования между вокселами x_1 и x_2 используется обычное евклидово расстояние, порождённое парами индексов. Связные области являются самым высоким уровнем детализации исходных данных. Для их поиска ис-

пользуется итеративный алгоритм разметки областей [2] в рамках конкретного сегмента по признаку 8-связности пикселов.

В случае послойной сегментации исходных данных (каждого слоя как 2D-изображения в отдельности) для дальнейшей идентификации объектов необходимо определить, какие сегменты на разных изображениях относятся к одним и тем же объектам. Для того, чтобы судить об этом, необходимо уметь сравнивать любые 2 сегмента друг с другом.

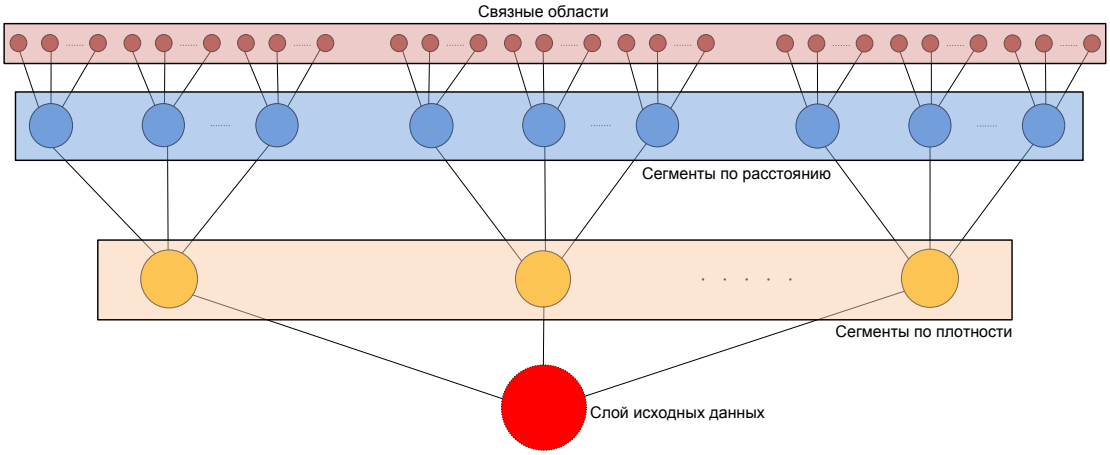


Рис. 1. Иерархическая структура сегментов после декомпозиции

Для построения дескриптора сегмента использовались следующие его атрибуты:

1) среднее значение плотности: $\bar{\mu} = \frac{1}{|S|}\sum_{i \in S} \mu_i$, (1)

где μ_i – значение плотности воксела с индексом i;

2) среднеквадратическое отклонение плотности: $\sigma_\mu = \frac{1}{|S|}\sum_{i \in S}\left(\mu_i - \bar{\mu}\right)^2$; (2)

3) геометрический центр масс $\left(\bar{x}, \bar{y}\right)$: $\bar{x} = \frac{1}{|S|}\sum_{i \in S} x_i$, $\bar{y} = \frac{1}{|S|}\sum_{i \in S} y_i$, (3)

где $\left(x_i, y_i\right)$ – координаты воксела с индексом i;

4) дисперсии координат вокселов (X и Y):

$$\sigma_X = \frac{1}{|S|}\sum_{i \in S}\left(x_i - \bar{x}\right)^2, \quad \sigma_Y = \frac{1}{|S|}\sum_{i \in S}\left(y_i - \bar{y}\right)^2; \quad (4)$$

5) объём сегмента в используемых кубических единицах:

$$V = |S| \cdot ScaleX \cdot ScaleY \cdot ScaleZ. \quad (5)$$

Атрибуты, вычисляемые по формулам (1) – (5), образуют признаковое описание сегмента S, или, по-другому, его ***дескриптор***, который далее будем обозначать $d(S)$.

Обнаружение объектов по найденным сегментам

Если исходные данные рассматривать как 3D-изображение, то результатом их сегментации является разбиение исходного множества вокселов на подмножества, соответствующие различным искомым объектам. Однако в том случае, когда применяется 2D-сегментация каждого слоя данных по отдельности, для идентификации объектов необходимо различить, какие сегменты на разных слоях относятся к одному и тому же или разным объектам. После этого для каждого объекта можно построить множество

точек, состоящее из соответствующих сегментов. Рассмотрим более детально метод, позволяющий решить данную проблему.

С помощью дескрипторов, которые ставятся в соответствие сегментам по формулам (1) – (5), можно определить степень рассогласования между сегментами s_1 и s_2 как расстояние ***Махаланобиса*** между их дескрипторами $d(s_1)$ и $d(s_2)$:

$$f(s_1, s_2) = \sqrt{\sum_{i=1}^{n} \frac{(d_i(s_1) - d_i(s_2))^2}{\sigma_i^2}}, \tag{6}$$

где

n – количество рассматриваемых атрибутов сегмента,

d_i – значение i - ого атрибута,

σ_i – среднеквадратическое отклонение i -го атрибута.

Определение. Последовательность сегментов на соседних слоях $\left(s_{i_{k_0}}, s_{i_{k_0+1}}, \ldots, s_{i_{k_0+K-1}}\right)$, где i_j – индекс сегмента на j - м слое, k_0 – начальный слой, а K – количество слоёв, назовём ***соответствием*** между данными сегментами.

Таким образом, задача идентификации объектов сводится к поиску всевозможных соответствий между сегментами, расположенными на разных слоях исходных данных. Поиск таких соответствий можно рассматривать как многошаговый ***динамический процесс***, в котором каждый момент времени определяется номером соответствующего слоя данных, состояниями являются сегменты, распределённые по слоям, а управлениями – переходы с сегментов текущего слоя на соседний. Следует отметить, что множество финальных состояний совпадает с множеством всех предварительно найденных 2D-сегментов, поскольку объект может начинаться и заканчиваться на любом слое исходных данных.

В процессе поиска соответствий рассматриваются сегменты самого верхнего уровня детализации, то есть листья иерархической структуры, описанной в разделе «Сегментация исходных данных». Все сегменты делятся на ***занятые*** (те, которые уже включены в одно из найденных соответствий) и ***свободные***. В самом начале работы алгоритма все сегменты помечаются как свободные. Как только сегмент оказался включён в какое-либо соответствие, он получает статус «занят». Поиск очередного соответствия начинается с первого свободного сегмента на слое с наименьшим индексом. Для того чтобы не рассматривать слишком маленькие и/или большие детали, полученные в результате иерархической декомпозиции сегментов, можно ввести ограничения на их объём *V*, который должен принадлежать диапазону $[V_{\min}; V_{\max}]$.

Определим стоимость перехода с текущего слоя на соседний как рассогласование между дескрипторами соответствующих сегментов (состояний), тогда значение функции $f(s_1, s_2)$ вычисляется по формуле (6), где s_1 – сегмент текущего слоя, а s_2 – сегмент соседнего слоя.

Тогда критерием качества найденного соответствия можно считать суммарное рассогласование между всеми соседними сегментами, образующими соответствие:

$$F = \sum_{k=k_0}^{k_0+K-2} f\left(s_{i_k}, s_{i_{k+1}}\right). \tag{7}$$

Таким образом, поиск соответствий между сегментами сводится к решению задачи минимизации функции F при перемещении от начального слоя к конечному по соответствующим сегментам (рис. 2).

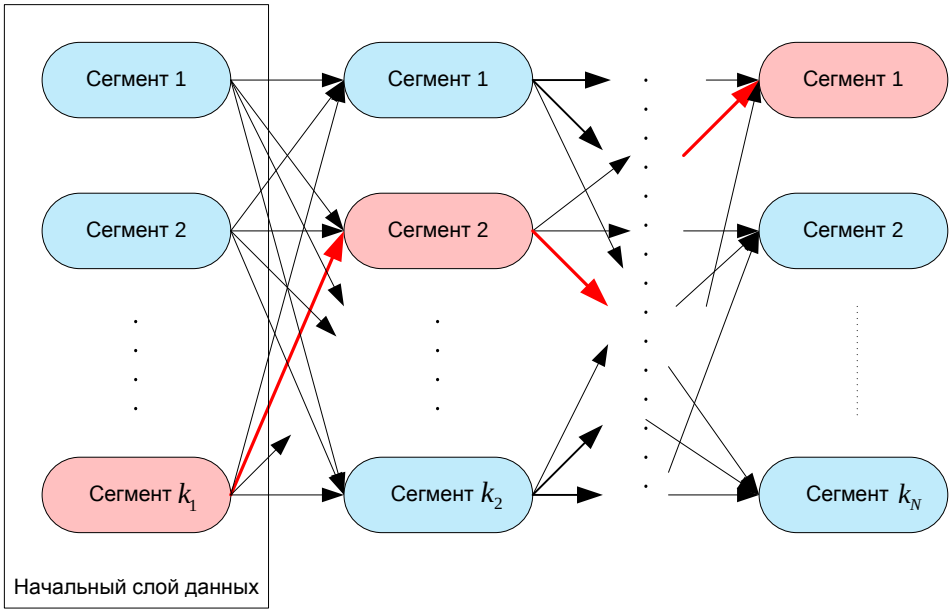


Рис. 2. Схема поиска соответствий между 2D-сегментами

Поскольку функция F аддитивна по промежутку слоёв, на которых расположены сегменты, и удовлетворяет принципу Беллмана в форме достаточного условия [3], для решения рассматриваемой задачи можно применить алгоритм пошаговой минимизации этой функции, основанный на методе рекуррентных соотношений Беллмана [3].

Приведём более подробное описание данного алгоритма применительно к рассматриваемой задаче, используя следующие обозначения:

s – массив сегментов,
IsSegmentUsed – булевы переменные (флаги) использования сегментов,
layer_index – индекс текущего слоя данных (служебная переменная),
Disparity – текущее значение минимальной стоимости переходов по соответствию,
N – общее количество слоёв исходных данных.

Шаг 1. Полагаем для всех сегментов **IsSegmentUsed = false**.
Шаг 2. В цикле по **I** от 0 до **N-1** и по сегментам **s[I][J]**, принадлежащим слою **I**, если ***IsSegmentUsed[I][J] == false***, выполняем ***Шаг 3 – Шаг 5***.
Шаг 3. Полагаем **layer_index = I, Disparity = 0**.
Шаг 4. Создаём новое соответствие и добавляем в него сегмент с индексом **J** на слое **layer_index**.
Шаг 5. Пока **layer_index<N**, выполняем ***Шаг 6 – Шаг 10***. После этого переходим на ***Шаг 2***.
Шаг 6. В цикле по **K** по сегментам слоя **layer_index + 1** выбираем тот, объём которого принадлежит заданному диапазону $[V_{\min}; V_{\max}]$ и который доставляет минимум величины **Disparity +** ***F(s[layer_index][J], s[layer_index+1][K]).*** Обозначим его **L**.
Шаг 7. Добавляем сегмент с индексом **L** в созданное соответствие.
Шаг 8. Полагаем **Disparity = Disparity + F(s[layer_index][J], s[layer_index+1][L])**.
Шаг 9. IsSegmentUsed[layer_index+1][L] = true.
Шаг 10. Увеличиваем **layer_index** на 1 и переходим на ***Шаг 5***.

Конец алгоритма

На рис. 3 в верхнем ряду показаны несколько последовательно расположенных слоёв одной из томограмм, а в нижнем – результаты сегментации каждого из слоёв. При этом одинаковыми цветами раскрашены сегменты, образующие один и тот же объект.

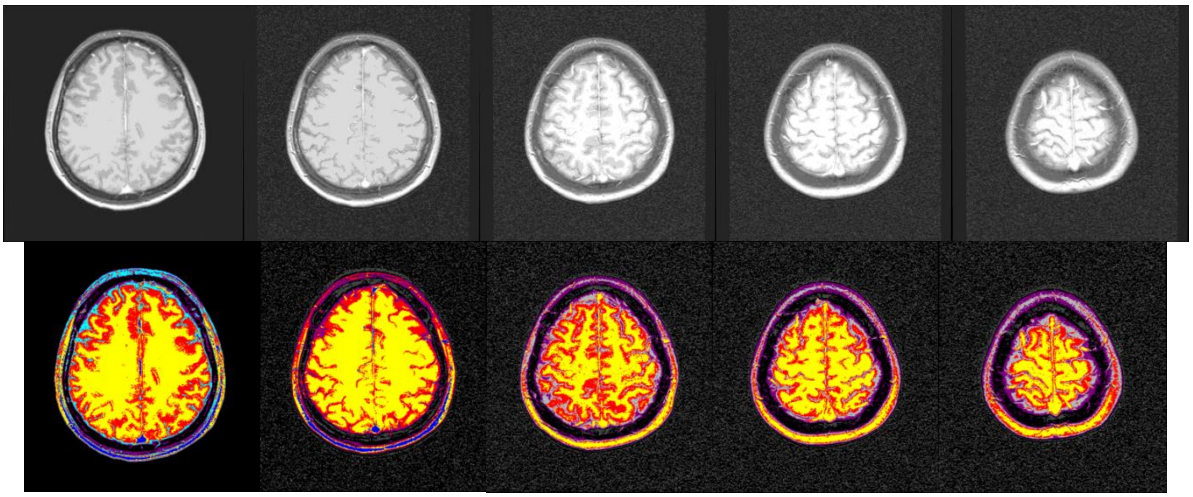


Рис. 3. Пример результатов поиска соответствий между 2D-сегментами

Повышение производительности вычислений

Вышеописанный алгоритм использовался при разработке программной системы Neocortex. В ходе этой разработки были реализованы как последовательная, так и параллельная (для GPU, с использованием технологии OpenCL) версии сегментации методом *k*-средних для решения поставленной задачи, которые используются для получения сегментов первого и второго уровней декомпозиции. При этом на каждом из этих двух этапов декомпозиции было получено ускорение от 4 до 10 раз в сравнении с последовательными реализациями [4].

Дальнейшее повышение производительности связано с разработкой параллельной схемы разметки связных областей (третий уровень декомпозиции сегментов) и алгоритма поиска соответствий между сегментами. Для этого целесообразно использовать возможности как центрального, так и графических процессоров.

Литература

1. Lloyd S. Least square quantization in PCM's. Bell Telephone Laboratories Paper, 1957.
2. Форсайт Д. Компьютерное зрение. Современный подход / Д. Форсайт, Ж. Понс. – Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 928 с.
3. Беллман Р. Динамическое программирование. – М.: Изд-во иностранной литературы, 1960.
4. Решетников А.Н. Параллельная реализация сегментации компьютерных томограмм методом центров тяжести с использованием технологии OpenCL // Высокопроизводительные параллельные вычисления на кластерных системах: Доклад на XI Всероссийской конференции, Н. Новгород, 1-3 ноября 2011. – С. 258–261.

КОМПАКТНЫЕ СУПЕР-ЭВМ ПРОИЗВОДСТВА ФГУП «РФЯЦ-ВНИИЭФ». ИТОГИ, ИННОВАЦИИ, ПЕРСПЕКТИВЫ

А.С. Рыбкин, Т.А. Агапова, Н.А. Дмитриев, В.И. Игрунов, Ю.Н. Корзаков, И.А. Крючков, В.Н. Лашманов, Ю.В. Логвин, А.Г. Ломтев, А.Н. Петрик, М.А. Рыбаков, А.Г. Селякин, Г.П. Семенов, В.Н. Стрюков, А.Ю. Ушаков, А.В. Шатохин, С.Н. Шлыков, Д.Р. Ярулин

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Начиная с 2009 года в ФГУП «РФЯЦ-ВНИИЭФ» ведутся интенсивные научно-исследовательские и опытно-конструкторские работы с целью создания компактных супер-ЭВМ рекордной производительности. Малогабаритные высокопроизводительные вычислительные системы, оснащенные базовым системным и прикладным программным обеспечением, ориентированы на решение широкого круга научных и инженерных задач. На первом этапе были созданы две компактные супер-ЭВМ производительностью 1 Тфлоп/с, как с использованием только универсальных процессоров, так и с применением арифметических ускорителей. В дальнейшем производительность была увеличена до 3-3,5 Тфлоп/с.

В 2012 году разработана компактная супер-ЭВМ производительностью 5 Тфлоп/с, задействующая ресурсы только универсальных процессоров. Прототип специализированной (с применением арифметических ускорителей) компактной супер-ЭВМ обеспечивает производительность не менее 8 Тфлоп/с.

Одной из уникальных особенностей созданных и разрабатываемых компактных супер-ЭВМ является инновационная жидкостно-воздушная подсистема охлаждения, которая позволяет понизить температуру основных компонентов и поддерживать в рабочем режиме, тем самым повышая их надежность, а так же снизить уровень акустического шума от вычислительной системы и повысить комфортность работы. В специализированном исполнении плотность компоновки арифметических ускорителей существенно выше, что, в условиях ограниченного физического объема, повышает общую производительность системы в целом.

Разработана система жидкостного охлаждения высокопроизводительных вычислительных систем на принципе полного погружении электронных компонент в диэлектрическую жидкость. Циркуляция этой жидкости обеспечивает перенос избытков тепла от электронных элементов через систему отвода в окружающую среду. Это позволяет значительно снизить уровень акустического шума элементов системы охлаждения относительно традиционной воздушной, а также позволяет в перспективе реализовать возможность повторного использования тепла. На этой базе создан макетный образец вычислительной системы производительностью 5 Тфлоп/с. Также на основе полученного опыта начата разработка blade-корзины, которую можно будет использовать для построения не только компактных супер-ЭВМ, а также супер-ЭВМ средней и предельной производительности.

Аппаратные подсистемы размещаются в корпусах уникальной конструкции, разработанных ведущими специалистами РФЯЦ-ВНИИЭФ.

Программное обеспечение компактных супер-ЭВМ построено на базе свободно распространяемого системного программного обеспечения и разработок РФЯЦ-ВНИИЭФ, как системных, так и прикладных.

АССОЦИАТИВНОЕ СОКРЫТИЕ ДАННЫХ С ПРИМЕНЕНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

А.Г. Савельев

Казанский национальный исследовательский технический университет им. А.Н. Туполева

В статье рассматривается стеганографический алгоритм сокрытия данных, реализованный с применением графических процессоров фирмы Nvidia.

Сокрытие данных происходит с помощью алгоритма сокрытия, основанного на двумерно-ассоциативном механизме маскирования [1].

Формирование стегоконтейнеров происходит на GPU, в которых реализована технология CUDA [2-5]. CUDA (англ. Compute Unified Device Architecture) – технология GPGPU (англ. General-Purpose computing on Graphics Processing Units – вычисления общего назначения на графических процессорах), позволяющая программистам реализовывать на языке программирования Си алгоритмы, выполнимые на графических процессорах видеокарт GeForce восьмого поколения и старше. Использование этой технологии значительно ускоряет процесс создания стегоконтейнеров. Причина заключается в том, что CUDA использует параллельную модель вычислений, когда каждый из SIMD процессоров выполняет ту же инструкцию над разными элементами данных параллельно.

CUDA включает два API: высокого уровня (CUDA Runtime API) и низкого (CUDA Driver API), хотя в одной программе одновременное использование обоих невозможно, нужно использовать или один, или другой. Высокоуровневый API работает сверху низкоуровневого, все вызовы Runtime транслируются в простые инструкции, обрабатываемые низкоуровневым Driver API. Но даже высокоуровневый API предполагает знание основ об устройстве и работе видеокарт NVIDIA.

Модель программирования в CUDA предполагает группирование потоков. Потоки объединяются в блоки потоков (thread block) – одномерные или двумерные сетки потоков, взаимодействующих между собой при помощи разделяемой памяти и точек синхронизации. Программа (ядро, kernel) исполняется над сеткой (grid) блоков потоков (thread blocks). Одновременно исполняется одна сетка (рис. 1). Каждый блок может быть одно-, двух- или трехмерным по форме, и может состоять из 512 потоков на текущем аппаратном обеспечении.

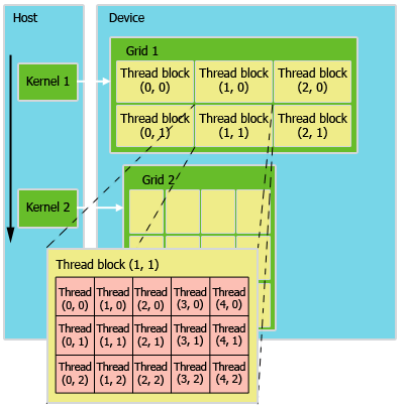


Рис. 1

Для решения задачи сокрытия данных стеганографическим методом было разработано две программы. Одна программа реализована с применением CPU, а другая – с применением GPU. В ходе эксперимента установлено, что для формирования 1000 стегоконтейнеров на процессоре Intel Pentium P6200 2,13GHz программе потребовалось 5 мс, а на процессорах видеокарты GeForce 8800GT – 3 мс.

Литература

1. Райхлин В.А., Вершинин И.С., Глебов Е.Е. К решению задачи маскирования стилизованных двоичных изображений // Вестник КГТУ им. А.Н. Туполева. 2001. №1. С. 42–47.
2. Основы CUDA. Введение, GPGPU – [http://steps3d.narod.ru/tutorials/cuda-tutorial.html].
3. Джейсон Сандерс, Эдвард Кэндрот. Технология CUDA в примерах. Введение в программирование графических процессоров: Пер. с англ. А.А. Слинкина – М.: Изд-во ДМК Пресс, 2011.
4. Параллельное программирование и вычислительная платформа CUDA, NVIDIA – [http://www.nvidia.ru/object/cuda_home_new_ru.html].
5. Nvidia CUDA – [www.gpgpu.ru].

РЕАЛИЗАЦИЯ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ В УЧЕБНОМ ПРОЦЕССЕ

С.В. Самохвалов

Владимирский госуниверситет им. А.Г. и Н.Г. Столетовых

Рассматривается создание среды для обучения параллельному программированию. Описаны компоненты, которые входят в эту среду, ее возможности. Приведены практические примеры использования в учебном процессе.

Введение

В настоящее время параллельные вычислительные системы широко применяются в промышленности, и спрос на них растет. В связи с этим существует потребность в подготовке кадров в данной области.

На настоящий момент для применения в промышленности существует немало готовых решений. Но не все из них могут быть легко применены в процессе обучения. В ряде случаев возникают проблемы с развертыванием и настройкой программного обеспечения, а также при интеграции продуктов разного типа друг с другом.

Целью данной работы является создание полноценной инструментальной среды для обучения параллельному программированию. Такая среда должна позволять и, по возможности, упрощать написание, отладку, тестирование и запуск параллельных программ.

1. Компоненты среды

Среда состоит из следующих компонетов:

1) вычислительные узлы, объединенные в сеть;
2) среда разработки и отладки (ide);
3) набор библиотек параллельного программирования (mpi, openmp и др.);
4) набор документации;
5) дополнительные компоненты (системы пакетной обработки заданий, системы мониторинга и др.).

2. Различные уровни паралеллизма

Среда должна помогать при обучении на различных уровнях гранулярности задания.

1. Крупнозернистый: задача распределяется по узлам кластера, внутри GRID-системы и т.д. То есть задача разбивается на подзадачи, которые могут решаться параллельно.
2. Среднезернистый: уровень потоков. К такому типу параллелизма относятся многоядерные и многопроцессорные вычислительные системы, технология HyperThreading – системы с общей памятью.
3. Мелкозернистый: параллелизм уровня инструкций. Сюда входят:
   - мультипроцессоры;
   - архитектуры VLIW и EPIC;
   - макрокоманды AMD.

3. Решение задач с использованием разного уровня параллелизма

В созданной среде были разработаны, отлажены и протестированы программы:

1. Умножение матрицы на вектор.
2. Умножение матрицы на матрицу.
3. Шифрование с помощью видеокарты.
4. Нахождение простых чисел Мерсенна.
5. Сортировка чисел (разными методами).

Литература

1. Богданов А.В., Станкова Е.Н., Павлова М.И. Высокопроизводительные алгоритмы – [http://www.csa.ru/analitik/distant/q_1.html].
2. Шпаковский Г.И. Реализация параллельных вычислений: кластеры, многоядерные процессоры, грид, квантовые компьютеры. – Минск: БГУ, 2010. 155 с.

АРХИТЕКТУРА ТЕЛЕМЕДИЦИНСКОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ГЕТЕРОГЕННОГО ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА В КАЧЕСТВЕ СЕРВЕРНОЙ ЧАСТИ

В.А. Сапрыкин

Нижегородский госуниверситет им. Н.И.Лобачевского

В настоящей работе описывается подход к созданию общей архитектуры телемедицинской системы на базе гетерогенного вычислительного кластера. Рассматривается задача обеспечения доступа к данным врачам, пациентам, обучающимся и исследователям. Предлагается механизм интегрирования в систему модулей решения вычислительно-емких задач в виде облачных сервисов. Затрагиваются вопросы разграничения доступа для разных категорий пользователей.

Введение

Целью работы является создание единой телемедицинской системы для различных медицинских учреждений и исследователей. Первый шаг, который нужно сделать – это распределенное хранение и обеспечение централизованного доступа к большим объемам изображений (в первую очередь томограмм) одного диагностического учреждения. Ввиду достаточно малой распространенности томографов, актуально предоставить врачам других учреждений доступ к медицинским изображениям конкретного пациента. При этом у диагностического учреждения должна быть возможность контролировать такой доступ. Обратно, несколько диагностических учреждений могут решить разместить свои данные в едином хранилище для облегчения обмена снимками. Пациент так же должен иметь возможность удаленно просматривать свои снимки со своего компьютера, предварительно запросив учетные данные у диагностического учреждения. Для выполнения исследований, создания алгоритмов предварительной диагностики и обучения персонала полезно иметь централизованный доступ к большой коллекции медицинских изображений от нескольких диагностических учреждений. В этом смысле необходимо, чтобы исследователям были доступны обезличенные изображения, а учреждению – полный объем информации по своим снимкам.

Большой объем трехмерных медицинских изображений и вычислительная сложность их обработки определяют высокие требования к аппаратному обеспечению для серверной и клиентской частей телемедицинской системы. Использование гетерогенного вычислительного кластера в качестве серверной части системы дает: ускорение обработки ресурсоемких запросов; уменьшение требований к аппаратному обеспечению на стороне клиента за счет переноса большей части вычислений на сервер; возможность встраивать в телемедицинскую систему модули высокопроизводительных вычислений и моделирования, которые могут быть выполнены за приемлемое время лишь на кластере. Реализация запуска на выполнение таких вычислений по команде с клиента открывает возможность создания облачных сервисов.

Постановка задачи

Для создания системы хранения медицинских изображений, способной решать описанные выше задачи, необходимо кроме самих изображений хранить метаинформацию по каждому изображению и информацию о взаимосвязи отдельных изображений.

Эти данные должны включать как минимум сведения об учреждении, о пациенте, дату снимка, тип патологии или признак ее отсутствия и часть тела на изображении.

Ввиду большого объема медицинских изображений (в частности томограмм) требуется способность системы работать в условиях, когда файлы изображений распределены по различным серверам в медицинских учреждениях и специально выделенным серверам. Жестким требованием является поддержка формата DICOM [4] – индустриального стандарта создания, хранения, передачи и визуализации медицинских изображений и документов обследованных пациентов. Нужно максимально ускорить передачу данных между клиентом и сервером.

Предусмотреть возможность отдельного хранения персональных данных, чтобы при различных вариантах развертывания системы иметь возможность оставить персональные данные в стенах медицинского учреждения. Отсюда вытекает необходимость создания подсистемы анонимизации и восстановления персональных данных DICOM изображений.

Для обеспечения конфиденциальности информации нужно предоставлять доступ к системе только уполномоченным пользователям, посредством запроса имени пользователя и пароля. На основе имени пользователя требуется разграничивать доступ к данным для разных категорий пользователей. Поскольку система работает через сеть, для обмена данных между узлами можно пользоваться протоколами с шифрованием.

Требуется разработать механизм подключения к системе модулей высокопроизводительных параллельных расчетов и моделирования на вычисленном кластере с использованием центральных и графических процессоров. Такие модули должны запускаться по команде с удаленного клиента. Входные данные должны извлекаться из системы хранения, а выходные файлы должны помещаться в хранилище по окончании расчетов.

Архитектура системы

Общая архитектура системы изображена на рис. 1. Она отражает ситуацию, когда несколько медицинских учреждений передали свои обезличенные изображения в единое хранилище, оставив себе заботу о персональных данных. Так же эти учреждения по запросу выдают пациентам учетные данные, позволяющие последним получить доступ на чтение к своим снимкам или передать такой доступ другому врачу. Наконец, группа исследователей может работать с совокупным объемом обезличенных медицинских изображений.

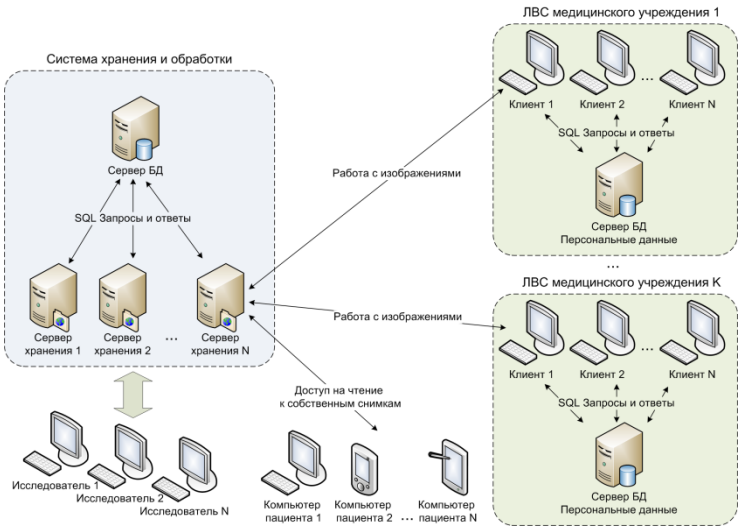


Рис. 1. Общая архитектура системы хранения и обработки медицинских изображений

Серверная часть хранилища представлена центральной СУБД и несколькими серверами хранения, которые непосредственно хранят и обрабатывают данные. Кроме того, в локальной сети каждого медицинского учреждения есть база данных, хранящая персональную информацию пациентов. Запросы к серверной части со стороны любого из клиентов могут быть выполнены удаленно.

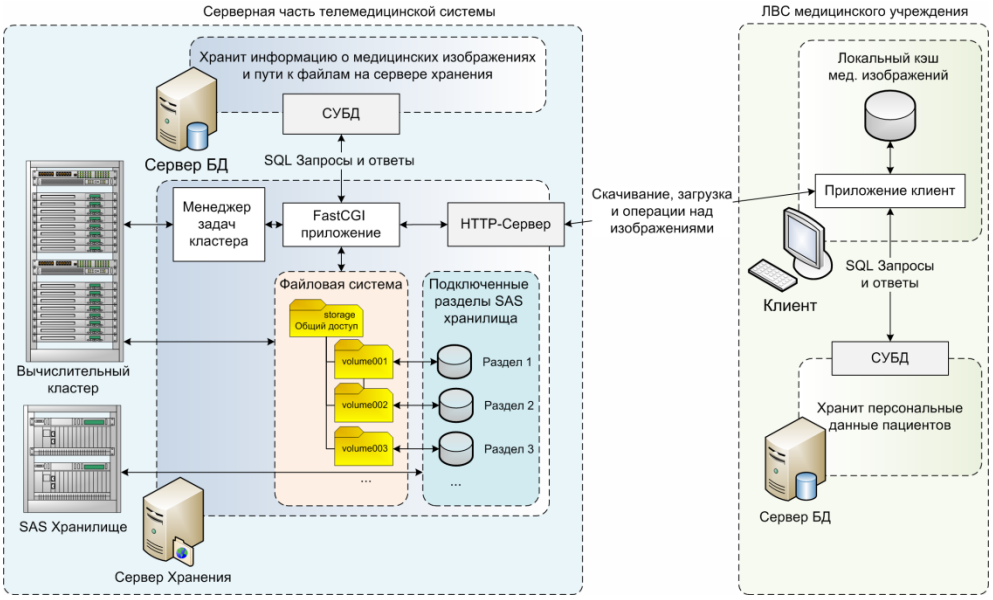


Рис. 2. Устройство компонентов системы хранения и обработки медицинских изображений

Детальное устройство различных компонентов описанной архитектуры показано на рис. 2. Основной программный модуль на сервере хранения – это сервер приложений, выполняющий запросы клиентов. Сервер приложений реализован как FastCGI приложение для web-сервера и взаимодействует с клиентским приложением по протоколу HTTP. Это существенно упрощает дальнейшую разработку, позволяя сосредоточиться на реализации нужной функциональности, абстрагируясь от решения технических проблем принятия запросов и передачи ответов.

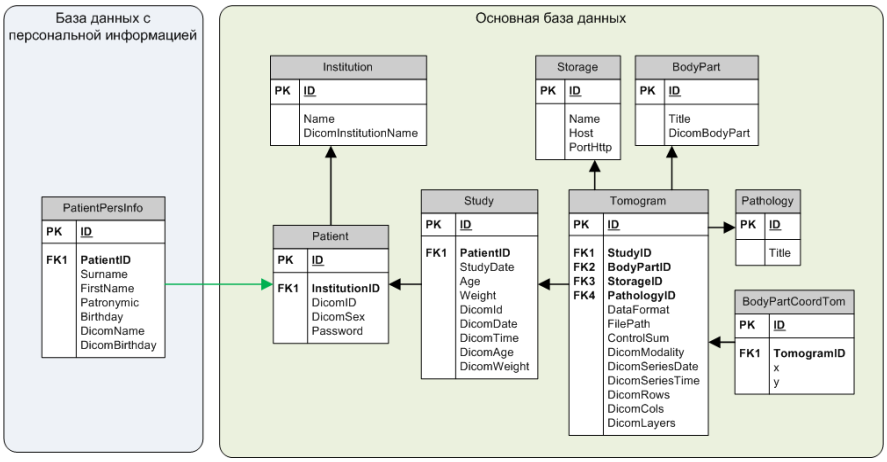


Рис. 3. Схема данных системы хранения и обработки медицинских изображений

Медицинские изображения не хранятся в базе данных, а размещаются в виде файлов на жестких дисках. Центральная база данных лишь хранит различные метаданные, касающиеся медицинских изображений. Сервер приложений имеет доступ к центральной базе данных и к каталогу storage, в который в виде подкаталогов смонтированы

разделы жестких дисков хранилища. FastCGI приложение при импорте снимков упаковывает серии изображений в формате DICOM в единый файл и без сжатия размещает в подкаталогах “volume001”, “volume002” и т.д. в зависимости от наличия свободного пространства (см. рис. 2). Затем для томограммы в центральной базе данных указывается сервер хранения, на котором она находится, и путь к ней на этом сервере относительно каталога storage, например «/volume001/tom245».

Второй программный модуль под названием «Менеджер задач кластера» предназначен для управления запуском вычислительно тяжелых задач на кластере. Предполагается, что модуль постоянно запущен и находится в состоянии ожидания команды от сервера приложений.

Схема данных (рис. 3) помимо отражения присутствующих в предметной области сущностей и их атрибутов содержит ряд специальных атрибутов. В их числе поля с префиксом Dicom, содержащие точную копию данных из DICOM-файлов, помещенных в хранилище. Эти поля используются для установления соответствия между объектами БД и DICOM-объектами, а так же для хранения персональных данных в отдельной базе данных с целью последующего их восстановления в DICOM-файлах при загрузке изображении из хранилища в исходном виде.

Развертывание серверной части

Разработанная первая версия серверной части была развернута на базе вычислительного кластера ННГУ под управлением MS Widnows Server 2008 HPC Edition. Серверная часть системы состоит из сервера базы данных PostgreSQL (хранящего метаданные об изображениях) и одного сервера хранения (на который непосредственно загружаются изображения). В качестве HTTP-сервера используется Microsoft IIS. Для непосредственного хранения данных используется SAS (Serial Attached SCSI) хранилище. В целях масштабируемости его разделы монтируются как подкаталоги в каталог storage, вместо того, чтобы присваивать латинскую букву каждому разделу. В настоящий момент используется один раздел объемом 2 Tb, смонтированный в каталог /storage/volume001. В хранилище помещен опытный набор из 10000 МРТ томограмм общим объемом 70 Гб.

Особенности реализации передачи DICOM-изображения

Реализована возможность пакетного импорта большого количества медицинских изображений с функцией их обезличивания. При этом кроме записей о самих изображениях в базе данных автоматически создаются и все связанные записи (пациент, исследование, часть тела и др.) на основе информации из DICOM-файлов. По ходу всего процесса импорта отображается полоса прогресса и доступна кнопка прерывания операции. Все передачи изображений проверяются на корректность с помощью подсчета контрольных сумм по обеим сторонам канала и их сверки между собой. Перед загрузкой в хранилище каждое многослойное изображение в формате DICOM подвергается упаковке в единый файл специально разработанного формата. Для ускорения передачи данных по медленным каналам связи используется сжатие «на лету» упакованного DICOM-изображения. Это позволяет примерно вдвое уменьшить время передачи данных. Для ускорения повторного доступа к недавно созданным/запрошенным изображениям, реализована подсистема локального дискового кэширования. Она позволяет избежать повторного скачивания недавно использованных изображений с сервера.

В клиентском приложении реализована функция предварительного просмотра слоев DICOM-изображения без его полного скачивания из хранилища. При передаче такого одиночного слоя, производится его кодирование в формат PNG (сжатие без потерь алгоритмом DEFLATE) с уменьшением глубины цвета до 8 бит. Наряду с обеспечением предварительного просмотра слоев, а так же, с уменьшением объема передаваемых

данных, формат PNG дает возможность удаленного просмотра слоев томограммы через браузер за счет использования протокола HTTP.

Интеграция с подсистемами ресурсоемких вычислений и визуализации

На стороне клиента интеграция с различными программными модулями заключается в запуске приложения с передачей ему параметров командной строки. При необходимости выполняется подготовка входных и анализ выходных данных; отображение результатов работы приложения; загрузка входных данных из хранилища и сохранение результатов работы в хранилище и др.

Клиентская программа интегрирована с просмотрщиком InVols [6], позволяющим просматривать томограммы в различных режимах визуализации. Запуск InVols осуществляется нажатием одной кнопки после выбора томограммы в списке. Если нужного изображения нет в кэше, оно будет автоматически загружено.

Кроме того интегрированы два ресурсоемких консольных приложения: XMCML [7] – моделирование распространения зондирующего излучения в многослойных биологических тканях с произвольной геометрией границ слоев в задаче оптической диффузионной томографии и Picador [1] – реализация моделирования динамики плазмы. Для обеих программ создан пользовательский интерфейс задания параметров моделирования и запуска процесса моделирования. По окончании вычислений запускается программа визуализации результатов. Весь процесс моделирования описывается в автоматически генерируемом командном (BAT) файле.

Заключение

В результате проделанной работы был создан и развернут на вычислительном кластере ННГУ макет телемедицинской системы. Хранилище наполнено первой опытной порцией МРТ-томограмм различных частей тела. Доступная через интернет система для долговременного хранения и дистанционного использования данных медицинской диагностики обеспечена необходимым функционалом для ее практической эксплуатации и построения на ее основе телемедицинских и других систем исследовательской и практической медицины.

Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

Литература

1. Bastrakov S., Donchenko R., Gonoskov A., Efimenko E., Malyshev A., Meyerov I., Surmin I., Particle-in-cell plasma simulation on heterogeneous cluster systems // Journal of Computational Science. 2012. 3. P. 474–479.
2. Breton V., Dean K., Solomonides T. From Grid to Healthgrid. Proceedings of Healthgrid 2005 // Studies in Health Technology and Infjrmatics. IOS Press, 2005. Vol 112. P. 249–321.
3. Isaac N. Bankman Handbook of medical imaging. Processing and analysis. – USA: Academic Press, 2000. – 911 p.
4. Pianykh O.S. Digital Imaging and Communication in Medicine (DICOM). – Berlin, Germany: Springer-Verlag, 2008. – 383 p.
5. Standart «Digital Imaging and Communications in Medicine (DICOM)» – USA: National Electrical Manufacturers Association, 2011.
6. Гаврилов Н.И., Белокаменская А. А. Стерео-визуализация научных и медицинских объемных данных трассировкой лучей в реальном времени. Материалы XX конференции GraphiCon'2010, 20-24 сентября, Санкт-Петербург, Россия. С. 350-352.

7. Горшков А.В., Кириллин М.Ю. Monte Carlo simulation of brain sensing by optical diffuse spectroscopy // Journal of Computational Science, Elsevier, 2012.
8. Грабер М. Введение в SQL. – М.: ЛОРИ, 1996. – 479 с.
9. Сапрыкин В.А. Архитектура системы хранения и обработки медицинских изображений. Материалы XI Всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах», 17-21 ноября, 2011, Нижний Новгород. С. 279–283.

МОДЕЛИРОВАНИЕ РОСТА НАНОСТРУКТУР МЕТОДОМ МОЛЕКУЛЯРНОЙ ДИНАМИКИ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ CUDA

А.М. Сатанин, Д.В. Федосеев, Д.В. Ахматнуров

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлена реализация метода молекулярной динамики многокомпонентных систем с короткодействующим потенциалом, выполняемая полностью на CUDA. Алгоритм адаптирован для задач, связанных с физикой твёрдого тела.

Введение

Для изучения процессов образования наноструктур часто требуется моделировать динамику громадного числа взаимодействующих частиц. Типичный размер задачи делает необходимым применение высокопроизводительных вычислений. В данной работе рассмотрена реализация метода молекулярной динамики с использованием CUDA, разработанная для изучения процесса роста нанокристаллов кремния в оксидной матрице. Предложенный алгоритм адаптирован, прежде всего, для исследования процессов в твёрдом теле.

Поскольку взаимодействие атомов в твёрдых телах в основном описывается короткодействующими потенциалами, использовались оптимизации, ограничивающие число взаимодействующих атомов.

1. Описание задачи

Схема алгоритма представлена на рис. 1. Начальные положения атомов либо читаются из файла, либо генерируются. Для задания начальных скоростей использовано распределение Максвелла при заданной температуре. Граничные условия могут быть либо жёсткими, либо периодическими.

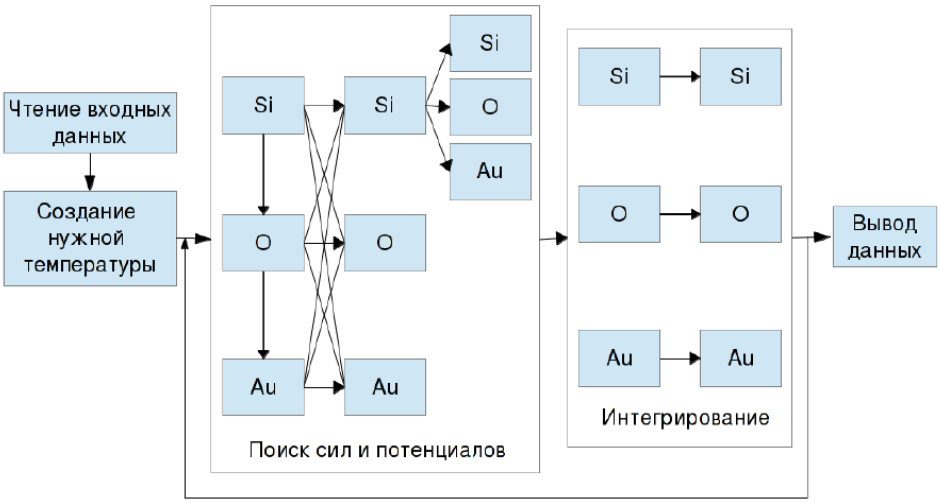


Рис. 1. Общая схема алгоритма

Итерацию моделирования молекулярной динамики можно разбить на два основных этапа:

1) Вычисление сил, действующих на каждый атом со стороны всех остальных атомов. Для моделирования многокомпонентных систем предусмотрена возможность задания разных потенциалов для описания взаимодействия разных пар атомов.
2) Интегрирование уравнений движения каждого атома.

Вычислительная сложность этапа нахождения сил зависит от типа используемого потенциала. В случае простейшего двухчастичного потенциала вида $\varphi_{ij} = f(r_i, r_j)$ она пропорциональна n^2. Вычислительная сложность этапа интегрирования – $O(n)$, где n – число атомов.

Для описания взаимодействия атомов кремния использовался эмпирический потенциал [1], имеющий вид

$$\varphi_{ij} = \left(Ae^{-\alpha r_{ij}} - B\left(\sum_k \psi(\vec{r}_i, \vec{r}_j, \vec{r}_k) \right) e^{-\beta r_{ij}} \right) f_{cut}(r_{ij}), \qquad (1)$$

где $f_{\text{cut}}(r)$, $B(x)$ и $\psi(x)$ – некоторые функции. Известно, что $f_{\text{cut}}(r)$ обращается в нуль при $r > R_{\text{cut}}$. Потенциал (1) является разновидностью потенциала Терсова [2], учитывающего влияние соседних атомов на характеристики связи. Наличие суммы по всем атомам в (1) приводит к увеличению вычислительной сложности этапа суммирования до $O(n^3)$. Таким образом, наиболее ресурсоёмким этапом является нахождение сил.

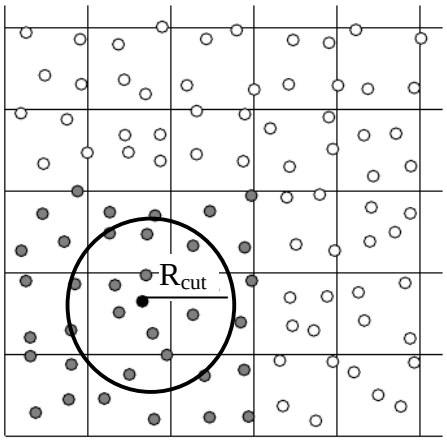


Рис. 2. Отсечение невзаимодействующих атомов

При оптимизации вычисления сил учтено, что потенциал обращается в нуль при $r > R_{\text{cut}}$. Наличие такого радиуса отсечения позволяет не рассчитывать взаимодействие с атомами, находящимися дальше R_{cut} от рассматриваемой. Применён подход spatial indexing [3], в котором всё пространство разбивается на кубические ячейки размером $d \geq R_{\text{cut}}$. Суммирование ведётся только по атомам, находящимся в ближайших ячейках (рис. 2). Это позволяет уменьшить сложность алгоритма до $O(n)$ при изменении размеров системы при постоянной плотности.

Рассмотрим этапы вычисления сил:

1) Для каждого атома вычисляется индекс ячейки, в которой он находится (пространственный хэш). Сложность этого этапа – $O(n)$. На данном этапе присутствует параллелизм по данным, поэтому он идеален для распараллеливания.
2) Сортировка массива атомов по их пространственному хэшу. Данная операция выполняется на GPU функцией из библиотеки thrust. Сложность $O(n \log(n))$.
3) Поиск минимальных и максимальных индексов атомов, принадлежащих каждой ячейке. Сложность $O(n)$. Выполняется на GPU, поддерживающем операции atomicMin и atomicMax в глобальной памяти, либо на CPU.

4) Нахождение сил с учётом отсечения. Сложность O(n) (при фиксированной плотности атомов), выполняется на GPU. В нашей реализации это самая длительная операция, занимающая до 80% от общего времени.

Поскольку положение атомов не может сильно измениться за один шаг, результаты этапов 1–3 могут быть использованы на нескольких шагах интегрирования.

2. Описание программы

Программа реализована на языках CUDA C и C++. Использовались контейнеры и алгоритмы из библиотеки thrust, в вспомогательных целях использовались библиотеки STL и boost.

Реализовано две версии алгоритма вычисления сил: с использованием spatial indexing и без. Для предотвращения “комбинаторного взрыва”, когда число необходимых подпрограмм вычисления сил равно произведению числа потенциалов на число версий алгоритма, использовалась эмуляция функций высшего порядка с помощью шаблонов C++.

Литература

1. Umeno Y. at el. Optimization of interatomic potential for Si/SiO 2 system based on force matching // Computational materials. 2002. V. 25. P. 447–456.
2. Tersoff J. New Empirical Model for the Structural Properties of Silicon // Physical Review Letters. 1986. V. 56. P. 632–635.
3. Goswami P., Schlegel P., Solenthaler B. Interactive SPH Simulation and Rendering on the GPU // Proceedings of the 2010. P. 1–10.

ИСПОЛЬЗОВАНИЕ РАСШИРЯЕМЫХ ЯЗЫКОВ ДЛЯ ПРОГРАММИРОВАНИЯ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ

А.П. Свиридов

Южно-Уральский госуниверситет, Челябинск

Введение

Одним из наиболее распространенных на сегодняшний день средств создания параллельных приложений является технология OpenMP [2, 6]. Эта технология предоставляет пользователю набор простых, интуитивно понятных директив для «разметки» кода на параллельные и последовательные части. По сути, директивы предоставляют пользователю удобный синтаксис, позволяющий программисту писать более декларативный код. Этот подход к программированию параллельных приложений оказался настолько удобен, что сравнительно недавно анонсированная технология OpenACC полностью его копирует.

Однако в этом подходе присутствует один существенный недостаток. Дело в том, что технология подобная OpenMP не может быть реализована посредством обычной библиотеки. Поддержка данной технологии конкретным компилятором полностью ложится на плечи разработчиков этого компилятора. Это существенно затрудняет процесс ее внедрения и изменения. Так, например, технология OpenACC сейчас поддерживается исключительно тремя коммерческими компиляторами, что по понятным причинам значительно ограничивает круг людей, которые могут ее применять и изучать.

Однако существуют языки программирования, в которых программист может добавлять новые специальные конструкции без изменения компилятора – расширяемые языки.

Расширяемый язык – это такой язык программирования, синтаксис и семантика которого могут быть изменены и расширены для облегчения написания программы, но при этом расширение должно выполняться без изменения компилятора языка. В настоящее время существует небольшое количество расширяемых языков программирования, самыми популярными из которых являются диалекты языка LISP.

По сравнению с традиционными языками программирования, расширяемые языки обладают рядом преимуществ. Строить системы программирования на основе расширяемых языков проще, и они по определению являются открытыми. В рамках расширяемого языка можно строить многоуровневые модели программирования. Программист получает, с одной стороны, больший контроль над используемыми преобразованиями программ, а с другой – возможность программировать на низком уровне в случае необходимости. Вследствие открытости системы программист получает возможность использовать преобразования, полезные только для специфических задач, и которые не имеет смысл реализовывать в самом компиляторе [5].

2. Библиотека CLiMP

В рамках данной работы ведется разработка библиотеки параллельного программирования CLiMP для языка программирования Common Lisp [3, 4], предоставляющей программисту инструменты, аналогичные стандарту OpenMP, но использующие механизм расширений для предоставления нового синтаксиса.

В отличие от технологии OpenMP, в библиотеке CLiMP большое внимание уделяется возможности расширения пользователем существующего функционала. Более того, при разработке библиотеки изначально были созданы инструменты, которые позволяют вносить добавления, а уже с их помощью была реализована большая часть библиотеки.

Так, например, в библиотеке CLiMP предусмотрена команда `define-parallel-command`, предоставляющая пользователю возможность с помощью специальных правил объявлять новые команды параллельного региона. С помощью `define-parallel-command` были реализованы параллельные аналоги циклов, средства синхронизации потоков (критические секции, барьерная синхронизация), средства контроля над потоком управления и др.

Еще одним примером инструмента расширения является команда `define-reduction`. С помощью данной команды пользователь может определить правила распределения некоторой структуры данных между потоками и правила сбора локальных данных обратно в глобальную по отношению к потокам переменную. С помощью данной команды были определены все варианты `reduction`, предоставляемые стандартом OpenMP.

На рис. 1 приведен пример «Hello world» программы с использованием библиотеки CLiMP. Конструкция `parallel` является аналогом директивы `#pragma omp parallel`. С ее помощью пользователь может создать параллельный регион и сконфигурировать его с помощью различных параметров (`number-of-threads`, `reduction`, `private` и др.) и внутренних команд (`single`, `barrier`, `master` и др.).

```
(parallel (:number-of-threads N)
 (format t "~&Hello from: ~a" thread-number)
 (single
  (format t "~&Number-of-threads: ~a" number-of-threads)))
```

Рис. 1. «Hello world» программа с использованием библиотеки CLiMP

Библиотека CLiMP также способствует пошаговому процессу создания параллельной программы. На рис. 2 приведен пример использования стандартного цикла dotimes и его параллельного аналога. В параллельных аналогах циклов доступны все параметры и внутренние команды конструкции parallel.

```
(dotimes (var niterations) => (parallel-dotimes (var
niterations)
  ...) => ...)
```

Рис. 2. Перевод однопоточной версии цикла к многопоточной

2.1. Архитектура библиотеки

Основным механизмом создания расширений является макрос [1]. *Макрос* – это аналог функции, которая исполняется на этапе компиляции (либо до него), принимает на вход фрагменты кода и возвращает новый фрагмент кода. После исполнения макроса возвращаемый фрагмент кода подставляется на место вызова макроса. При этом макросы в языке Common Lisp могут использовать весь функционал, уже реализованный в системе. Они могут использовать как пользовательские, так и стандартные функции, макросы, переменные и константы.

Примером эффективного использования макросов является CLOS (Common Lisp Object System). CLOS – впервые была реализована именно как сторонняя библиотека, однако она предоставляла пользователю полноценный синтаксис для создания объектно-ориентированных приложений.

На рис. 3 представлена общая архитектура библиотеки. Прямоугольниками на схеме изображен функционал, который реализован с помощью функций, а шестигранниками с помощью макросов.

Рис. 3. Архитектура библиотеки CLiMP

Как мы видим, с помощью функции реализована лишь низкоуровневая работа по пересылке данных в пул потоков. Важно, что с помощью подобной функции и набора средств синхронизации (мьютексов, переменных состояния) можно «эмулировать» функционал предоставляемый, например, стандартом OpenMP: используя мьютекс, переменную состояния и счетчик можно реализовать барьерную синхронизацию, зная номер потока, можно организовать цикл со статическим распределением итераций и т. д.

Другими словами, можно вывести набор правил, с помощью которых может быть осуществлен перевод декларативной конструкции (например, директивы OpenMP) в код, реализующий ее функционал.

В библиотеке CLiMP подобный перевод осуществляется в рамках конструкции `parallel`. Важно, что набор таких правил не фиксирован и пользователь может вводить в библиотеку новые высокоуровневые конструкции параллельного программирования.

2.2. Результаты проведения экспериментов

В ходе выполнения работы проведено тестирование производительности компонентов разрабатываемой библиотеки. В качестве одного из экспериментов выбрана задача построения графического представления множества Мандельброта.

Решение данной задачи выполнено в двух вариантах: с использованием языка C (Intel C Compiler 12.1.0) и технологии OpenMP и с использованием языка Common Lisp (Steel Bank Common Lisp 1.0.56) и разработанной библиотеки CLiMP.

Тестирование проводилось на узле суперкомпьютера «СКИФ-Аврора ЮУрГУ». Результаты тестирования приведены в табл. 1 и 2 и на рис. 4 и 5.

Таблица 1. Время выполнения программы в секундах в зависимости от количества потоков

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ICC + OpenMP	13,85	6,98	4,72	3,55	2,97	2,49	2,16	1,91	1,71	1,57	1,42	1,31
SBCL + CLiMP	18,38	9,28	6,25	4,73	3,94	3,32	2,89	2,54	2,28	2,07	1,89	1,75

Таблица 2. Ускорение работы программы в зависимости от количества потоков

	1	2	3	4	5	6	7	8	9	10	11	12
ICC + OpenMP	1,00	1,98	2,93	3,90	4,66	5,56	6,41	7,25	8,10	8,82	9,79	10,61
SBCL + CLiMP	1,00	1,98	2,94	3,89	4,67	5,55	6,37	7,25	8,08	8,88	9,72	10,53

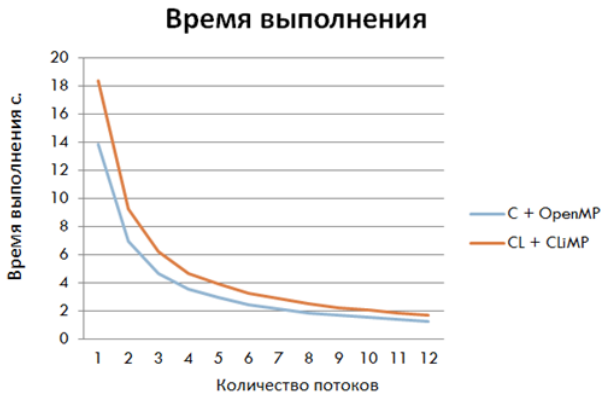


Рис. 4. Время выполнения программы

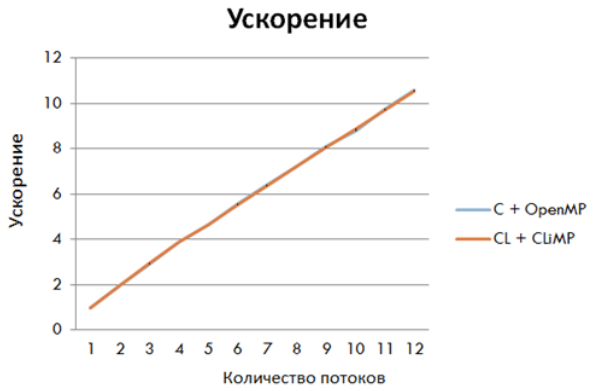


Рис. 5. Ускорение работы программы

Как видно из полученных результатов, несмотря на то, что CLiMP – это просто библиотека, реализованная стандартными средствами языка Common Lisp, она позволяет получать ускорение приблизительно равное ускорению программы, использующей технологию OpenMP.

3. Заключение

В статье рассказано о парадигме расширяемого программирования, ее преимуществах и возможностях использования для программирования параллельных вычислений, описана архитектура библиотеки CLiMP, расширяющей язык Common Lisp для программирования многопоточных вычислений. Разработанная библиотека протестирована на ряде задач, для которых показала увеличение производительности в несколько раз без существенного увеличения объема исходного кода. При этом если бы соответствующий код писался вручную, его объем вырос бы в несколько раз, а читабельность программы значительно снизилась.

Литература

1. Graham P. On Lisp: Advanced Techniques for Common Lisp. Prentice Hall, 1993. 413 p.
2. OpenMP Version 3.1 Complete specification. – [http://www.openmp.org/mp-documents/OpenMP3.1.pdf].
3. Seibel P. Practical Common Lisp. Apress, 2005. 500 p.
4. Steele G. Common LISP. The Language. Second Edition. Digital Press, 1990. 1029 p.
5. Адинец А.В.. Использование расширяемых языков для программирования графических процессоров // Параллельные вычислительные технологии (ПаВТ’2011): Труды Международной научной конференции (Москва, 28 марта–1 апреля 2011 г.). Челябинск: Издательский центр ЮУрГУ, 2011. С. 15–26.
6. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Спб.: БХВ-Петербург, 2002. 606 с.

ИНТЕРАКТИВНЫЙ РЕНДЕРИНГ НА GPU: ВЫЧИСЛЕНИЕ ФУНКЦИИ ВИДИМОСТИ

С.С. Свистунов

Тульский госуниверситет

Тени в компьютерной графике придают реалистичность. Для получения мягких теней необходимо вычислять значения функции видимости, что является затратной операцией. Предлагаются способы расчета функции видимости при помощи GPU с использованием OpenGL и буферов глубины, а также основанные на технологии NVIDIA CUDA.

Введение

Изучается задача интерактивного рендеринга трехмерной сцены, состоящей из модели и удаленного окружающего освещения L. Рассматривается случай первичного освещения, определяемого уравнением освещенности Kajiya [1]

$$B_p(v)=\int_{S^2} L(x)V_p(x)\rho_p(x,v)d\mu(x). \tag{1}$$

Здесь $B_p(v)$ – яркость отраженного освещения из точки модели p в направлении $v\in S^2$; $V_p(x)$– функция видимости; $\rho_p(x,v)$ – BRDF с учтенным ламбертовским множителем $(n_p x)_+$, где n_p – нормаль к поверхности модели в точке p.

Одной из самых затратных операций является вычисление значений функции видимости $V_p(x)$. Это бинарная функция, равная нулю, если луч, выпущенный из точки p в направлении x, пересекает модель, и единице – иначе. Для непрозрачной модели $V_p(x)=0$, если $n_p x<0$.

Учет функции видимости в уравнении (1) дает затенения модели, что приводит к большей реалистичности изображения. Легко посчитать аналитическое выражение функции видимости для шара. Все остальные случаи сложны, поэтому требуются качественные аппроксимации функции видимости. В работе [2] предлагается способ быстрого приближенного вычисления функции видимости в случае, когда модель покрывается шариками. Однако он имеет свои недостатки, в частности, непросто получать подобные покрытия (особенно в интерактивном режиме). Поэтому несомненный интерес представляет быстрый подсчет значений функции видимости для дискретных направлений прямо для полигональной модели (или, возможно, ее низкополигональной аппроксимации).

1. Расчет функции видимости

Основной вклад в затенение модели в точке p дает нулевой сферический коэффициент Фурье функции видимости, тесно связанный с моделью затенения Ambient Occlusion (AO) [3].

В методе AO (также как в интегральной теореме о среднем значении) полагается

$$B_p(v)\approx A_p\int_{S^2} L(x)\rho_p(x,v)d\mu(x), \tag{2}$$

где константа A_p также называется АО в точке p. Отметим, что приближенное вычисление интеграла в формуле (2) уже заметно проще и есть разные способы его аппроксимации, в частности, метод PRT [4]. Нами разработан более общий, быстрый и удобный метод сферических дизайнов SDPRT [5, 6].

Есть несколько способов определения АО. В диффузном случае для ламбертовской поверхности, когда $\rho(x,v)=(n_p x)_+$, константа A_p выбирается из условия точности формулы (2) для константного освещения. В этом случае

$$A_p = 4\int_{S^2} V_p(x)(n_p x)_+ \, d\mu(x).$$

Таким образом, задача нахождения АО сводится к приближенному вычислению интеграла по сфере (на самом деле полусфере из-за множителя $(n_p x)_+$) от негладкой численно определяемой функции. Подобные интегралы вычисляются при помощи кубатурных формул Монте-Карло или, что предпочтительнее, сферических дизайнов [5, 6]. В обоих случаях приближенно имеем

$$A_p = \frac{4}{N}\sum_{k=1}^{N} \lambda_k V_p(x_k)(n_p x_k)_+ , \qquad (3)$$

где λ_k, x_k – соответственно веса и узлы выбранной кубатурной формулы. Для метода Монте-Карло при использовании стандартных генераторов случайных чисел на сфере обычно используется значение $N = 10^4 - 10^5$. Метод дизайнов при схожей точности позволяет использовать кубатуры с $N = 10^2 - 10^3$, что является их несомненным преимуществом.

В итоге приходим к задаче, типичной для рейтрейсинга: в каждой точке модели необходимо найти пересечения пучка лучей с полигонами модели. Прямое решение при помощи метода грубой силы требует $O(V \cdot N \cdot F)$ итераций, где V – число вершин, F – число полигонов, поскольку для вычисления значения $V_p(x_k)$ требуется найти пересечение луча x_k из точки p с моделью, что приводит к необходимости перебора всех полигонов модели. Его выполнение на CPU для моделей с сотнями тысяч вершин и полигонов занимает часы.

Согласно (3) алгоритм расчета сводится к следующим действиям:

1) для каждой вершины p выпустить пучок из N лучей;

2) для каждого луча x_k определить пересечение с моделью и, как следствие, вычислить значение функции видимости $V_p(x_k)$;

3) просуммировать результат с учетом весов и ламбертовского множителя $(n_p x_k)_+$.

Естественно, для ускорения расчетов в первую очередь необходимо воспользоваться ускоряющими структурами, например, BVH-деревьями или пространственным хешированием. Тогда в лучшем случае получаем цикл из $O(V \cdot N)$ итераций при условии, что построение вспомогательных структур выполняется за $O(F)$ итераций и они заполняются равномерно.

Однако оказалось, что гораздо большее ускорение можно получить при использовании возможности современных GPU по параллельным вычислениям: технологии NVIDIA CUDA и OpenCL. Некоторые из использованных нами идей приведены в [7].

2. Способы ускорения расчетов с использование GPU

Первая идея состоит в том, что вычисления функции видимости в разных точках модели не зависят друг от друга и, значит, их можно распараллелить по вершинам мо-

дели, что для метода грубой силы в лучшем случае приводит к циклу из $O(N \cdot F)$ итераций. Вторая идея использует то, что современные видеокарты имеют возможность в заданной точке модели быстро получать буфер глубины по всем пространственным направлениям. Третья идея заключается в том, что пространственное хеширование также можно очень быстро выполнять в параллельном режиме.

Кратко приведем особенности реализаций.

2.1. Использование буферов глубины

Для ускорения расчетов воспользуемся возможностью GPU сохранять буфер глубины. Представим, что в каждой вершине модели расположено шесть камер, ориентированных по нормали и направленных по соответствующим координатным осям. После этого каждая камера «снимает» (рендерит) буфер глубины. Таким образом получаем кубическую карту глубины в данной точке модели. Важно, что эти операции на GPU выполняются очень быстро.

Теперь для поиска пересечений луча с моделью достаточно определить необходимую сторону карты окружения и с помощью простых матричных преобразований вычислить пиксель, отвечающий лучу. После этого по цвету пикселя в текстуре буфера глубины легко определить, было ли пересечение или нет (если цвет не черный, значит пересечение есть). Это процесс схож с известным методом построения теней от точечных источников света – теневые карты (Shadow mapping) [8].

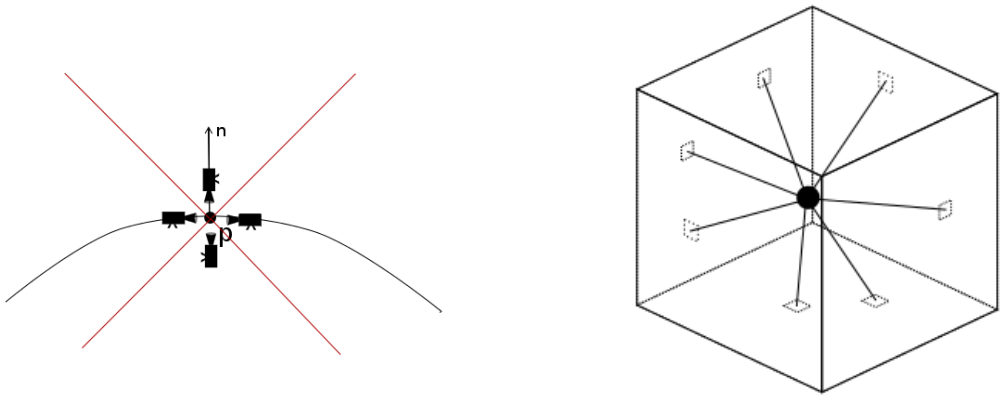


Рис. 1. Организация расчетов с использованием буфера глубины: расположение камер для получения карты окружения (слева); выбор значений из карты окружения (справа)

Такой подход позволяет избежать вложенного цикла по полигонам при поиске пересечений луча с моделью и заменить его созданием карты глубины, что делается очень быстро с помощью стандартных средств OpenGL/DirectX.

2.2. Использование NVIDIA CUDA

Для выполнения расчетов с использованием технологии NVIDIA CUDA применялись различные способы распараллеливания вычислений: по вершинам модели (ядро с одномерной топологией), по вершинам и лучам (ядро с двумерной топологией), по полигонам и лучам (ядро с двумерной топологией). Наиболее удобным и результативным подходом оказалось распараллеливание по вершинам и лучам.

При реализации возникла проблема ограничения максимального времени выполнения одного запуска ядра (около одной секунды). Для обхода этого ограничения вычисления разбивались на несколько этапов: обработка по несколько лучей, обработка частей модели.

Стоит уделять особое внимание работе с различными типами памяти (особенно на бюджетных видеокартах). Самый простой, но наименее эффективный способ реализации – это трансляция алгоритма однопроцессорной версии и использование глобальной памяти видеокарты. Такой подход дает ускорение в расчетах, но простая замена глобальной памяти на текстурную позволяет в разы улучшить производительность.

Предпринимались попытки использования различных типов памяти видеокарты – от глобальной и текстурной до реализации управляемого кэша в разделяемой памяти и побитового хранения результатов вычислений. При использовании разделяемой памяти в ней сохранялась информация о полигонах, что позволило (как и положено) снизить количество чтений из глобальной памяти и ускорить расчет.

Однако наибольшей производительности удалось достичь при использовании разделяемой памяти и битового формата хранения значений функции видимости.

3. Результаты

Тестовая сцена состояла примерно из пяти тысяч полигонов и вершин. Для нее вычисление функции видимости (AO) заняло приблизительно 0.8 сек (расчеты велись на бюджетной видеокарте Nvidia GTX 260):

Метод	Время расчета AO
CPU	более 1 часа
Буферы глубины	около 11 сек
NVIDIA CUDA	около 0.8 сек

По ссылке http://goo.gl/AYNk7 можно скачать видео, иллюстрирующее процесс визуализации.

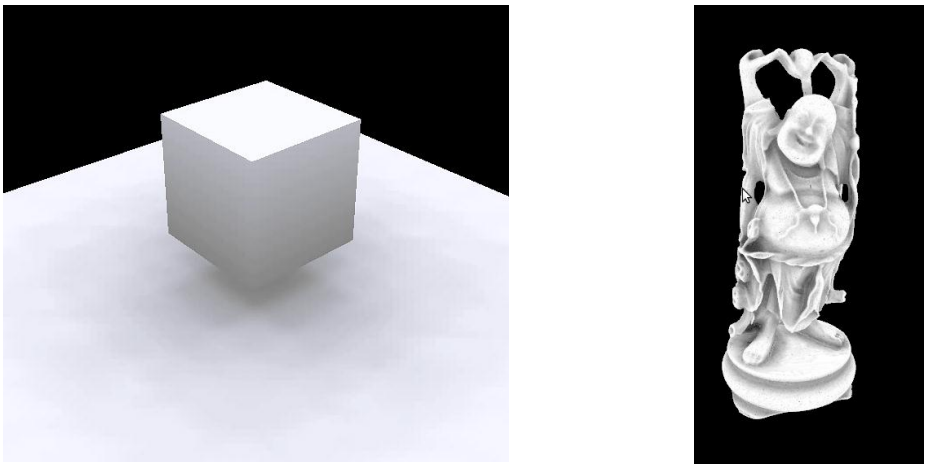


Рис. 2. Результат расчета функции видимости: расчет на тестовой сцене (слева); расчет на модели buddha [9] (справа)

Литература

1. Kajiya J.T. The rendering equation, Computer Graphics (Proceedings of SIGGRAPH 1986), 1986, 20 (4), 143–150.
2. Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng and Baining Guo. Real-time Soft Shadows in Dynamic

Scenes using Spherical Harmonic Exponentiation // ACM Transaction on Graphics. Proceedings of SIGGRAPH . 2006. V. 25, N 3. P. 977–986.
3. A real time self ambient occlusion method from Nvidia's GPU Gems 2 book. http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch14.pdf
4. Green R. Spherical harmonic lighting: The gritty details // In Game Developers' Conference. 2003.
5. Свистунов С.С. Интерактивный рендеринг при помощи сферических дизайнов (SDPRT) для низкочастотного окружающего освещения и BRDF Фонга // Изв. ТулГУ. Сер. Естественные науки. 2011. Вып. 1. С. 188–199.
6. Горбачев Д.В., Иванов В.И., Странковский С.А. Моделирование освещения в интерактивной графике при помощи сферических дизайнов // Изв. ТулГУ. Сер. Естественные науки. 2007. Т. 1, Вып. 1. С. 17–36.
7. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: ДМК-Пресс, 2010.
8. Gary King, Shadow Mapping Algorithms – [ftp://download.nvidia.com/developer/presentations/2004/GPU_Jackpot/Shadow_Mapping.pdf].
9. The Stanford 3D Scanning Repository – [http://graphics.stanford.edu/data/3Dscanrep/].

ВЫЧИСЛЕНИЯ ВЫСОКОЙ ТОЧНОСТИ НА КОМПЬЮТЕРЕ НОВОГО ТИПА

Я.Д. Сергеев[1,2], Д.Е. Квасов[2]

[1]Калабрийский университет, Ренде, Италия
[2]Нижегородский госуниверситет им. Н.И. Лобачевского

Предлагается новая позиционная система с бесконечным основанием, позволяющая записывать бесконечно большие и бесконечно малые величины в явной форме конечным числом символов. Обсуждаются вопросы построения новых высокоточных численных методов на основе данной системы с использованием нового типа суперкомпьютеров – компьютера бесконечности.

Введение

Вопросам, связанным с бесконечностью и бесконечно малыми величинами посвящена обширная литература (см., например, [3, 4, 11, 13]). Эти вопросы, как правило, затрагивают проблемы, напрямую связанные с основаниями математики. Те ответы на них, которые считаются удовлетворительными в каждый конкретный исторический период, активно используются как в формировании фундамента математики этого времени, так и при решении прикладных задач. Именно использование понятий бесконечно большого и бесконечно малого в численных методах и изучение способов их эффективной реализации на высокопроизводительных компьютерах, а также расширение границ их применения является основной задачей настоящей работы.

1. Новая система счисления для представления конечных, бесконечно больших и бесконечно малых величин

Современная точка зрения на бесконечность во многом определяется идеями Г. Кантора, который показал, что существуют бесконечные множества, имеющие разную мощность (см. [4]). Современные системы записи конечных чисел позволяют проводить вычисления с высокой точностью с конечными величинами. При работе с бесконечными числами возникают трудности. Существующие системы счисления не позволяют нам численно (не символьно) работать с бесконечными и бесконечно малыми величинами на компьютере с использованием тех же формальных правил, что и при работе с конечными числами. Среди причин, которые не позволяют нам организовать такую работу, можно выделить, как минимум, следующие: существование неопределенных форм (например, $\infty-\infty$, ∞/∞, $0*\infty$ и т.д.), невыполнение для бесконечности аристотелевского принципа «часть меньше целого» (действительно, для любого конечного z следует $z + 1 > z$, тогда как $\infty+1=\infty$) и невозможность хранения бесконечного числа знаков в конечной памяти компьютера.

В работах [14, 15, 18, 20, 21] предлагается новая позиционная система записи с бесконечной базой, позволяющая выражать бесконечно большие и бесконечно малые величины в явной форме конечным числом символов. При этом вводится новый нумерал ①, называемый *гросс-единица* (от англ. “grossone”, «большая единица»), определяемый как число элементов в множестве натуральных чисел, а символ ∞ (а также $\aleph_0$, $\aleph_1$, ω) исключается из числа используемых нумералов (под нумералом понимается сим-

вол или группа символов, используемых для представления числа). Некая величина A в новой позиционной системе записи представляется в виде:

$$A = a_{k_1} ①^{k_1} a_{k_2} ①^{k_2} \ldots a_{k_K} ①^{k_K} = \sum_{i=1}^{K} a_{k_i} ①^{k_i}, \text{ где } k_1 > k_2 > \ldots > k_{K-1} > k_K. \quad (1)$$

В записи (1) конечные нумералы a_{k_i}, $1 \le i \le K$, называются *гросс-цифрами* и могут быть как положительными, так и отрицательными; величины k_i, $1 \le i \le K$, называются *гросс-степенями* и могут быть конечными, бесконечными и бесконечно малыми; сама же величина A называется *гросс-нумералом*. Таким образом, конечные величины a составляют частный случай записи (1), а именно, $a = a①^0$. Правила выполнения арифметических операций с нумералами типа (1) подробно описаны в [5, 14–17].

Предлагаемая методология позволяет вычислить у определенных бесконечных множеств число их элементов (см. [14, 15, 21]), соблюдая принцип «часть меньше целого». Например, легко определить, что количество нечетных натуральных чисел равно ①/2, также как и количество четных натуральных чисел, число целых чисел равно 2①+1 и т.д. Если из множества целых чисел исключить ноль, то в нем останется 2① элементов. При этом следует ①/2 < ① < 2① < 2①+1. Аналогично, можно вычислить число элементов с точностью до одного элемента у определенных множеств континуальной мощности и показать, например, что $2^{①} < 3^{①} - 1 < 3^{①} < 3^{①} + 1 < ①^{①}$. При этом новый подход не противоречит теориям Г. Кантора (см. [4]) и нестандартного анализа А. Робинсона (см. [3, 13]), а дополняет их, постоянно отслеживая разницу между числами и нумералами и изучая возможность практического выполнения операций (см. [7, 12, 20, 21]).

Тем самым, данный физически ориентированный подход к численным расчетам позволяет построить простую и наглядную арифметику для практической работы не только с конечными числами, но и с бесконечно большими и бесконечно малыми величинами. При этом наблюдаемое в окружающем нас мире свойство «часть всегда меньше целого», используемое при работе с конечными числами, естественно распространяется на бесконечно большие и бесконечно малые величины.

2. Компьютер бесконечности

При решении многих прикладных задач, в частности, численного анализа математических моделей, негладкой и глобальной оптимизации, аппроксимации и т.д. возникают условия, при которых требуется исследовать математические модели в их предельных состояниях и проводить вычислительные эксперименты с супервысокой точностью. Поскольку традиционные процессоры не позволяют проводить вычисления с бесконечными и бесконечно малыми числами, эти требования приводят к необходимости прекращения вычислений на компьютерах и привлечении экспертов для решения этих проблем путем создания новых математических моделей и т.п.

Рассмотренная система записи чисел с бесконечным основанием открывает новое перспективное направление в вычислительной математике на базе «компьютера бесконечности», позволяющего автоматически производить численные (не символьные) расчеты с бесконечными, бесконечно малыми, конечными числами, а также с бесконечными числами с конечными и бесконечно малыми частями (см. [5, 8, 16, 17]). В рамках этого направления могут быть развиты принципиально новые вычислительные модели и алгоритмы с использованием исчисления бесконечно малых и бесконечно больших величин для нового поколения аппаратно-программных компьютерных платформ, обладающих экстремальными характеристиками по производительности и точности. При этом повышение точности вычислительных операций осуществляется не за счет количественного увеличения вычислительной мощи компьютеров, а за счет применения ка-

чественно новых форматов данных и позиционной системы счисления с основанием гросс-единица.

Существует программный прототип компьютера бесконечности, использующий специально разработанную (на языках C++, C#) библиотеку классов и методов (см. [8]), которая дополняет существующие типы данных новым типом *Grossnumber* для операций с гросс-нумералами. Библиотека позволяет создавать объекты класса *Grossnumber*, соответствующие нумералам (1), и взаимодействовать с ними путем вызова специально разработанных методов, эффективно реализующих основные математические операции и дополнительные функции для расширения возможностей использования гросс-нумералов (см. [5, 8, 16, 17]).

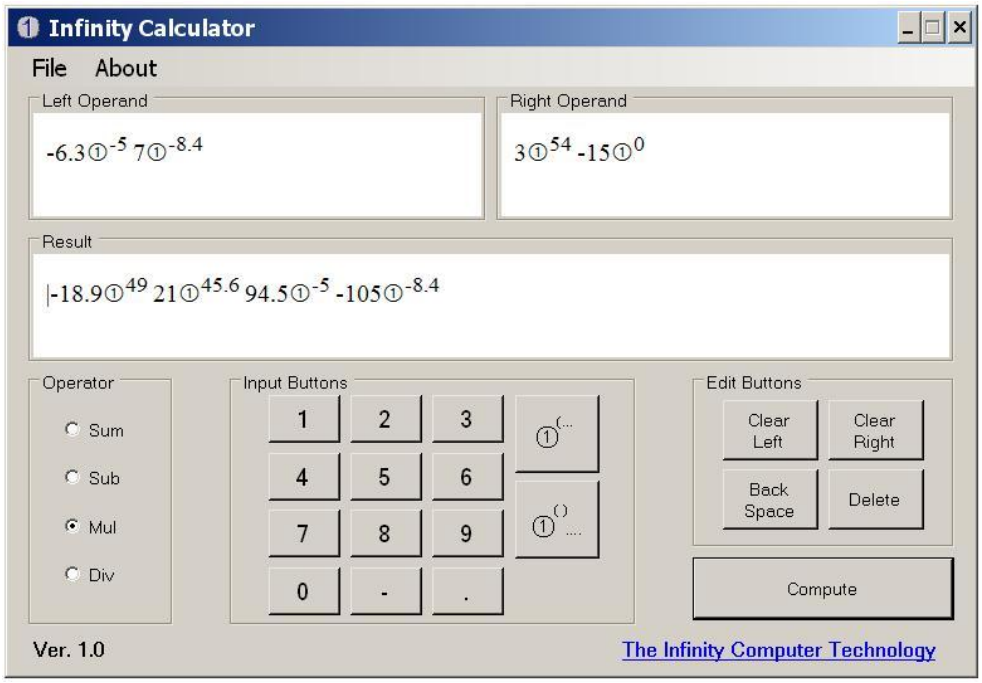


Рис. 1. Интерфейс калькулятора бесконечности (с примером умножения двух гросс-нумералов)

С целью демонстрации работы компьютера бесконечности (см. [23]), в качестве его программного прототипа предлагается так называемый «калькулятор бесконечности», который позволяет осуществлять арифметические операции с гросс-нумералами. На рис. 1 приведен интерфейс калькулятора: в данном примере произведено умножение двух гросс-нумералов, A_{left} = -6.3①$^{-5}$ 7①$^{-8.4}$ и A_{right} = 3①^{54} -15①^{0}, с результатом операции, A_{result} = -18.9①^{49} 21①$^{45.6}$ 94.5①$^{-5}$ -105①$^{-8.4}$. Как видно из записи операндов и результата, гросс-нумерал A_{left} представляет бесконечно малую величину (все гросс-степени конечны и отрицательны), A_{right} – бесконечно большую величину (в которой присутствует также и конечная часть, -15①^{0} = -15), а результат A_{result} состоит как из бесконечно больших (-18.9①^{49} 21①$^{45.6}$), так и бесконечно малых (94.5①$^{-5}$ -105①$^{-8.4}$) гросс-нумералов.

3. Примеры прикладных задач

Предлагаемый подход позволяет дать детальные ответы на серию классических вопросов и парадоксов, имеющих дело с бесконечно большими и бесконечно малыми величинами. Например, с новых позиций могут быть рассмотрены машины Тьюринга (см. [22]) и первая проблема Гильберта (см. [18]). В работах [9, 10, 12, 19, 21, 24, 25] обсуждаются границы применимости нового подхода и рассматриваются приложения (численный анализ, теория вероятности, оптимизация, фракталы и др.).

В качестве простого, но эффектного примера рассмотрим задачу условной оптимизации. Как известно (см., например, [1, 2, 6]), одним из методов ее решения является использование штрафных функций с последующим анализом условий регулярности. При этом возникает проблема подбора штрафных параметров, которая не всегда может быть удовлетворительно решена. Оказывается (см. [10]), использование ①-нумералов позволяет избежать неинтуитивного итерационного выбора коэффициентов штрафных функций, автоматизируя решение задач безусловной оптимизации.

Пусть, например, требуется найти решение следующей простой задачи:

$$\min\,(1/2\,x_1^2 + 1/6\,x_2^2),\qquad x = (x_1,x_2):\ x_1 + x_2 = 1. \tag{2}$$

Используя метод штрафных функций (см., например, [2]), эта задача может быть решена путем (безусловной) минимизации некоей новой функции, например:

$$\min\,(1/2\,x_1^2 + 1/6\,x_2^2) - 1/(2\varepsilon)(1 - x_1 - x_2)^2, \tag{3}$$

где выбор штрафного параметра ε осуществляется либо путем его последовательного уменьшения, либо априорно. И в том, и в другом случаях имеются известные трудности с определением данного коэффициента. Заменяя же в (3) величину 1/ε на ① и используя необходимые условия экстремума

$$x_1 - ①(1-x_1-x_2) = 0 \text{ и } 1/3x_2 - ①(1-x_1-x_2) = 0,$$

получаем следующую стационарную точку минимизируемой функции:

$$x^*_1 = 1①/(1+4①),\qquad x^*_2 = 3①/(1+4①). \tag{4}$$

Легко заметить, что значения x^*_1, x^*_2 из (4) могут быть записаны также как:

$$x^*_1 = 1/4 - ①^{-1}(1/16 - 1/64①^{-1}\ \ldots),\qquad x^*_2 = 3/4 - ①^{-1}(3/16 - 3/64①^{-1}\ \ldots),$$

где в записях гросс-нумералов x^*_1, x^*_2 можно выделить конечные части ($\bar{x}_1$=1/4 и $\bar{x}_2$=3/4, соответственно) и бесконечно малые (гросс-нумералы с отрицательными гросс-степенями). Оставляя в этих записях лишь конечные части, автоматически получаем решение $\bar{x}_1$=1/4, $\bar{x}_2$=3/4 исходной задачи (2) без привлечения процедуры подбора штрафного параметра в (3).

Подобная техника может быть использована не только при решении задач безусловной оптимизации, но и во многих других областях, включая методы решения систем линейных и дифференциальных уравнений.

Работа выполнена при финансовой поддержке Минобрнауки России, государственное соглашение о предоставлении гранта № 14.B37.21.0878.

Литература

1. Баркалов К.А. Оценки эффективности параллельного индексного метода глобальной оптимизации // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия “Математическое моделирование. Оптимальное управление”. – 2011. – Т. 3, № 2. – С. 13–19.
2. Гергель В.П., Стронгин Р.Г., Городецкий С.Ю., Гришагин В.А., Маркина М.В. Современные методы принятия оптимальных решений. – Н. Новгород: Изд-во Нижегородского госуниверситета, 2002. – 190 с.
3. Дэвис М. Прикладной нестандартный анализ. – М.: Мир, 1980. – 236 с.
4. Кантор Г. Труды по теории множеств. – М.: Наука, 1988. – 431 с.
5. Сергеев Я.Д. Компьютерная система для хранения бесконечных, бесконечно малых и конечных величин и выполнения с ними арифметических операций: Патент РФ № 2395111. – 20.07.2010.

6. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008. – 352 с.
7. Сочков А.Л. Философские аспекты новейшей арифметики бесконечности // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия “Социальные науки”. – 2009. – Т. 3, № 15. – С. 72–77.
8. Brugnano L., Consegni L., Kvasov D.E., Sergeyev Ya.D. On implementation aspects of the Infinity Computer // Proceedings of the International Workshop “Infinity-2010: Infinite and Infinitesimal in Mathematics, Computing and Natural Sciences”, Cetraro (Italy), May 17–21, 2010. – DEIS, Università della Calabria: Rende (CS). – P. 19.
9. D’Alotto L. Cellular automata using infinite computations // Applied Mathematics and Computation. – 2012. – Vol. 218, № 16. – P. 8077–8082.
10. De Cosmis S., De Leone R. The use of grossone in Mathematical Programming and Operations Research // Applied Mathematics and Computation. – 2012. – Vol. 218, № 16. – P. 8029–8038.
11. Lolli G. Infinitesimals and infinites in the history of mathematics: A brief survey // Applied Mathematics and Computation. – 2012. – Vol. 218, № 16. – P. 7979–7988.
12. Margenstern M. Using Grossone to count the number of elements of infinite sets and the connection with bijections // p-Adic Numbers, Ultrametric Analysis and Applications. – 2011. – Vol. 3, № 3. – P. 196–204.
13. Robinson A. Non-standard analysis. – Princeton University Press, 1996. – 308 p.
14. Sergeyev Ya.D. Arithmetic of infinity. – CS: Edizioni Orizzonti Meridionali, 2003. – 112 p.
15. Sergeyev Ya.D. A new applied approach for executing computations with infinite and infinitesimal quantities // Informatica. – 2008. – Vol. 19, № 4. – P. 567–596.
16. Sergeyev Ya.D. Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them: EU patent 1728149. – 03.06.2009.
17. Sergeyev Ya.D. Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them: USA patent 7,860,914. – 28.12.2010.
18. Sergeyev Ya.D. Counting systems and the First Hilbert problem // Nonlinear Analysis Series A: Theory, Methods & Applications. – 2010. – Vol. 72, № 3-4. – P. 1701–1708.
19. Sergeyev Ya.D. Higher order numerical differentiation on the Infinity Computer // Optimization Letters. – 2011. – Vol. 5, № 4. – P. 575–585.
20. Sergeyev Ya.D. Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals // Rendiconti del Seminario Matematico dell'Università e del Politecnico di Torino. – 2010. – Vol. 68, № 2. – P. 95–113.
21. Sergeyev Ya.D. Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains // Nonlinear Analysis Series A: Theory, Methods & Applications. – 2009. – Vol. 71, № 12. – P. e1688–e1707.
22. Sergeyev Ya.D., Garro A. Observability of Turing Machines: A refinement of the theory of computation // Informatica. – 2010. – Vol. 21, № 3. – P. 425–454.
23. The Infinity Computer web page, http://www.theinfinitycomputer.com
24. Vita M.C., De Bartolo S., Fallico C., Veltri M. Usage of infinitesimals in the Menger’s Sponge model of porosity // Applied Mathematics and Computation. – 2012. – Vol. 218, № 16. – P. 8187–8195.
25. Zhigljavsky A. Computing sums of conditionally convergent and divergent series using the concept of grossone // Applied Mathematics and Computation. – 2012. – Vol. 218, No 16. – P. 8064–8076.

РАЗРАБОТКА МЕТОДОВ И ЯЗЫКОВЫХ СРЕДСТВ СБОРОЧНОГО ПРОГРАММИРОВАНИЯ ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕД

И.А. Сидоров, Е.И. Поздняк

Институт динамики систем и теории управления СО РАН, Иркутск

Рост потребности в высокопроизводительных вычислительных системах для организации ресурсоемких научно-исследовательских экспериментов повысил качественные требования к системному и прикладному программному обеспечению таких систем. Однако обилие технологий для организации высокопроизводительных вычислений не уменьшает числа проблем при использовании различных программно-аппаратных архитектур и требует от предметного специалиста все больших навыков технического характера для их эффективного использования. Кроме того различные подходы к организации вычислений, используемые методологии параллельного программирования, а также методы оптимизации программ порой требуют существенной модификации реализующих алгоритмов, что зачастую является труднорешаемой задачей.

Однако существует широкий класс ресурсоемких задач, для решения которых не требуется существенной модификации реализующих алгоритмов и их адаптации к применению на высокопроизводительных вычислительных системах. Такие задачи могут характеризоваться необходимостью проведения многовариантных расчетов над полем независимых между собой входных данных различными программами для их обработки. Одним из подходов к организации такого рода вычислений является построение программ на основе парадигмы сборочного программирования [1].

При таком подходе к созданию программ для параллельных и распределенных вычислительных сред основные усилия будут направлены не на непосредственное кодирование, а на подбор, настройку и организацию совместного существования совокупности позаимствованных (возможно, из различных источников) многократно используемых компонентов. При этом эффективность от использования высокопроизводительных ресурсов может достигаться за счет применения концепции крупноблочного параллелизма [2].

Для создания программ такого рода и их применения в распределенных вычислительных средах авторами разрабатывается программный комплекс DISCOMP [3]. Вычислительные модули, составляющие функциональное наполнение, представляют собой исполняемые программы, которые могут быть реализованы на различных языках программирования (например, C, Fortran, Pascal и др.) и быть платформо-зависимыми. Удаленный запуск модулей, обмен данными между модулями через файлы и мониторинг узлов ВС реализуются средствами системной части ИК DISCOMP. Языковые средства предоставляют возможности для описания концептуальной модели предметной области, ее объектов и основных свойств, а также непосредственно каркаса распределенной программы. В общем случае, благодаря гибкости и универсальности языковых средств, становится возможным расширить класс допустимых задач, более эффективно использовать ресурсы РВС и повысить надежность вычислений.

В данной работе описываются методы и языковые средства сборочного программирования для распределенных вычислительных сред, в основе информационного ядра

которых используется простая вычислительная модель, а в качестве управляющей структуры - сеть Петри. Приводятся примеры условных и циклических конструкций, конструкций fork/join, методов wait и continue, многометодных технологий и др.

Литература

1. Горбунов-Посадов М. Расширяемые программы. М.: Полиптих, 1999. 336 с.
2. Малышкин В.Э. Мощности вычислительных систем прирастают с параллелизмом // Наука в Сибири. 2001. № 32-33. 3 с.
3. Сидоров И.А., Опарин Г.А., Феоктистов А.Г. Технология организации распределенных вычислений в инструментальном комплексе DISCOMP // Современные технологии. Системный анализ. Моделирование. – 2009. – № 2. – С. 175-180.

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОГО ДВУХУРОВНЕВОГО ИНДЕКСНОГО АЛГОРИТМА ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

С.В. Сидоров

Нижегородский госуниверситет им. Н.И. Лобачевского

Данная работа продолжает развитие параллельных алгоритмов минимизации многоэкстремальных функций при невыпуклых ограничениях, построенных на базе индексного метода. Редукция многомерных задач к одномерным основана на приближенном фрактальном отображении единичного отрезка вещественной оси на гиперкуб (развертки типа кривых Пеано). В одном из параллельных алгоритмов используется схема построения множества кривых Пеано («вращаемые развертки»), которую можно эффективно применять при решении задачи на кластере с десятками и сотнями процессоров. Другой подход к распараллеливанию ориентирован на вычислительные узлы с общей памятью. Двухуровневый параллельный индексный алгоритм глобального поиска объединяет эти подходы, учитывая иерархическую структуру современных высокопроизводительных кластерных систем.

1. Постановка задачи глобальной оптимизации

Рассмотрим задачу глобальной оптимизации вида

$$\varphi^*=\varphi(y^*)=\min\{\varphi(y): y\in D,\ g_j(y)\le 0,\ 1\le j\le m\},$$
$$D=\{y\in R^N: a_i\le y_i\le b_i,\ 1\le i\le N\}, \qquad (1)$$

где целевая функция $\varphi(y)$ (в дальнейшем обозначаемая также $g_{m+1}(y)$) и левые части ограничений $g_j(y), 1\le j\le m$, удовлетворяют условию Липшица с соответствующими константами $L_j, 1\le j\le m+1$, а именно

$$|g_j(y_1)-g_j(y_2)|\le L_j|y_1-y_2|,\ 1\le j\le m+1,\ y_1,y_2\in D.$$

Используя кривые типа развертки Пеано $y(x)$, однозначно отображающие отрезок [0,1] на N-мерный гиперкуб P

$$P=\{y\in R^N: -2^{-1}\le y_i\le 2^{-1},\ 1\le i\le N\}=\{y(x): 0\le x\le 1\},$$

исходную задачу можно редуцировать к следующей одномерной задаче:

$$\varphi(y(x^*))=\min\{\varphi(y(x)): x\in[0,1],\ g_j(y(x))\le 0,\ 1\le j\le m\}. \qquad (2)$$

Рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией и липшицевыми ограничениями одномерную задачу, в которой соответствующие функции удовлетворяют равномерному условию Гельдера (см. [2]), т.е.

$$|g_j(y(x'))-g_j(y(x''))|\le K_j|x'-x''|^{1/N},\ x',x''\in[0,1],$$
$$1\le j\le m+1,$$

где N есть размерность исходной многомерной задачи, а коэффициенты K_j связаны с константами Липшица L_j исходной задачи соотношениями $K_j\le 4L_j\sqrt{N}$.

Различные варианты индексного алгоритма для решения одномерных задач и соответствующая теория сходимости представлены в работах [1, 4, 7].

2. Вращаемые развертки и параллельный метод для кластера

Редукция многомерных задач к одномерным с помощью разверток имеет такие важные свойства, как непрерывность и сохранение равномерной ограниченности разностей функций при ограниченности вариации аргумента. Однако при этом происходит потеря части информации о близости точек в многомерном пространстве, так как точка $x \in [0,1]$ имеет лишь левых и правых соседей, а соответствующая ей точка $y(x) \in R^N$ имеет соседей по 2^N направлениям. А при использовании отображений типа кривой Пеано близким в N-мерном пространстве образам y', y'' могут соответствовать достаточно далекие прообразы x', x'' на отрезке [0, 1]. Как результат, единственной точке глобального минимума в многомерной задаче соответствует несколько (не более 2^N) локальных экстремумов в одномерной задаче, что, естественно, ухудшает свойства одномерной задачи.

Сохранить часть информации о близости точек позволяет использование множества отображений

$$Y_L(x)=\{y^1(x),\ldots, y^L(x)\} \qquad (3)$$

вместо применения единственной кривой Пеано $y(x)$ (см. [3, 5]). Каждая кривая Пеано $y^i(x)$ из $Y_L(x)$ может быть получена в результате поворота исходной кривой. При этом найдется отображение $y^i(x)$, которое точкам многомерного пространства y', y'', которым при исходном отображении соответствовали достаточно далекие прообразы на отрезке [0, 1], будет сопоставлять более близкие прообразы x', x''.

Максимальное число различных поворотов развертки, отображающей N-мерный гиперкуб на одномерный отрезок, составляет 2^N. Использование всех из них является избыточным, требуется выбрать лишь часть из всех возможных вариантов. В предложенной схеме преобразование развертки осуществляется в виде поворота на угол $\pm\pi/2$ в каждой из координатных плоскостей. Число подобных пар поворотов определяется числом координатных плоскостей пространства, которое равно $C_N^2 = \frac{N(N-1)}{2}$, а общее число преобразований будет равно $N(N-1)$. Учитывая исходное отображение, приходим к заключению, что данный способ позволяет строить до $N(N-1)+1$ развертки для отображения N-мерной области на соответствующие одномерные отрезки. В случае необходимости данный способ построения множества отображений может быть легко «отмасштабирован» для получения большего (вплоть до 2^N) числа разверток.

Использование множественных отображений (3) позволяет решать задачу (1) путем параллельного решения L задач вида (2) на наборе отрезков $[0, 1]$. Каждая одномерная задача решается на отдельном процессоре с использованием развертки y^s, $1 \le s \le L$. Результаты испытания в точке x^k, полученные конкретным процессором для решаемой им задачи, интерпретируются как результаты испытаний во всех остальных задачах (в соответствующих точках $x^{k1}, \ldots, x^{kL}$) и рассылаются другим процессорам. При таком подходе испытание в точке $x^k \in [0, 1]$, осуществляемое в s-й задаче, состоит в последовательности действий:

1. Определить образ $y^k = y^s(x^k)$ при соответствии $y^s(x)$.
2. Проинформировать остальные процессоры о начале проведения испытания в точке y^k (*блокирование* точки y^k).
3. Вычислить величины $g_1(y^k),\ldots,g_\nu(y^k)$, где значения индекса $\nu \le m$ определяются условиями

$$g_j(y^k) \le 0,\ 1 \le j < \nu,\ g_\nu(y^k) > 0,\ \nu \le m.$$

Выявление первого нарушенного ограничения прерывает испытание в точке y^k. В случае, когда точка y^k допустима, испытание включает вычисление значений всех

функционалов задачи, при этом значение индекса принимается равным величине $\nu=m+1$. Тройка

$$y_s(x^k), \nu=\nu(x^k), z^k=g_\nu(y^s(x^k)) \tag{4}$$

является результатом испытания в точке x^k.

4. Определить прообразы $x^{kl}\in[0,1]$, $1\le l\le L$,точки y^k, и интерпретировать испытание, проведенное в точке $y^k\in D$, как проведение испытаний в L точках

$$x^{k1},\dots,x^{kl}, \tag{5}$$

с одинаковыми результатами

$$\nu(x^{k1})=\dots=\nu(x^{kL})=\nu(x^k),$$
$$g_\nu(y^1(x^{k1}))=\dots=g_\nu(y^L(x^{kL}))=z^k.$$

Проинформировать остальные процессоры о результатах испытания в точке y^k, разослав им тройки (y^k, ν, z^k).

Каждый процессор имеет свою копию программных средств, реализующих вычисление функций задачи, и решающее правило алгоритма, а также собственное хранилище поисковой информации. Для организации взаимодействия на каждом процессоре создается очередь, в которую процессоры помещают информацию о выполненных итерациях в виде троек: точка очередной итерации, индекс и значение из (4), причем индекс заблокированной точки полагается равным –1, а значение функции в ней не определено.

3. Параллельный метод для вычислительного узла с общей памятью

Второй подход к распараллеливанию эксплуатирует характеристическую схему индексного алгоритма. В редуцированной одномерной задаче на очередной итерации для каждого интервала (x_{i-1},x_i),$1\le i\le k+1$, вычисляется характеристика R(i), где

$$R(i)=\begin{cases}\Delta_i+\dfrac{(z_i-z_{i-1})^2}{r_\nu^2\mu_\nu^2\Delta_i}-2\dfrac{(z_i+z_{i-1}-2z_\nu^*)}{r_\nu\mu_\nu}, & \nu=\nu(x_{i-1})=\nu(x_i)\\[2ex] 2\Delta_i-4\dfrac{(z_i-z_\nu^*)}{r_\nu\mu_\nu}, & \nu=\nu(x_i)>\nu(x_{i-1})\\[2ex] 2\Delta_i-4\dfrac{(z_{i-1}-z_\nu^*)}{r_\nu\mu_\nu(l,\tau)}, & \nu=\nu(x_{i-1})>\nu(x_i)\end{cases}$$
$$\Delta_i=(x_i-x_{i-1})^{1/N}.$$

Величины $r_\nu>1$, $1\le\nu\le m+1$, являются параметрами алгоритма.

Далее определяется s (s – как правило, число вычислительных ядер на узле кластера) интервалов $(x_t, x_{t+1})_k, 1\le k\le s$, которым соответствует s наибольших характеристик

$$R_k(t)=\max\{R(i): 1\le i\le k+1\}, 1\le k\le s. \tag{6}$$

На s вычислительных ядрах параллельно проводится очередное испытание в серединной точке интервала $(x_t, x_{t+1})_k, 1\le k\le s$, если индексы его концевых точек не совпадают, т.е.

$$x^{q+1}=\frac{(x_t+x_{t-1})_k}{2},\qquad \nu(x_{t-1})\ne\nu(x_t), 1\le k\le s.$$

В противном случае провести испытание в точке

$$x^{q+1}=\frac{(x_t+x_{t-1})_k}{2}-\mathrm{sign}(z_t-z_{t-1})_k\frac{1}{2r_\nu}\left[\frac{|z_t-z_{t-1}|_k}{\mu_\nu}\right]^N,$$
$$\nu(x_{t-1})=\nu(x_t), 1\le k\le s.$$

Результаты испытания заносятся в общее хранилище поисковой информации.

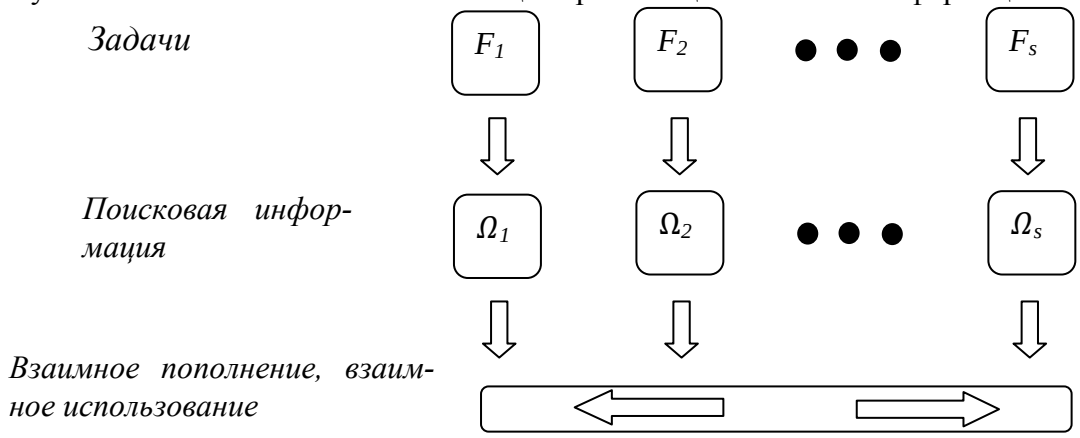


Рис 1. Информационная связанность подзадач оптимизации на узле с общей памятью

4. Двухуровневый параллельный алгоритм

Объединение двух описанных подходов позволяет получить гибридный алгоритм, эффективно использующий распределение поисковой информации по узлам кластера и общую память в рамках одного узла.

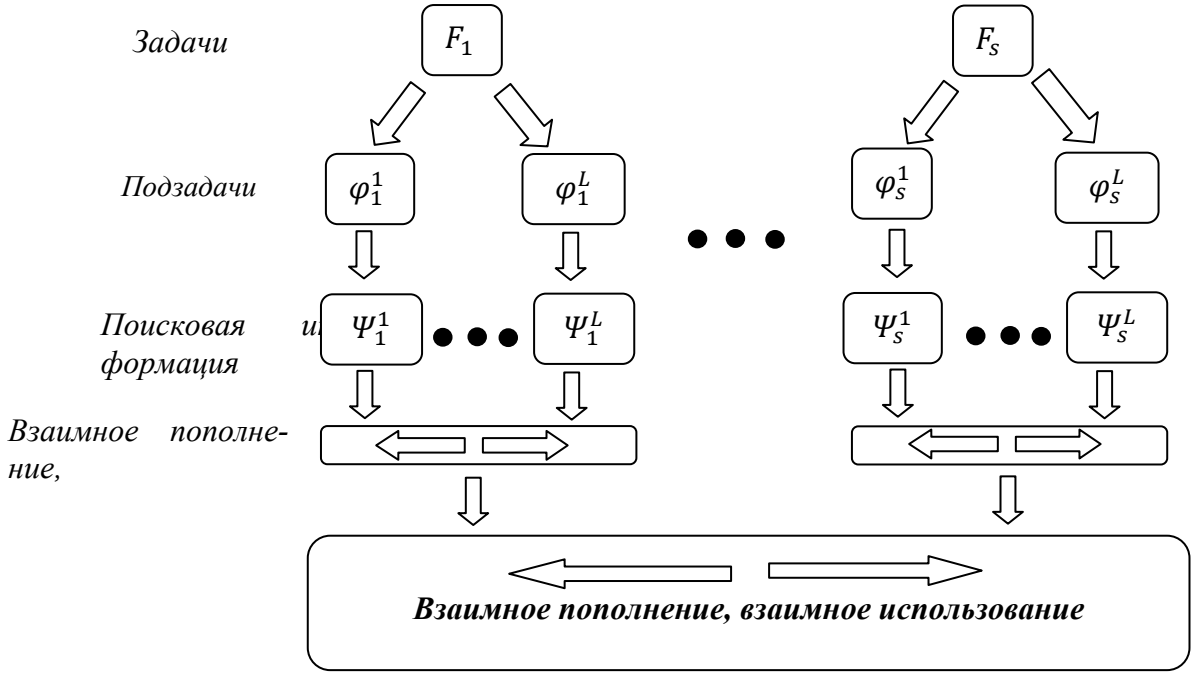


Рис 2. 2-х уровневая модель информационная связанности подзадач

Заключение

Данная работа исследует двухуровневую схему параллельных вычислений, позволяющую исключить передачу данных на вычислительном узле с общей памятью. Вся вычислительная система асинхронно решает информационно связанные подзадачи для каждой разверки (по одной на узел кластера). Проведено исследование эффективности данного алгоритма. Результаты вычислений приводятся в докладе.

Работа выполнена при поддержке Минобрнауки России (соглашение 14.B37.21.0878).

Литература

1. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. (Информационно-статистические алгоритмы). М.: Наука, 1978.
2. Стронгин Р.Г. Поиск глобального оптимума. М.: Знание, 1990.
3. Стронгин Р.Г. Параллельная многоэкстремальная оптимизация с использованием множества разверток // Ж. вычисл. матем. и матем. физ. 1991, Т.31, №8. С. 1173–1185.
4. Стронгин Р.Г., Баркалов К.А. О сходимости индексного алгоритма в задачах условной оптимизации с ε-резервированными решениями // Математические вопросы кибернетики. М.: Наука, 1999. С. 273 – 288.
5. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000.
6. Баркалов К.А., Стронгин Р.Г. Метод глобальной оптимизации с адаптивным порядком проверки ограничений // Ж. вычисл. матем. и матем. физ. 2002, Т.42, №9. С. 1338–1350.
7. Баркалов К.А. Ускорение сходимости в задачах условной глобальной оптимизации. Нижний Новгород: Изд-во Нижегородского гос. ун-та, 2005.
8. Баркалов К.А., Рябов В.В., Сидоров С.В. Использование кривых Пеано в параллельной глобальной оптимизации // Материалы Девятой международной конференции-семинара «Высокопроизводительные параллельные вычисления на кластерных системах», Владимир, 2009. С. 44–47.
9. Стронгин Р.Г., Гергель В.П., Баркалов К.А. Параллельные методы решения задач глобальной оптимизации // Известия вузов. Приборостроение. – 2009, Т. 52, № 10. С. 25–32.
10. Баркалов К.А., Рябов В.В., Сидоров С.В. Масштабируемые параллельные алгоритмы глобальной оптимизации со смешанной локально-глобальной стратегией // Материалы международной научной конференции «Параллельные вычислительные технологии 2010», Уфа, 2010. С. 402–409.
11. Стронгин Р.Г., Маркин Д.Л. Минимизация многоэкстремальных функций при невыпуклых ограничениях // Кибернетика. 1986. №4.
12. Törn A. and Zilinskas A. Global Optimization. Lecture Notes in Computer Science, No 350, Springer-Verlag, Berlin, 1989.

СИГНАЛЬНЫЕ ПРОЦЕССЫ В НЕЙРОННЫХ СЕТЯХ МОЗГА. ВИРТУАЛЬНОЕ МОДЕЛИРОВАНИЕ ДЛЯ ЗАДАЧ ДИАГНОСТИКИ ПАТОЛОГИЙ И ТЕСТИРОВАНИЯ ЛЕКАРСТВ

А.Ю. Симонов[1], В.И. Миронов[1], И.С. Прокин[1], С.Ю. Гордлеева[2], И.В. Мухина[3], В.Б. Казанцев[2]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]Институт прикладной физики РАН, Нижний Новгород
[3]Нижегородская государственная медицинская академия

Моделирование процессов генерации и распространения электрических и химических сигналов в нейрональных сетях мозга является в настоящее время одним из актуальных приложений современных суперкомпьютерных технологий. В работе мы рассматриваем различные аспекты такого моделирования. В частности, процессы биоэлектрической активности диссоциированных культур нейронов гиппокампа могут быть эффективно исследованы в виртуальных моделях спайковых нейронных сетей, связанных между собой посредством синаптических контактов. Объединяемые в большие модельные сети, виртуальные клетки способны демонстрировать эффекты сетевой сигнализации, наблюдаемые экспериментально. Здесь принципиальными моментами являются исследование механизмов генерации и распространения характерных паттернов нейросетевой активности, параметрических условий существования ключевых режимов генерации, механизмов переключения между такими режимами и соответствующих им структурно-функциональные перестроек.

Другая задача связана с компьютерной реконструкцией роста нейрональной сети согласно уравнениям диффузионного транспорта строительных белков и навигации конуса роста дендритных отростков и аксона согласно распределению факторов роста. Ветвление конуса роста описывается в рамках стохастической феноменологической модели. Процесс формирования отростка является зависимым от многих факторов, имеющих как внутриклеточную природу, так и поступающих из межклеточного пространства, которые и являются причиной столь широкого разнообразия типов нейрональных клеток. Процесс удлинения отростка зависит от величины концентрации активных веществ (белков роста), продуцируемых в соме клетки. Пространственное ориентирование отростка осуществляется посредством специальных сигнальных молекул (факторов роста), продуцируемых нейронами, которые, диффундируя сквозь межклеточное пространство, формируют управляющее поле, воздействующее на конус роста нейрита. Применяя данную модель можно конструировать виртуальные сети нейронов с реалистичной морфологией и тестировать различные условия развития нейронных систем в онтогенезе.

Другой тип клеток в мозге, астроциты, а также внеклеточный матрикс мозга, составляющий перинейрональные сети тоже могут принимать активное участие в передаче сигналов между нейронами и влиять на процессы обработки информации и формирования высших когнитивных функций. Например, экспериментально установлено, что кальциевые сигналы в астроцитах связаны с выбросом во внеклеточное пространство особых химических веществ, которые могут воздействовать на пре- и постсинаптические рецепторы, изменяя их функции. Данное направление также является перспектив-

ным в части математического моделирования и создания виртуальных моделей астроцитарных и астроцит-нейронных сетей.

Важной задачей здесь, в том числе и с точки зрения приложений параллельных вычислений, является обработка больших объёмов экспериментальных данных. Примером таких данных являются многоканальные записи электрической активности диссоциированных культур нейронов гиппокампа, выращиваемых на мультиэлектродных зондах. Сигнал с такой культуры регистрируется множеством планарных электродов, расположенных на дне пробирки, в которой выращиваются нейроны. С увеличением пространственного разрешения такой технологии возрастает потребность в вычислительных мощностях, необходимых для обработки и анализа данных. Помимо этого существует ряд задач, в которых требуется обработка потока многоканальных данных в режиме реального времени, например, при формировании моторных команд для управления внешним электронно-механическим устройством (роботом). Такие культуры являются одной из наиболее перспективных экспериментальных моделей для исследования патологических процессов, лежащих в основе многих нейродегенеративных заболеваний. Кроме того данный подход может быть использован для задач тестирования воздействия лекарственных препаратов на функциональное состояние нервной ткани.

В докладе будут изложены основные современные представления в перечисленных областях науки, представлены недавние результаты группы соавторов и сформулированы наиболее интересные и перспективные задачи в этих областях для привлечения высокопроизводительных ресурсов современных суперкомпьютерных технологий.

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ВЫДЕЛЕНИИ ЦЕЛЕВЫХ РЕГИОНОВ НА МУЛЬТИСПЕКТРАЛЬНЫХ КОСМИЧЕСКИХ СНИМКАХ

А.И. Смирнова

Российский университет дружбы народов, Москва

Рассмотрен математический аппарат для обработки мультиспектральных снимков. Приведены примеры классификации методом «закраски». Рассмотрена схема применения параллельных вычислений в обработке мультиспектральных снимков.

Введение

В данной работе рассмотрена задача распознавания объектов и областей на мультиспектральных снимках, полученных в процессе дистанционного зондирования Земли. Задача анализа космических снимков в настоящее время широко распространена. Существуют различные системы, в которых реализованы алгоритмы обработки и тематической классификации космических снимков.

В работе рассмотрен алгоритм тематической классификации изображений, основанный на обобщенной метрике. Обработка мультиспектральных снимков требует больших вычислительных мощностей. В связи с этим возникает задача распараллеливания вычислений с целью ускорения процесса обработки и классификации изображений.

Особенности космических снимков

Космические снимки представляют собой изображения земной поверхности, полученные в результате дистанционного зондирования Земли (ДЗЗ). Они отличаются друг от друга в зависимости от масштаба, обзорности, пространственного разрешения и количества спектральных каналов.

Мультиспектральные снимки получают с различных околоземных спутников. В данной работе используются снимки со спутника TERRA, сделанные с помощью камеры ASTER(Advanced Spaceborne Thermal Emission and Reflector Radiometer).

Снимки представлены 15 каналами, характеристики которых даны в табл. 1.

Таблица 1. Технические характеристики сканера ASTER

№ канала	***Пространственное разрешение, м***	***Спектральные диапазоны, мкм***	***Полоса обзора, км***
1	15	0.51 - 0.60	60
2	15	0.63 - 0.69	
3	15	0.76 - 0.86	
Стерео	15	0.76 - 0.86	
4	30	1.60 - 1.70	
5	30	2.145 - 2.185	
6	30	2.185 - 2.225	
7	30	2.235 - 2.285	
8	30	2.295 - 2.365	

9	30	2.360 - 2.430	
10	90	8.125 - 8.475	
11	90	8.475 - 8.825	
12	90	8.925 - 9.275	
13	90	10.25 - 10.95	
14	90	10.95 - 11.65	

Тематическая классификация мультиспектральных снимков

Под тематической классификацией космических снимков понимают разделение всех пикселей изображения на отдельные классы, каждому из которых соответствуют различные объекты или области. Каждый пиксель мультиспектрального изображения характеризуется набором яркостей, полученных в разных спектрах. Анализ данных показателей в совокупности и по группам позволяет выделять на снимках объекты и области.

Классификация может быть параметрической и непараметрической. При параметрической классификации считается, что значение яркостей в различных спектральных каналах распределено по нормальному закону. Могут использоваться другие статистические характеристики. При непараметрической классификации распределения яркостей не учитывается. Также различают классификацию с обучением (контролируемая) и без обучения (неконтролируемая). В данной работе рассмотрена непараметрическая классификация с обучением.

Математическая модель

Существуют различные алгоритмы тематической классификации космических снимков. Среди них способ расстояния Евклидова и способ «Расстояние Махаланобиса». Они основаны на соответствующих метриках. Рассмотрим каждую из них.

Метрика Евклида

$$R_E^2(x_1, x_2) = (x_1 - x_2)^T \cdot (x_1 - x_2)$$

позволяет определить расстояние между двумя точками. В случае работы с мультиспектральными снимками роль точки играет пиксель. Каждый пиксель имеет n значений яркости, где n - количество спектров. Таким образом, каждый пиксель представляет собой n-мерный вектор. Кроме того считается, что заранее выделено несколько эталонных областей – классов пикселей. Задача классификации – отнести каждый пиксель к одному из таких классов.

Метрика Евклида не позволяет учитывать распределение точек в эталонном, заранее выделенном классе, поэтому ее использование при классификации может привести к появлению ошибок. В связи с этим используется другая метрика – метрика Махаланобиса:

$$R_M^2(x, Y) = (x - \overline{y})^T C^{-1} (x - \overline{y}).$$

Здесь x – n-мерный вектор, соответствующий одному пикселю, Y – эталонный класс, $\overline{y}$ – центр класса Y, C – матрица ковариации для класса Y.

С помощью метрики Махаланобиса можно измерить расстояние от точки до центра класса. Данная метрика учитывает распределение пикселей в эталонных классах. Однако может возникнуть ситуация, когда матрица C необратима. Это возможно, если значения соответствующих признаков разных пикселей одного класса близки друг к другу. В таком случае данная метрика неприменима.

В работе [1] рассматривается обобщенная метрика Евклида–Махаланобиса,

$$R_G^2(x,Y) = (x-\overline{y})^T A^{-1}(x-\overline{y}),$$

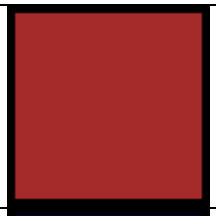
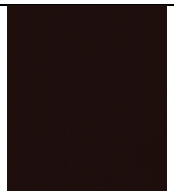
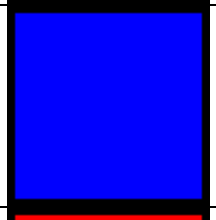
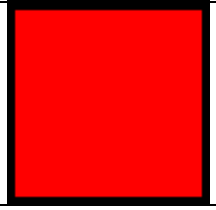
которая учитывает распределение точек в заданном классе, и которая применима при малых значениях дисперсий, входящих в ковариационную матрицу. Здесь x – вектор, соответствующий определенному пикселю, Y – эталонный класс точек, $\overline{y}$ – центр класса, $A = C + E$, C – ковариационная матрица класса Y, E – единичная матрица. Именно наличие единичной матрицы обеспечивает обратимость матрицы A и дает возможность применять данную метрику в более широком классе ситуаций. В работе [1] более подробно рассмотрены свойства обобщенной метрики и приведены примеры ее применения.

В данной работе метрика Евклида-Махаланобиса является основным математическим инструментом тематической классификации мультиспектральных снимков.

Результаты тематической классификации

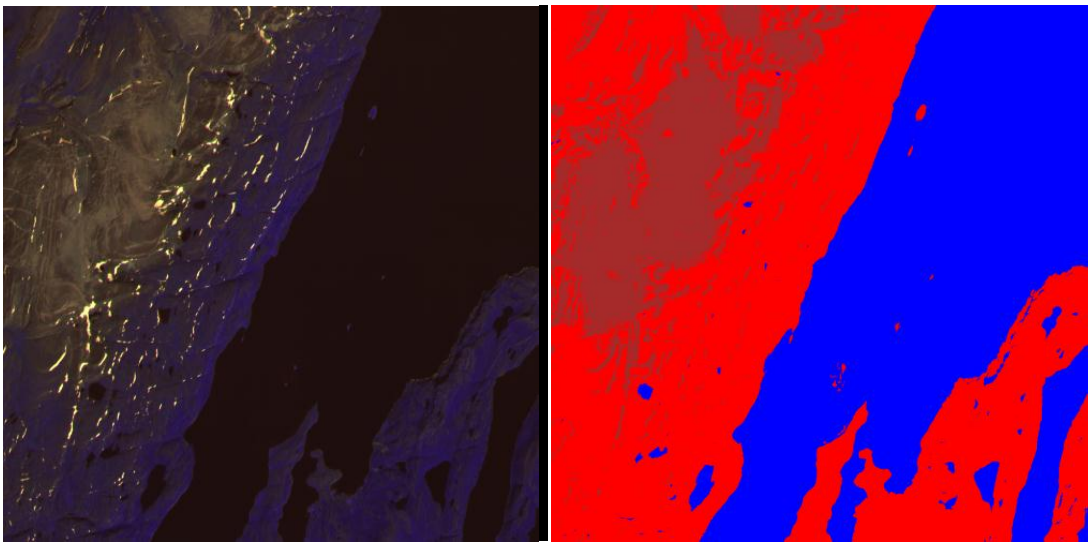
Приведем результаты тематической классификации снимков со спутников TERRA, полученных из базы [2]. В приведенном ниже примере классификация проводилась с учетом яркостей каналов 1, 2, 3. Для просмотра исходного снимка использовалась цветовая система RGB: Red – 1, Green – 2, Blue – 3. В табл. 2 приведены названия выбранных областей, соответствующие им участки на исходном снимке, цвет, присваиваемый пикселям данного класса при классификации и матрица ковариаций. Ниже на рис. 1 представлены исходный снимок и снимок, полученный в результате классификации.

Таблица 2. Назначение цветов экспертами

Название	Исходная область	Назначенный цвет	Матрица ковариации
Земля 1			93.9088 71.99 50.2181 71.99 74.0846 41.309 50.2181 41.309 38.5354
Вода			0.46609 -0.169965 -1.32732 -0.169965 0.545563 1.75268 -1.32732 1.75268 10.5845
Земля 2			77.5992 58.7464 19.0058 58.7464 65.9987 7.73242 19.0058 7.73242 134.27

Использование параллельных вычислений в тематической классификации мультиспектральных снимков

Мультиспектральные космические снимки при хранении занимают большие объемы памяти, а их обработка и классификация связана с использованием больших вычислительных мощностей. При таких условиях целесообразно использование параллельных вычислений для ускорения процесса классификации.



а) б)

Рис. 1. Исходный снимок и результат тематической классификации

При распараллеливании вычислительного процесса важно правильно разделить исходную задачу на подзадачи. В условиях тематической классификации возможен параллельный расчет характеристик каждого класса и разбиение изображения на части для ускорения процесса классификации. Такое распараллеливание значительно сократит время выполнения задачи, что является важным фактором при обработке больших массивов космических мультиспектральных снимков. Важным также является правильное согласование мощностей и пропускных способностей вычислительных кластеров.

Программный комплекс, реализующий тематическую классификацию с закрашиванием областей, состоит двух блоков: блок визуализации исходных и классифицированных изображений, написанный на языке C# и расчетный блок, написанный на языке C++. Средства языка C++ дают возможность модифицировать программу, введя параллельные процессы, что ускорит процесс решения задачи.

Литература

1. Амелькин С.А., Хачумов В.М. Обобщенное расстояние Евклида-Махаланобиса и его свойства. – Информационные технологии и вычислительные системы. № 4, 2006. С. 40–44.
2. База мультиспектральных снимков ASTER – [http://gis-lab.info/projects/aster/index.html].
3. Амелькин С.А., Хачумов В.М. Обобщенное расстояние Евклида -Махаланобиса и его применение в задачах распознавания образов // Доклады 12-ой Всероссийской конференции «Математические методы распознавания образов». – М.: МАКС Пресс, 2005. С. 7–9.

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ РАСПРЕДЕЛЕННЫХ РАСЧЕТОВ В СУПЕРКОМПЬЮТЕРНЫХ СРЕДАХ

С.И. Соболев

Научно-исследовательский вычислительный центр МГУ имени М.В. Ломоносова

Представлен подход к анализу эффективности расчетов в распределенных средах, построенных на основе суперкомпьютерных систем с использованием технологии X-Com. Подход может применяться при исследовании выполнения сверхбольших задач, для которых ресурсов одного суперкомпьютера оказывается недостаточно. Подход также позволяет получить статистическую информацию об особенностях прохождения заданий определенного класса в очередях суперкомпьютеров и выявить возможные проблемы распределения заданий по узлам вычислительных систем.

Введение

В статье описываются результаты, полученные в рамках объединения двух направлений исследования методов организации распределенных расчетов с помощью системы метакомпьютинга X-Com [1, 2]. В 2010 г. система X-Com была адаптирована к работе совместно с основными системами управления потоками заданий суперкомпьютерных систем [3]. В 2011 г. было начато исследование и формирование системы оценок и характеристик распределенных неоднородных вычислительных сред с помощью инструментария на базе X-Com. Был рассмотрен простейший (базовый) случай организации двухуровневой распределенной среды (один сервер X-Com – множество клиентов X-Com, непосредственно работающих на вычислительных узлах) [4]. В данной работе приводятся результаты исследований свойств распределенных сред и расчетов в более сложном случае, когда задания от сервера X-Com поступают в системы очередей суперкомпьютеров через промежуточные серверы X-Com. Целью работы является обобщение и расширение методики оценки распределенных систем, полученной для простейшей архитектуры распределенных сред, на архитектуру, включающую промежуточные серверы X-Com и системы очередей, а также разработка инструментария для получения и анализа свойств распределенных систем.

1. Особенности многоуровневой архитектуры распределенной среды

Промежуточные серверы X-Com [2] работают с центральным сервером по тому же протоколу, что и клиенты X-Com, поэтому с точки зрения центрального сервера они ничем не отличаются от клиентов. Однако промежуточные серверы скрывают всю инфраструктуру, находящуюся под ними, будь то множество клиентов X-Com или система очередей. Если характеристики узла, на котором работает клиент X-Com – тип и частота процессора, объем оперативной памяти и т.д. – не меняются в течение расчета, то набор вычислительных ресурсов, координируемых промежуточным сервером, очевидно, может изменяться. Согласно протоколу взаимодействия клиентов и серверов X-Com, клиенты сообщают серверу характеристики вычислительных узлов, на которых они запущены, только при первом обращении. Параметры, передаваемые центральному серверу в качестве «характеристик узла», задаются в настройках промежуточного сервера перед началом его работы, они могут служить только в качестве примерного опи-

сания нижележащей вычислительной среды, так как не учитывают ее возможную динамику.

Еще одна проблема связана с наличием дополнительных накладных расходов при работе с системами очередей через промежуточные серверы. В базовой архитектуре основной источник задержек между отправкой порции данных клиенту на узел и получением результата их обработки – передача данных по сети (накладные расходы работы самого клиента достаточно малы). В многоуровневой архитектуре добавляется время ожидания порции во входном буфере промежуточного сервера, время ожидания задания в очереди и время ожидания готовой порции в выходном буфере (подробнее механизм буферизации описан в [2]). Эти параметры тесно связаны с настройками системы очередей и загруженностью вычислительной системы. Причем, если настройки системы очередей (максимальное число одновременно выполняющихся заданий и максимально допустимое число заданий в очереди) обычно прописаны в политиках доступа к вычислительному комплексу и известны заранее, то загрузка комплекса является, вообще говоря, динамическим фактором. В соответствии с политиками доступа настраиваются размеры буферов и другие параметры модуля взаимодействия промежуточного сервера X-Com с системой очередей на конкретной вычислительной системе.

Невозможным оказывается напрямую использовать показатели, связанные с производительностью (*MAXpeak, INTpeak, AVGpeak*), поскольку они опираются только на данные, полученные из настроек промежуточных серверов. Коммуникационная эффективность расчета *Ecomm* окажется довольно низкой, поскольку она учитывает неизбежные накладные расходы на ожидание заданий в очереди и порций данных во входных/выходных буферах. Из-за буферизации также снизится комплексная эффективность *Ecomplex*. Для того, чтобы разобраться в структуре такого расчета, протоколов только центрального сервера оказывается недостаточно. Необходим анализ событий всех промежуточных серверов, участвовавших в расчете. Для реализации этого функциональность промежуточных серверов X-Com была расширена возможностью протоколирования своих действий в лог-файлы. Для анализа таких файлов был разработан специальный инструментарий.

2. Вычислительный эксперимент

Проиллюстрируем исследование характеристик распределенного расчета на примере решения модельной задачи в распределенной среде, состоящей из 4-х суперкомпьютеров. Характеристики задействованных вычислительных ресурсов приведены в таблице 1.

Таблица 1. Список суперкомпьютеров, составлявших основу распределенной вычислительной среды

Суперкомпьютер	**Вычислительные узлы**	**СУПЗ**	**Очередь**	**Число заданий в очереди**
«Ломоносов» (МГУ, Москва)	2x Intel Xeon 5570 2.93 ГГц, 12 Гб	Slurm	regular4	20
			test	20
«Чебышев» (МГУ, Москва)	2x Intel Xeon E5472 3.0 ГГц, 8 Гб	Cleo	regular	10
			test	10
СКИФ Урал (ЮУрГУ, Челябинск)	2x Intel Xeon E5472 3.0 ГГц, 8 Гб	Torque	batch (по умолчанию)	20
СКИФ Cyberia (ТГУ, Томск)	2x Intel Xeon 5150 2.66 ГГц, 8 Гб	Torque	batch (по умолчанию)	20

Параметры модельной задачи были подобраны так, чтобы время обработки одной порции составляло не более 10 минут (при общем числе порций 5000). Это позволило

использовать тестовые очереди суперкомпьютеров МГУ в дополнение к основным очередям. На головных машинах этих суперкомпьютеров было запущено по два промежуточных сервера X-Com, каждый из которых работал со своей очередью. На суперкомпьютерах СКИФ Урал и СКИФ Cyberia использовались только очереди по умолчанию, на их головных машинах работало по одному промежуточному серверу X-Com. Центральный сервер X-Com располагался на головной машине суперкомпьютера «Чебышев».

Рассмотрим характеристики [4], полученные в результате анализа протокола работы центрального сервера. Расчет продолжался 4 часа 39 минут. Ускорение *Speedup* достигло значения 70.5 – это можно считать хорошим результатом с учетом того, что в рамках задачи одновременно работало не более 100 процессов. Коэффициент ускорения *Cspeedup* составил 11.75, но в данном случае это значение нельзя считать корректным, поскольку с точки зрения центрального сервера над расчетом работало всего 6 процессов. Комплексная эффективность *Ecomplex* составила 95.92% – снова хороший результат: размеры входных и выходных буферов промежуточных серверов были малы по сравнению с общим числом порций в задаче, поэтому избыточные обмены и перерасчеты оказались незначительными на общем фоне. А вот коммуникационная эффективность *Ecomm*, как и предполагалось, оказалась достаточно низкой – 35.07%. Эта величина отражает временн***ы***е затраты на организацию всего расчета. Чтобы разобраться в структуре этих накладных расходов и причинах низкого уровня этого показателя, необходим анализ потоков событий каждого из промежуточных серверов. Для этого построим графики на основе табличных файлов, генерируемых анализаторами лог-файлов серверов X-Com.

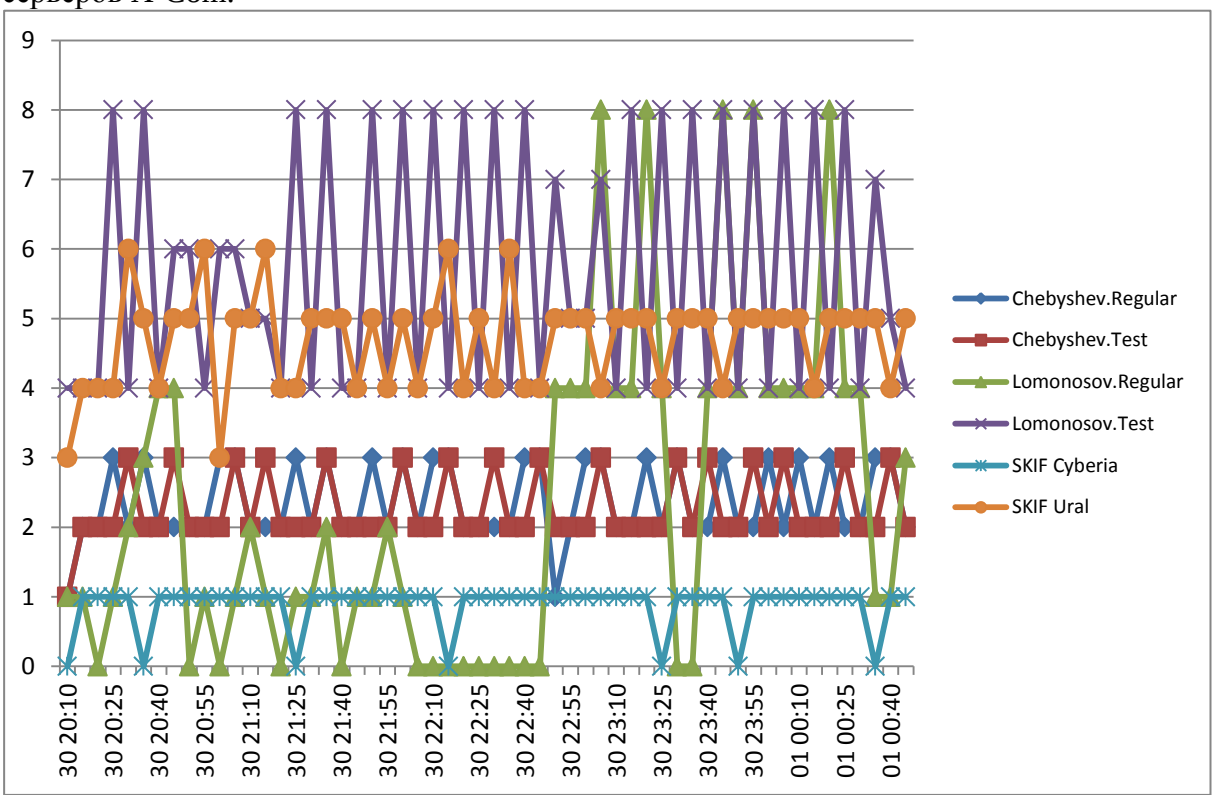


Рис. 1. График интенсивности отправки результатов центральному серверу

Рисунок 1 демонстрирует интенсивность получения центральным сервером результатов обработки порций данных от промежуточных серверов. По вертикальной оси отложено число запросов типа ASW, в каждом из которых передавалось по 5 результирующих порций.

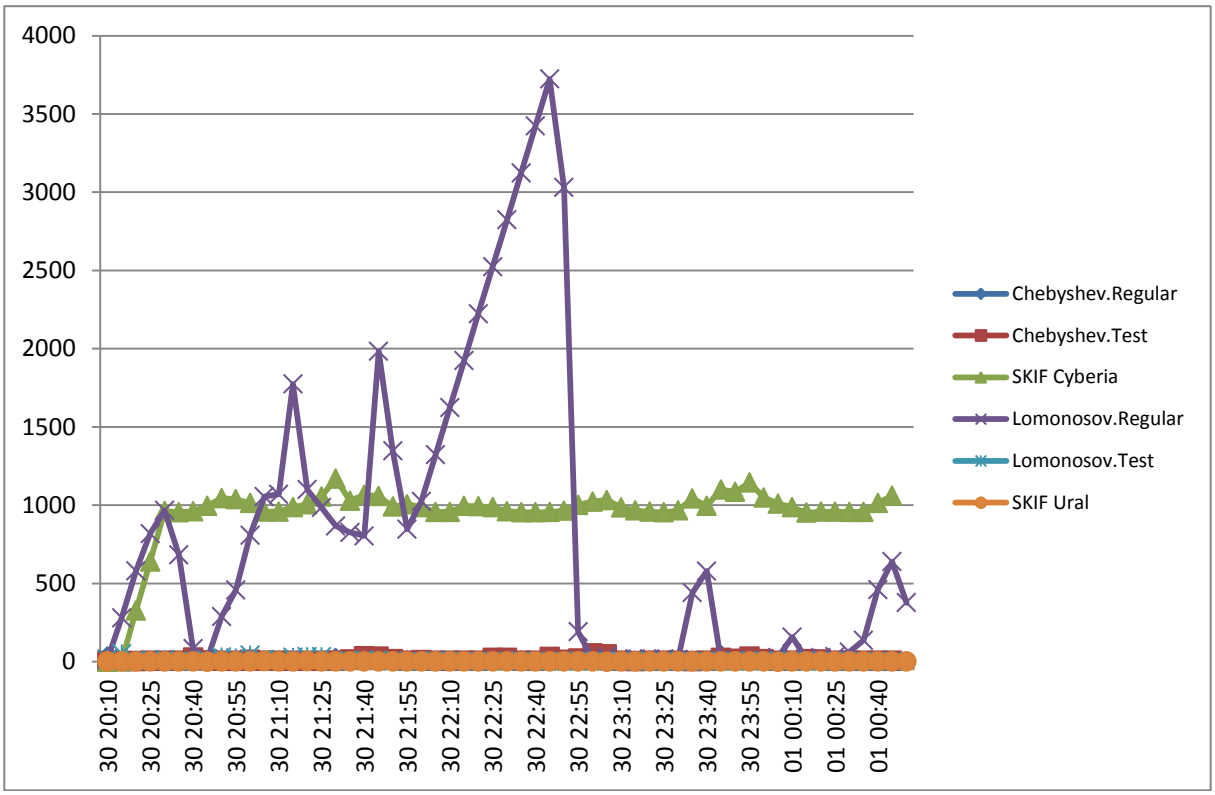


Рис. 2. Среднее время ожидания заданий в очереди

Среднее время ожидания заданий в очередях суперкомпьютеров приведено на рис. 2 (значения по вертикальной оси отложены в секундах). Хорошо видна постоянная и однородная загрузка очереди суперкомпьютера СКИФ Cyberia, а также интенсивное использование основной очереди «Ломоносова». Относительно небольшая загрузка была зафиксирована у тестовых очередей суперкомпьютеров МГУ, основной очереди «Чебышева», а также у суперкомпьютера СКИФ Урал. Эти данные хорошо коррелируют с оценками комплексной эффективности *Ecomplex*, полученными отдельно для каждой очереди: чем больше среднее время ожидания, тем ниже эти значения. Для «свободных» очередей оно составляло 44-48%, для «занятых» – 12-19%. Отметим, что понятия «загрузка», «свободно» и «занято» мы используем здесь неформально; их нельзя трактовать в широком смысле и обобщать на другие временн***ы***е интервалы, однако при описании ситуации в очередях на заданный момент времени для задач того же класса, что и модельная, они имеют вполне адекватную интерпретацию.

Интересный эффект был выявлен при анализе времен выполнения заданий. Поскольку с вычислительной точки зрения все задания были одинаковые, предполагалось получить одинаковые значения для каждого суперкомпьютера. Но это оказалось верно только для «Ломоносова». На остальных суперкомпьютерах время для большей части заданий было одинаковым и минимальным, времена же остальных укладывались в кластеры, которые и видны на графике (рис. 3) выше основных линий (по горизонтальной оси графика отложен номер порции, по вертикальной – время счета задания в секундах). Вероятно, такие задания попадали на узлы одновременно с другими заданиями. Такая ситуация может быть штатной (политики очереди могут допускать группировку нескольких однопроцессорных заданий на одном узле) или же ошибочной, свидетельствующей о некорректной работе механизма распределения заданий. На графике также заметно некоторое количество заданий с нулевым временем счета. Эти задания были завершены сразу после постановки на счет; причина такого поведения выясняется.

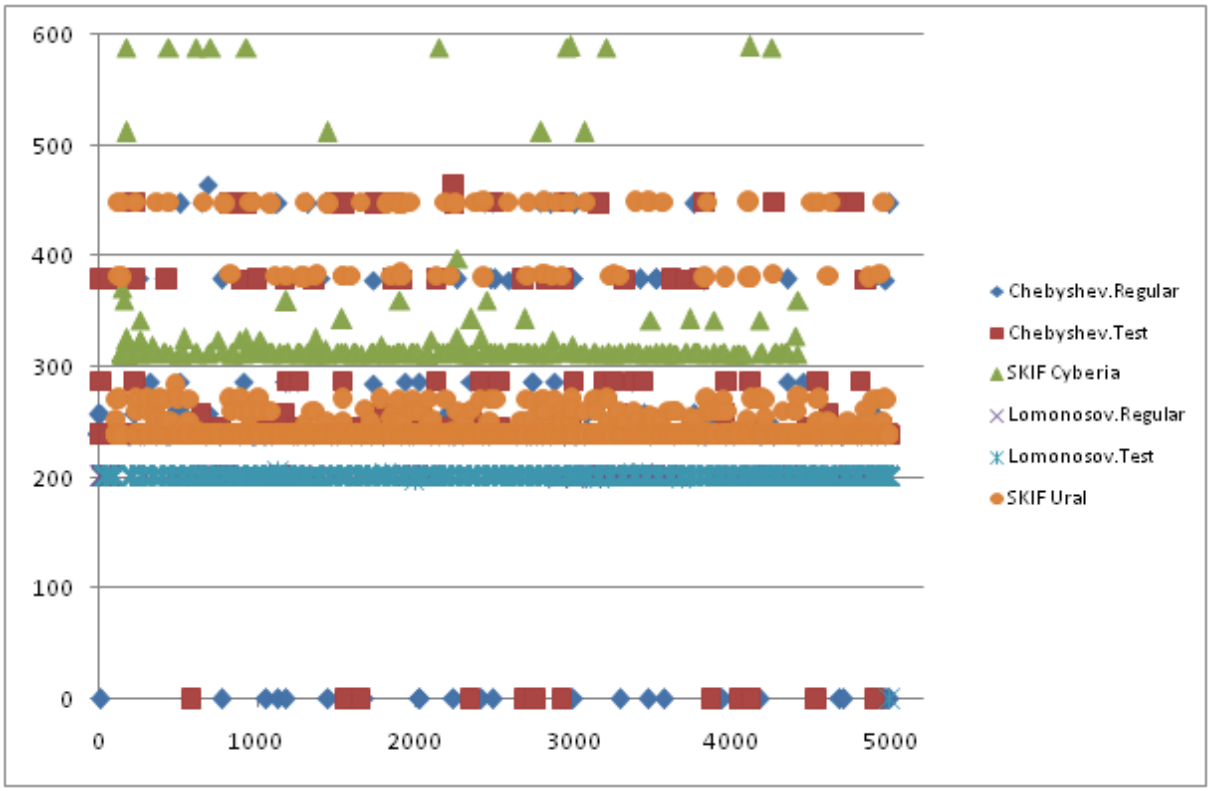


Рис.3. Время выполнения заданий на суперкомпьютерах

Таким образом, при анализе распределенных расчетов в многоуровневых средах следует использовать подход, отличный от анализа расчетов, выполняемых в рамках базовой архитектуры. В силу динамичности процесса многие характеристики также являются динамичными и могут быть представлены в виде графиков. Источником для них служат данные, формируемые инструментарием анализа потока событий серверов X-Com.

Работа выполняется при поддержке гранта Президента Российской Федерации для молодых российских ученых-кандидатов наук МК-5104.2011.9.

Литература

1. Система метакомпьютинга X-Com (официальный сайт) – [http://x-com.parallel.ru].
2. Воеводин Вл.В., Жолудев Ю.А., Соболев С.И., Стефанов К.С. Эволюция системы метакомпьютинга X-Com // Вестник Нижегородского государственного университета им. Н.И. Лобачевского. №4. 2009. С. 157–164.
3. Соболев С.И. Интеграция системы метакомпьютинга X-Com с системами управления прохождением заданий суперкомпьютерных комплексов. Распределенные вычисления и Грид-технологии в науке и образовании: Труды четвертой международной конференции (Дубна, 28 июня – 3 июля 2010 г.). С. 417–422.
4. Жолудев Ю.А., Соболев С.И., Стефанов К.С. Оценки и характеристики распределенных вычислительных сред. Высокопроизводительные параллельные вычисления на кластерных системах: Материалы XI Всероссийской конференции (Нижний Новгород, 1–3 ноября 2011 г.), С. 129–133.

СРАВНИТЕЛЬНОЕ ТЕСТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ УЗЛОВ ДВУХ СУПЕРКОМПЬЮТЕРОВ

А.А. Соколов, Д.В. Андрюшин

ФГУП «НИИ «Квант», Москва

Приводятся результаты исследования производительности кластерного суперкомпьютера, развернутого для Нижегородского государственного университета им. Н.И. Лобачевского, с использованием новых процессоров Intel Sandy Bridge. Выполнено сравнение с кластерным суперкомпьютером К-100, развёрнутым в ИПМ им. М.В. Келдыша и созданным с использованием процессоров предыдущего поколения Intel Westmere. При проведении исследования использовались элементы оригинальной многоуровневой методики тестирования, позволяющей проводить комплексный анализ производительности вычислительных систем. Особое внимание уделено сравнению вычислительных узлов, базовых блоков суперкомпьютеров.

Введение

В 2012 году для Нижегородского государственного университета им. Н.И. Лобачевского развернут вычислительный кластер, базовым блоком которого является двухсокетный узел с новыми 8-ми ядерными процессорами Intel Xeon E5-2660 (Sandy Bridge-EP) 2.2 ГГц. Десять узлов оснащены тремя и пятьдесят узлов оснащены двумя графическими ускорителями NVIDIA M2090 Tesla. Каждый узел снабжен 64 Гбайт оперативной памяти DDR3-1333 МГц. Все шестьдесят узлов кластера объединены вычислительной коммуникационной сетью Mellanox QDR InfiniBand, служебной сетью Gigabit Ethernet и работают под управлением ОС MS Windows Server 2008 HPC Edition.

Микропроцессор Intel Sandy Bridge-EP (далее - Sandy Bridge) содержит принципиально новые решения по микроархитектуре [1], среди которых необходимо выделить поддержку новых векторных команд AVX, четырехканальный контроллер памяти, а также наличие встроенного в кристалл контроллера PCI-E 3.0, который позволяет значительно повысить эффективность работы с GPU и коммуникационными сетями, например, InfiniBand.

В докладе представлены результаты исследования производительности кластера ННГУ с использованием элементов оригинальной базовой многоуровневой методики оценочного тестирования [2] – тесты 1, 3 и 5-го уровня. В рамках методики выполняется комплексная оценка производительности, от отдельных элементов до системы в целом. В приведенных результатах на специальных тестах уровня 1 и 3 показывается, как влияют на производительность особенности обращений к памяти при выполнении вычислений. Тесты 5-го уровня – комплексные, ядра прикладных задач. В них сочетание режимов работы оборудования определяется решаемой задачей. На этих задачах представляет интерес результат совместного влияния особенностей работы оборудования в разных режимах – достигаемая реальная производительность на тесте.

Для сравнения приведены результаты тестирования суперкомпьютера К-100, расположенного в ИПМ им. М.В. Келдыша [3]. Суперкомпьютер К-100 состоит из шестидесяти четырёх двухсокетных вычислительных узлов с 6-ти ядерными процессорами предыдущего поколения Intel Xeon X5670 Westmere 2,93 ГГц, снабженных тремя графическими ускорителями NVIDIA C2050 Fermi. Каждый узел снабжен 96 Гбайт оперативной памяти DDR3-1333 МГц. Узлы объединены вычислительными сетями

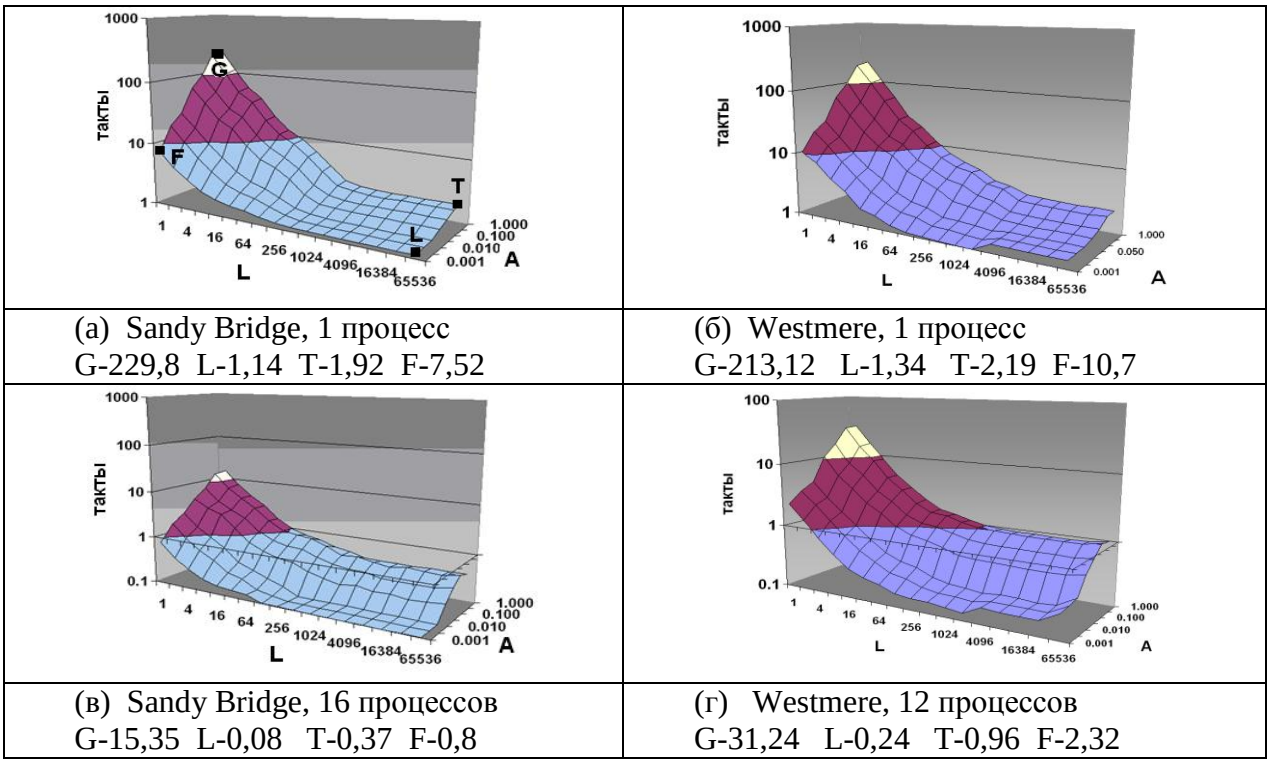
Qlogic QDR InfiniBand, МВС-Экспресс gen 2, служебной сетью Gigabit Ethernet и находятся под управлением ОС SUSE Linux Enterprise Server 11 SP1 (2.6.32.12-0.7).

При проведении исследования использовался компилятор icc версии 12.1 (кластер ННГУ) и 11.0 (К-100).

Результаты исследований

В тесте APEX-MAP [4] (уровень 1) измеряется среднее время доступа вычислительных процессов к памяти в тактах или микросекундах в зависимости от пространственно-временной локализации обращений. В процессе работы выполняется последовательное считывание с шагом 1 элементов некоторого массива L, начало которого в области памяти M (ее размер сопоставим с доступной физической памятью исследуемого оборудования) определяется случайным образом и управляется параметром A. Размер L – задает пространственную локализацию обращений к памяти, меняется от 1 (худшая) до 65536 (лучшая). Параметр A задает временную локализацию, изменяется от 0.001 (лучшая), до 1.0 (худшая). Для каждого сочетания L и A тест измеряет среднее время выполнения обращения к памяти в тактах процессора. По этим измерениям строится APEX-поверхность. Для анализа представляет интерес форма этой поверхности и точки сочетания предельных пространственно-временных локализаций – G, L, T и F (рис. 1а).

При рассмотрении APEX-поверхностей одного ядра (рис. 1а и б) можно отметить, что Sandy Bridge демонстрирует лучшие, чем Westmere показатели работы с памятью при наличии хорошей временной или пространственной локализации (точки F,T и L), что говорит о лучшей оптимизации алгоритмов работы с кэш памятью в новой микроархитектуре. Однако в самой плохой точке – G – результат Sandy Bridge немного хуже, чем у Westmere. При рассмотрении APEX-поверхностей варианта максимальной занятости вычислительных ядер (рис. 1в и г) видно общее снижение количества тактов на одно обращение, что говорит о наличии толерантности микропроцессоров к задержкам выполнения операций с памятью за счет увеличения одновременно выполняемого их числа. При этом результаты Sandy Bridge лучше результатов Westmere.



(а) Sandy Bridge, 1 процесс
G-229,8 L-1,14 T-1,92 F-7,52

(б) Westmere, 1 процесс
G-213,12 L-1,34 T-2,19 F-10,7

(в) Sandy Bridge, 16 процессов
G-15,35 L-0,08 T-0,37 F-0,8

(г) Westmere, 12 процессов
G-31,24 L-0,24 T-0,96 F-2,32

Рис. 1. Тест APEX-MAP

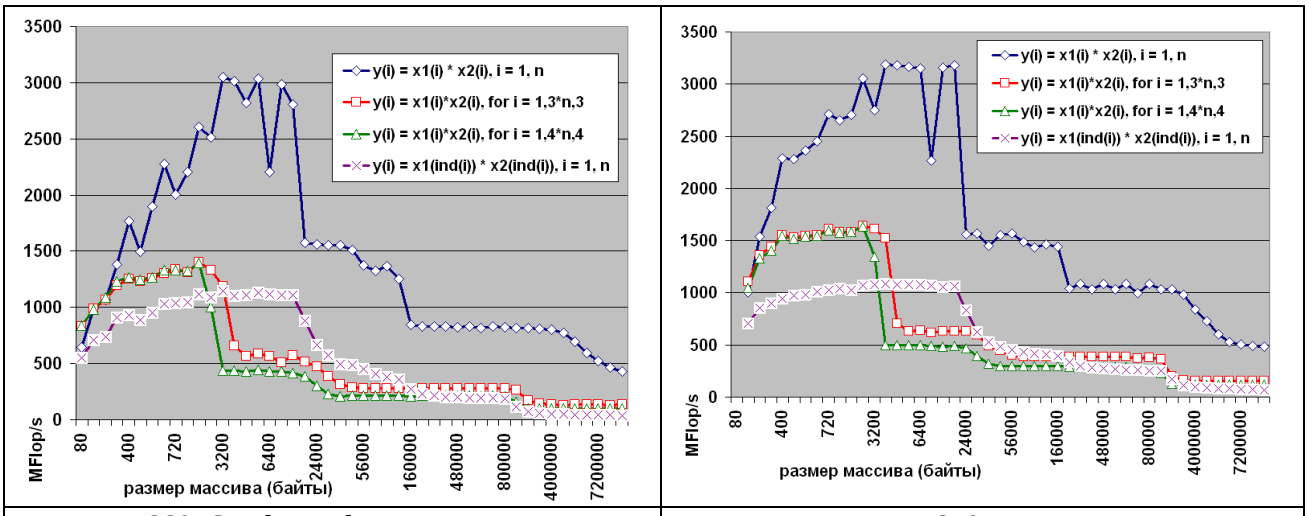
Тесты EuroBen Benchmark [5] (уровень 3) – это 31 тест в виде операций разной сложности над векторами. Собранные специальным образом в группы они используются для испытания процессоров в условиях целенаправленно изменяемой нагрузки. В табл. 1 приведены две группы тестов. В группе 1 выполняется операция перемножения векторов с изменением режима доступа к элементам массива. В группе 2 изменяется соотношение между количеством вычислительных операций и операций обращения в память. Результаты испытаний на этих группах даны на рис.2-3.

В табл. 1 (группа 2) тест 9th DegrPolyn – это вычисление полинома 9-й степени по схеме Горнера: y(i) = ((…(a9*x(i) + a8)*x(i) + …)*x(i) + a0.

Таблица 1. Группы тестов со специально изменяемой нагрузкой оборудования (область точки T apex-map)

Группа 1 (усложнение доступа к памяти)	**Группа 2** (повышение интенсивности вычислений)
y(i) = x1(i) * x2(i), i = 1, n	y(i) = x1(i) + x2(i), i = 1, n
y(i) = x1(i) * x2(i), i = 1,3*n,3	y(i) = x1(i)*x2(i) + x3(i)*x4(i)
y(i) = x1(i) * x2(i), i = 1,4*n,4	y(i) = x1(i+1) - 2*x1(i) + x1(i-1)
y(i) = x1(ind(i)) * x2(ind(i)), i = 1, n	9th DegrPolyn

Анализ результатов показывает, что при выполнении операции умножения векторов при меньшей тактовой частоте процессор Sandy Bridge показывает сравнимые с Westmere показатели производительности при работе в кэш памяти L1. При усложнении режима доступа к массиву, расположенному в кэш памяти, процессор Westmere, за счет большей тактовой частоты, незначительно опережает Sandy Bridge. При работе с оперативной памятью показатели производительности процессоров выравниваются. При этом полученные результаты показывают резкое падение реальной производительности при усложнении доступа к данным из памяти (группа 1).

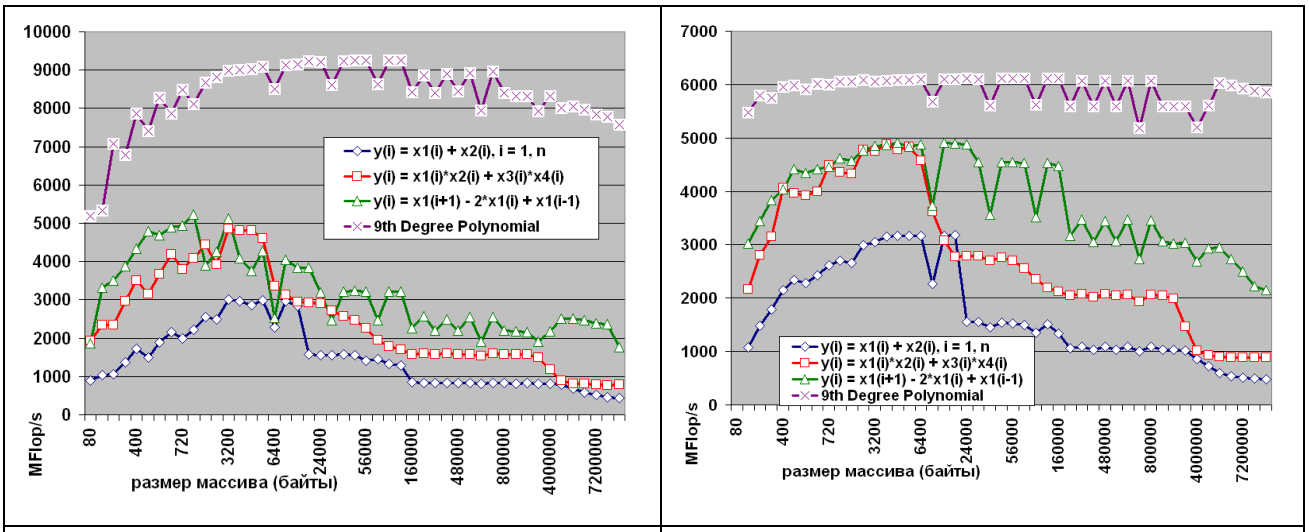


E5-2660, Sandy Bridge | X5670, Westmere

Рис. 2. Результаты измерений Euroben (группа 1)

На тесте вычислений по схеме Горнера процессор Sandy Bridge лучше процессора Westmere за счет наличия 256-разрядных устройств типа SIMD (команды AVX). На этом тесте на два обращения к памяти приходится двадцать семь вычислительных операций, и показатели производительности процессоров приближаются к пиковым значениям. Увеличение операций доступа к памяти по отношению к вычислительным опера-

циям и разбалансировка операций умножения и сложения приводит серьезному ухудшению демонстрируемой производительности для процессора Sandy Bridge, а большая тактовая частота процессора Westmere позволяет ему обогнать Sandy Bridge при работе с кэш памятью.



E5-2660, Sandy Bridge	X5670, Westmere

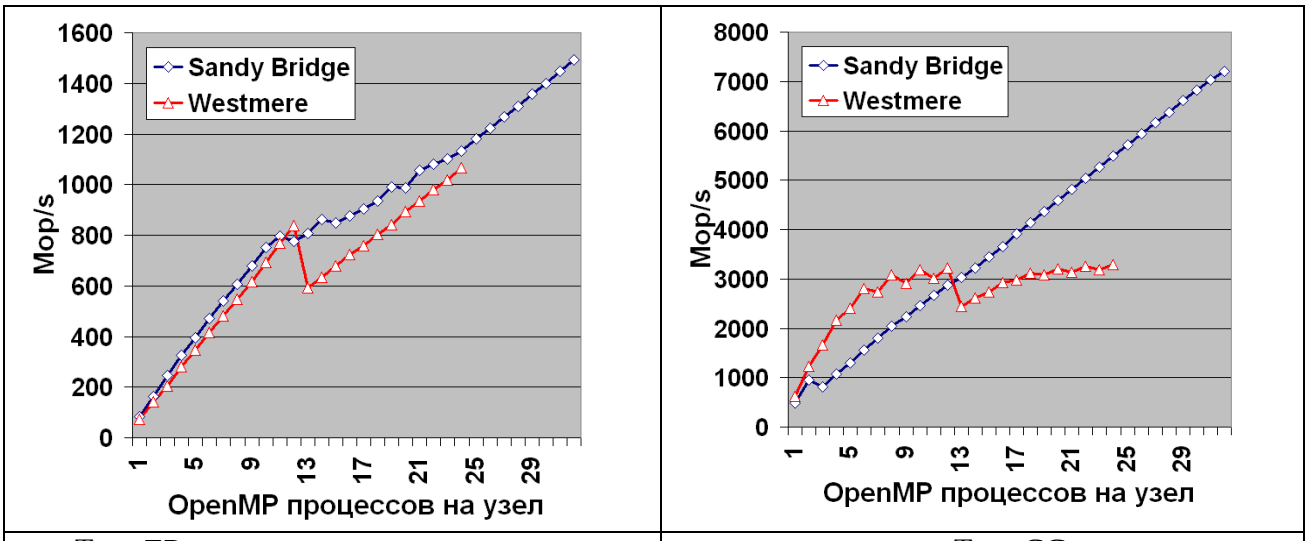
Рис. 3. Результаты измерений Euroben (группа 2)

Тесты пакета NAS Parallel Benchmark [6] (уровень 5) используются для комплексной оценки производительности. Приведены результаты для OpenMP-вариантов тестов EP, CG (класс C).

Тест EP реализует алгоритм накопления двумерной статистики для псевдослучайных чисел с распределением Гаусса, является типичным для задач, решаемых методом Монте-Карло, и позволяет оценивать производительность на задачах с интенсивными вычислениями с плавающей точкой двойной точности.

Тест CG реализует метод сопряженных градиентов для вычисления приближения наименьшего собственного значения большой разреженной симметричной, положительно определенной матрицы. Для теста CG характерно попадание в область точки G APEX-поверхности с плохой пространственно-временной локализацией обращений к памяти.

Результаты измерений на тесте EP показывают, что, несмотря на меньшую тактовую частоту, процессор Sandy Bridge демонстрирует сравнимые результаты при интенсивных вычислениях с плавающей точкой, а за счет большего количества вычислительных ядер, общая производительность узла с процессорами Sandy Bridge в 1,5 раза выше производительности узла с процессорами Westmere. Результаты измерений на тесте CG показывают, что для процессора Westmere при переходе рубежа в 10 вычислительных процессов на узел, подсистема памяти перестает справляться с потоком обращений в память, что ограничивает масштабируемость приложения. Однако при любом количестве процессов в диапазоне от 1 до 12 производительность узла с процессорами Westmere выше производительности узла с процессорами Sandy Bridge. При использовании узла с процессорами Sandy Bridge, за счет большей ширины каналов памяти и кэш, масштабируемость с ростом вычислительных узлов сохраняется, и, как следствие, общая производительность узла в 2,2 раза выше производительности узла с процессорами Westmere.



Тест EP	Тест CG

Рис. 4. Тесты пакета NAS Parallel Benchmark (сравнительные результаты)

Результаты сравнительного тестирования показывают, что новые архитектурные решения обеспечивают процессору Sandy Bridge превосходство над Westmere. Использование 256-битных блоков SIMD обработки и более широких трактов передачи данных позволяет процессору Sandy Bridge эффективнее, по сравнению с Westmere, выполнять операции с плавающей точкой, особенно в случае размещения данных в кэш памяти. Наличие 4-х канального контролера памяти в процессоре Sandy Bridge (вместо 3-х канального у Westmere) позволяет обеспечивать большую толерантность узла по обращениям к памяти и, как следствие масштабируемость приложений с ростом числа вычислительных процессов.

В настоящее время заканчиваются работы по тестированию коммуникационной сети InfiniBand и графических ускорителей кластера ННГУ. Результаты тестирования планируется привести в докладе на конференции.

Литература

1. Sandy Bridge for Servers D. Kanter, 07/08/2011, сайт http://www.realworldtech.com/
2. Горбунов В.С., Эйсымонт Л.К., Речинский А.В., Заборовский В.С., Забеднов П.В. Суперкомпьютеры для промышленности – вопросы тестирования, анализа и разработки // Материалы Второй Всероссийской конференции по суперкомпьютерным технологиям (СКТ2012). Дивноморское, октябрь 2012.
3. http://www.kiam.ru/MVS/resourses/k100.html. Сетевой ресурс.
4. Strohmaier E., Shan H. Architecture independent performance characterization and benchmarking for scientific applications. 2004.
5. EuroBen Benchmark Homepage – [http://www.hpcresearch.nl/euroben/index.php].
6. NAS Parallel Benchmark Homepage – [http://www.nas.nasa.gov/publications/npb.html].

РАСПАРАЛЛЕЛИВАНИЕ СХЕМ ХИМИЧЕСКИХ РЕАКЦИЙ ПРИ РЕШЕНИИ ОБРАТНЫХ ЗАДАЧ

С.И. Спивак[1], А.С. Исмагилова[2]

[1]Башкирский госуниверситет, Институт нефтехимии и катализа РАН
[2]Башкирский госуниверситет

Современные технологии параллельных вычислений позволяют при интерпретации экспериментальных исследований обратиться к таким классам задач, решение которых традиционными методами затруднено. Здесь речь идет не о сокращении машинного времени при решении задач. Под распараллеливанием схем будем понимать разложение всей задачи на подзадачи меньшей размерности, анализ которых существенно упрощает решение обратных задач химической кинетики.

Постановка обратной задачи состоит в том, что предполагается известным стадийный механизм реакции. Задача состоит в определении кинетических констант, в рамках этой схемы описывающих имеющийся экспериментальный материал.

Пусть механизм реакции задан в виде совокупности элементарных стадий:

$$\sum_{j=1}^{n}\alpha_{ij}A_j \underset{k_i^-}{\overset{k_i^+}{\leftrightarrow}} \sum_{j=1}^{n}\beta_{ij}A_j\ ,\ 1\le i\le m,$$

где A_j, $1\le j\le n$, – символы участвующих в реакции веществ, n – общее число веществ, m - общее число элементарных стадий, α_{ij}, β_{ij} – стехиометрические коэффициенты, k_i^+, k_i^- – константы скорости i-й стадии в прямом и обратном направлениях.

Скорость i-й элементарной стадии w_i определяется следующим образом:

$$w_i = w_i^+ - w_i^-\ ,\ 1\le i\le m,$$

где w_i^+ – скорость стадии в прямом направлении, w_i^- – скорость стадии в обратном направлении.

Выражения для w_i^+ и w_i^- выписываются в соответствии с законом действующих масс:

$$w_i^+ = k_i^+\prod_{j=1}^{n}a_j^{\alpha_{ij}}\ ,\ w_i^- = k_i^-\prod_{j=1}^{n}a_j^{\beta_{ij}}\ , \tag{1}$$

где a_j, $1\le j\le n$, – концентрации веществ A_j.

Система дифференциальных уравнений, описывающая изменение концентраций во времени, выглядит следующим образом:

$$\frac{da}{dt} = \Gamma^T w,\ a(0) = a^0\ , \tag{2}$$

где $a = (a_1\ldots a_n)^T$ – вектор-столбец концентраций, $w = (w_1\ldots w_m)^T$ – вектор-столбец скоростей стадий, выписанных в соответствии с (1), $a^0 = (a_1^0\ldots a_n^0)^T$ – вектор-столбец начальных концентраций, Γ – стехиометрическая $(m\times n)$-матрица, элемент которой $\gamma_{ij} = \beta_{ij} - \alpha_{ij}$, $1\le i\le m$, $1\le j\le n$.

Матрица Γ удовлетворяет закону постоянства атомов данного вида

$$\Gamma \cdot A \equiv 0, \qquad (3)$$

где A – молекулярная матрица, элемент которой a_{jk}, $1 \le k \le q$, представляет собой число атомов k-го элемента в молекуле j-го вещества, q – общее число различных типов элементов, из которых состоят A_j. Условие (3) означает существование в системе (2) не менее q линейных законов сохранения.

Одним из основных моментов в работе является анализ неединственности решения обратной задачи для моделей нестационарной химической кинетики и построение алгоритма исследования неединственности, позволяющего описать нелинейные параметрические функции кинетических констант, допускающих однозначное оценивание по заданной структуре исходной модели [1]. Основная сложность – неоднозначность решения: в эксперименте измеряются только исходные вещества и продукты реакции, отсутствует информация по промежуточным веществам [2]. Следовательно, повышается степень неопределенности при оценивании параметров математических моделей кинетики сложных реакций. При анализе конкретных сложных реакций возникает трудность с громоздкостью вычислений. Имеются в виду аналитические вычисления с нелинейными выражениями.

В связи с этим целесообразно применить теоретико-графовый метод. Графы значительно расширяют круг решаемых задач химической кинетики и позволяют изучать свойства сложных химических реакций. Двудольные графы наиболее точно отражают свойства системы реакций, поскольку в рассмотрение вводятся вершины двух типов: вершины, характеризующие вещества, и вершины, характеризующие реакции [3].

Направление ребер определяется следующим образом. Если вещество A_j расходуется в реакции w_i, то отрезок имеет направление от A-вершины к w-вершине и имеет вес, численно равный стехиометрическому коэффициенту α_{ij} (рис.1). Если вещество A_j образуется в реакции w_i, то отрезок имеет направление от w-вершины к A-вершине и имеет вес, численно равный стехиометрическому коэффициенту β_{ij} (рис.2). Если вещество A_j не участвует в реакции w_i, то соответствующие вершины не соединены отрезком.

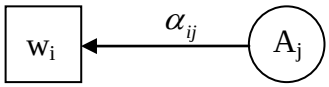


Рис.1. Вещество расходуется в реакции

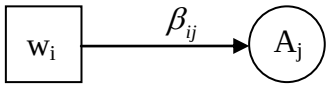


Рис.2. Вещество образуется в реакции

Одним из основных понятий является маршрут [4]. Маршрут есть вектор, умножение элементов которого на соответствующие стадии механизма сложной реакции вместе с последующим сложением приводит к суммарному уравнению реакции, которое уже не содержит промежуточных веществ.

Справедлива следующая теорема [5].

Теорема. Маршрут реакции есть циклический подграф исходного графа. Объединение таких подграфов образует полный граф, т.е. граф исходной системы реакций. Число независимых маршрутов равно числу независимых циклов графа Вольперта.

Маршрут выделяет из всей совокупности стадий некоторую подсистему, в которую входит только часть стадий исходного механизма. Таким образом, вместо анализа информативности для всей сложной системы рассматриваются те, которые отвечают за

протекание реакции по каждому из независимых маршрутов. Вместо одной сложной системы получили несколько существенно более простых схем реакций. Число таких простых систем равно числу независимых маршрутов. Маршруты позволяют уменьшить размерность исследуемой системы реакций, а также определить физико-химический смысл каждой подсистемы.

Теорема. Объединение базисов нелинейных параметрических функций кинетических параметров, допускающих однозначное оценивание, совпадает с базисом нелинейных параметрических функций исходной сложной системы реакций.

Действительно, поскольку детальный механизм сложной системы реакций может быть представлен четырьмя эквивалентными формами – символьной (с перечислением всех участников и стадий реакции), кинетические уравнения, матричной и в виде двудольного графа, то это позволяет установить взаимно однозначное соответствие между графом системы реакций, стехиометрической матрицей и якобианом.

При построении кинетических моделей сложных реакций важное значение также имеет понятие ключевого вещества [6]. Под ключевыми веществами понимают такие, для которых концентрации всех веществ линейно выражаются через базис ключевых веществ. Использование информации о ключевых веществах имеет решающее значение при решении обратных задач идентификации механизмов сложных химических реакций.

Известно, что набор ключевых веществ может быть установлен путем анализа структуры стехиометрической матрицы. Для этого достаточно найти набор линейно независимых столбцов этой матрицы. В качестве набора ключевых веществ могут использоваться такие, каждое из которых участвует только в одной стадии реакции, входящей в совокупность линейно независимых стадий.

В работе [7] рассмотрен теоретико-графовый метод определения ключевых веществ. Для анализа использованы специальные графы, аналогичные с введенными А.И. Вольпертом [3] графами механизмов сложных химических реакций. Отличие от графов Вольперта состоит в том, что множество вершин разбивается на множество концентраций всех веществ, участвующих в реакции, и множество атомов, из которых эти вещества состоят.

Имеет место следующая теорема.

Теорема. Существует преобразование, переводящее исходный граф в граф, часть вершин которого не имеет исходящих дуг. Указанные вершины достижимы из базиса ключевых веществ.

В [8] сформулирован критерий единственности решения обратной задачи химической кинетики. Условием единственности решения становится возможность экспериментального измерения любого базиса ключевых веществ, участников реакции. Использование всех законов сохранения может позволить исключить лишь некоторые концентрации неизмеряемых промежуточных веществ. Для исключения всех концентраций неизмеряемых веществ, приведения кинетической модели к системе дифференциальных уравнений относительно только измеряемых веществ, необходимо измерение хотя бы части промежуточных участников реакции. Отсюда следуют возможности специального планирования измерений с целью однозначного решения обратной задачи.

Процесс нахождения ключевых веществ графическим методом алгоритмизирован [7], допускает компьютерную интерпретацию, что является предметом нашей дальнейшей работы.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект № 11-01-97020.

Литература

1. Горский В.Г., Спивак С.И. Исследование идентифицируемости параметров – один из важнейших этапов построения математических моделей в химии // Журнал структурной химии. – 1988. – Т. 29, № 6. – С. 119.
2. Яблонский Г.С., Спивак С.И. Математические модели химической кинетики. – М.: Знание, 1977.
3. Вольперт А.И., Худяев С.И. Анализ в классах разрывных функций и уравнения математической физики. – М.: Наука, 1975.
4. Темкин М.И. Кинетика стационарных сложных реакций // Механизм и кинетика сложных каталитических реакций. Лекции, прочитанные на первом симпозиуме международного конгресса по катализу. – М.: Наука, 1970.
5. Спивак С.И., Исмагилова А.С., Хамитова И.А. // ДАН. 2010. Т. 434. № 4. С.499–501.
6. Горский В.Г. Планирование кинетических экспериментов. М.: Наука, 1984. 241 с.
7. Спивак С.И., Исмагилова А.С., Хамитова И.А. // ДАН. 2012. Т. 443. № 6. С. 1–3.
8. Спивак С.И., Горский В.Г. О полноте доступных кинетических измерений при определении констант скоростей сложной химической реакции // Химическая физика. 1982. Т.1. №2. С. 237–243.

ОТОБРАЖЕНИЕ ГРАФА ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ НА ГРАФ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

Н.В. Старостин, М.А. Панкратова

Нижегородский госуниверситет им. Н.И.Лобачевского

В работе рассматривается актуальная задача отображения графа на граф. Предлагается математическая постановка задачи, изучены и программно реализованы известные подходы к решению задачи, разрабатывается генетический и многоуровневый подходы решения поставленной задачи.

В работе рассматривается актуальная задача отображения графа на граф, которая возникает на этапе назначения параллельной программы планировщиком на выделенный вычислительный ресурс. Рассмотрим задачу отображения на примере упрощенной модели вычислительной системы $H(V,P,E)$, где V – множество всех сетевых устройств вычислительной системы; $P=\{p_1,...,p_n\}\subseteq V$ – вычислители (ядра, процессоры, узлы); $E\subseteq V^{(2)}$ – непосредственные двунаправленные связи между элементами вычислительной системы. Имеется неориентированный граф $G(Q,C)$ виртуальной топологии, где $Q=\{q_1,...,q_n\}$ – множество вершин граф виртуальной топологии; $C\subseteq Q^{(2)}$ – множество связей виртуальной топологии. Требуется назначить элементы Q виртуальной топологии G по вычислителям P системы H.

Введем симметричную матрицу коммуникаций (расстояний, трудоемкости) $(d_{ij})_{n*n}$, где d_{ij} – расстояние (трудоемкость коммуникаций) между вычислителями p_i и p_j в графе H. Обозначим через $(a_{ij})_{n*n}$ матрицу смежности графа топологии G.

Варьируемые параметры: матрица перестановок $X=(x_{ij})_{n*n}$, где $x_{ij}=1$, если элемент $q_i\in Q$ виртуальной топологии приведен в соответствие с вычислителем $p_j\in P$; $x_{ij}=0$ в противном случае.

Ограничения

$$\sum_{j=1}^{n} x_{ij},\ i=\overline{1,n}; \tag{1}$$

$$\sum_{i=1}^{n} x_{ij}=1,\ j=\overline{1,n}; \tag{2}$$

$$x_{ij}\in\{0,1\},\ i,j\in\overline{1,n}. \tag{3}$$

Критерии

В задаче отображения требуется минимизировать издержки на коммуникации. При построении критерия можно исходить из суммарных затрат на коммуникацию или минимизировать самые «дорогие» коммуникации:

$$F_{SUM}(X)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}\left(\sum_{l=1}^{n}\sum_{k=1}^{n}x_{il}x_{jk}d_{lk}\right)\to\min \tag{4}$$

$$F_{MAX}(X) = \max_{i,j=\overline{1,n}} a_{ij} \left(\sum_{l=1}^{n} \sum_{k=1}^{n} x_{il} x_{jk} d_{lk} \right) \to \min \qquad (5)$$

В общем случае задачи отображения с критериями (4) и (5) являются NP-трудными в сильном смысле [1]. На практике порядки прикладных задач – сотни тысяч. Использование точных методов для решения прикладных задач не представляется возможным.

В работе рассматриваются эвристические алгоритмы, которые способны за приемлемое время найти решение задачи. Предлагаются алгоритмы, основанные на генетическом и многоуровневом подходах. Генетический алгоритм [2], перебирая различные варианты отображения, генерирует новые, копируя лучшие характеристики имеющихся решений. Многоуровневый алгоритм основан на многократной редукции исходной задачи, поиске и восстановлении решения.

В ходе работы были проведены вычислительные эксперименты на разных классах задач, определены особенности алгоритмов, сделаны выводы.

Работа выполнена при частичной финансовой поддержке Минобрнауки России, госсоглашение №14.B37.21.0878.

Литература

1. El-Rewini H., Lewis T.G., Ali H.H. Task Scheduling in Parallel and Distributed Systems. Prentice Hall, 1994.
2. Батищев Д.И. Генетические алгоритмы решения экстремальных задач: учеб. пособие. Воронеж. гос. техн. ун-т; Нижегородский гос. ун-т. Воронеж, 1995. 69 с.
3. Батищев Д.И., Старостин Н.В., Филимонов А.В. Многоуровневая декомпозиция гиперграфовых структур // Информационные технологии: прилож. к журналу. 2008. № 5 (141). С. 1–32.

ВЫЧИСЛИТЕЛЬНЫЕ ЗАДАЧИ И ПЕРСПЕКТИВЫ 3D-РЕКОНСТРУКЦИИ В ОПТОАКУСТИКЕ

П.В. Субочев, В.В. Перекатова, И.В. Турчин

Институт прикладной физики РАН, Нижний Новгород

Обсуждаются проблемы, возникающие при решении обратных задач оптоакустического биоимиджинга. Рассматриваются вычислительные задачи и реконструктивные алгоритмы, используемые в оптоакустической томографии и микроскопии.

Оптоакустическая (ОА) диагностика – современный метод биомедицинской визуализации, основанный на регистрации ультразвуковых волн, возбуждаемых в исследуемой среде при поглощении импульсного лазерного излучения оптическими неоднородностями [1].

Основным преимуществом ОА методов визуализации биотканей перед полностью оптическими является улучшенное пространственное разрешение на глубинах от нескольких миллиметров до нескольких сантиметров [2]. В частности, современные импульсные лазеры позволяют перестраивать длину волны излучения для обеспечения повышенной контрастности оптического поглощения исследуемых внутренних биологических структур по отношению к окружающим тканям. Таким образом, существует возможность оптимизации ОА контраста произвольных свето-поглощающих агентов (таких как гемоглобин, меланин, вода, и т.д.), что позволяет визуализировать сосудистый рисунок биотканей, определять локальный кислородный статус крови, скорость циркуляции крови, локальную внутреннюю температуру. Для дальнейшего повышения контраста возможно использование экзогенных контрастных маркеров (таких как органические красители, наночастицы, флуоресцентные белки, репортерные гены [2]).

В современной оптоакустической диагностике [3] используются две основных схемы визуализации: оптоакустическая томография и оптоакустическая микроскопия. При осуществлении оптоакустической томографии оптической засветке подлежат обширные области исследуемой биологической ткани. Оптоакустические импульсы регистрируются посредством многоэлементных антенных систем, в результате чего сигналы от одного и того же источника могут быть зарегистрированы различными элементами антенной системы с различными временными задержками, что позволяет восстанавливать пространственное распределение источников за счет использования алгоритмов реконструкции [4].

Другим распространенным способом получения оптоакустических изображений является микроскопия [5], связанная с использованием одноэлементных фокусируемых акустических приемников. В отличие от томографии, микроскопия вообще говоря не требует применения алгоритмов реконструкции, поскольку конечный размер фокусной перетяжки акустического приемника позволяет конкретизировать область визуализации в ходе сканирования. Между тем, решение обратных задач [6] при микроскопии позволяет уточнить реальный размер визуализируемых объектов, избавившись от артефактов, связанных с конечностью диаграммы направленности антенны.

Способы решения обратной задачи трехмерной оптико-акустической томографии, а также упрощенные реконструктивные алгоритмы оптико-акустической микроскопии будут подробнее рассмотрены в докладе.

Работа поддержана Российским фондом фундаментальных исследований (номер проекта 12-02-31309 мол_а).

Литература

1. Khokhlova T.D., Pelivanov I.M., Karabutov A.A. Methods of optoacoustic diagnostics of biological tissues // Acoustical Physics. 2009. 55(4-5). P. 674.
2. Wang L.V., Hu S. Photoacoustic Tomography: In Vivo Imaging from Organelles to Organs // Science 23. 2012. 335 (6075). P. 1458.
3. Beard P.C. Biomedical photoacoustic imaging // Interface Focus. 2011. 1(4). P. 602.
4. Cox B., Laufer J.G., Arridge S.R., Beard P.C. Quantitative spectroscopic photoacoustic imaging: a review // Journal of Biomedical Optics. 2012. 17(6). P. 061202.
5. Zhang H.F., K. Maslov, G. Stoica, L.V. Wang Functional photoacoustic microscopy for high-resolution and noninvasive in vivo imaging // Nature Biotechnology. 2006. 24. P. 848.
6. Kuchment P. and Kunyansky L. Mathematics of thermoacoustic tomography // Euro. J. Appl. Math. 2008. 19. P. 191.

МЕТОД МАССОВОГО РАСПАРАЛЛЕЛИВАНИЯ ВЫЧИСЛЕНИЙ ПРИ РЕШЕНИИ КРАЕВЫХ ЗАДАЧ МАТЕМАТИЧЕСКОЙ ФИЗИКИ

А.Н. Супрун

Нижегородский государственный архитектурно-строительный университет

Разработанной для расчета сложных строительных объектов метод деления на проектные единицы распространяется на краевые задачи математической физики. В качестве примера приводится последовательность решения краевой задачи теории потенциала.

Положим, что некоторая вычислительная система *CS* (рабочая станция, кластер рабочих станций и т.п.) использует для решения краевых задач математической физики прикладной программный комплекс *P*. Пусть далее программа $p \in P$ использует какой-либо метод дискретизации, сводящий некоторую стационарную краевую задачу *B* для области Ω к решению системы *S* линейных алгебраических уравнений. Следует заметить, что обычно значительную часть общей трудоёмкости решения задачи *B* будут составлять операции формирования, решения и обработки результатов решения системы уравнений *S*. Поэтому можно условно принять, что время *t*, затрачиваемое вычислительной системой *CS* (“стоимость” процедуры [1]), будет определяться некоторой возрастающей функцией *f* от числа *n* уравнений:

$$t = f(n). \qquad (1)$$

Предположим далее, что из некоторых разумных соображений принята максимально допустимая продолжительность t_* вычислений для одной задачи на *CS*. Тогда из (1) можно получить следующее ограничение на порядок дискретизации Ω:

$$n \le n_* = f^{-1}(t_*),$$

где f^{-1} – обратная f функция.

Известно, что решение нестационарных и нелинейных краевых задач со сложными геометрическими и физическими особенностями (задачи класса *H*) не обеспечивается современными многопроцессорными вычислительными системами даже с пиковой производительностью в тысячи Tflops. Это связано, прежде всего, с низкой производительностью современных мультипроцессорных систем применительно к актуальным задачам класса *H*. К таким, например, как моделирование возможных чрезвычайных ситуаций при оценке живучести проектируемых технических объектов, реконструкция техногенных катастроф и террористических актов и т.п. При этом главным препятствием к существенному повышению производительности вычислительных систем на задачах *H* является отсутствие математических методов, позволяющих строить вычислительные процессы по технологии массового распараллеливания.

В связи с этим, в настоящее время назрела необходимость в переосмысливании традиционных подходов к построению параллельных процедур. Однако в настоящее время в этом направлении уже получены некоторые обнадеживающие результаты. Так для расчета сложных стержневых строительных конструкций был разработан метод разделения объекта на проектные единицы [1, 2] , продемонстрировавший возможность построения эффективных алгоритмов массового распараллеливания.

Суть метода состоит в условном разделении сложного строительного объекта на составляющие элементы – проектные единицы с последующим построением вычисли-

тельного процесса в распределенных вычислительных средах или на многомашинных комплексах в виде параллельных и согласующихся решений задач уменьшенной статической неопределимости. Указанный метод может быть распространен на сплошные среды и широкий класс краевых задач математической физики.

Пусть p – прикладная программа, разработанная на базе некоторого метода дискретизации, обеспечивающего достаточно точное решение корректно поставленной краевой задачи B с начальными C_0 и краевыми C_B условиями для допустимых программой p областей Ω. Для решения задачи B разделим Ω на конечное множество E из M элементов $e_1, \ldots, e_m, \ldots, e_M$, каждый из которых допустим для расчета программой p. Взаимосвязь элементов E можно иллюстрировать графом смежности (рис.1).

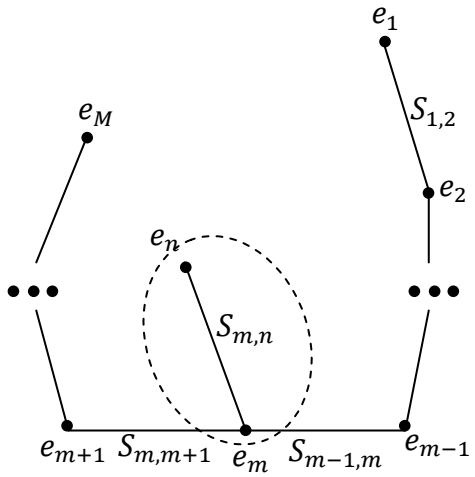


Рис.1. Граф смежности элементов (вершины – элементы, рёбра – поверхности разделения элементов)

Положим для определенности, что прикладная программа p позволяет получать численные решения третьей краевой задачи. Требуется решить первую краевую задачу: найти потенциал $U(x)$, удовлетворяющий уравнению Лапласа в области Ω по заданным его значениям $U(x)= \varphi(x)$ на границе $\overline{\Omega}$. При этом программа p использует замену областей изменения независимых переменных Ω и $\overline{\Omega}$ на дискретные множества точек, соответственно, ω и $\overline{\omega}$. В результате этой процедуры исходной задаче сопоставляется система конечного числа n линейных алгебраических уравнений. Тем самым p будет выполнять преобразование дискретного множества численных значений $U(x)= \varphi(x)$ на $\overline{\Omega}$ (обозначим $\overline{U}_{\overline{\omega}}$) в численное множество значений $U(x)$ на Ω (U_ω)*:*

$$U_\omega = A\,\overline{U}_{\overline{\omega}}, \qquad (2)$$

где A – линейный оператор.

Разделением Ω на элементы выполняется первый шаг к распараллеливанию вычислений:

$$U_\omega = \sum_{m=1}^{M}(A\overline{U}_m^* + A\overline{U}_m^{**}),$$

где

$$\overline{U}_m^* = \begin{cases} \varphi(x) & \text{на} \quad S_m^- , \\ 0 & \text{на} \quad \overline{\omega}_m^+ , \end{cases} \qquad \overline{U}_m^{**} = \begin{cases} 0 & \text{на} \quad S_m^-, \\ \overline{U}_m^+ & \text{на} \quad \overline{\omega}_m^+ , \end{cases}$$

S_m^- – часть общей поверхности S_m элемента e_m , исключая поверхности S_m^+ , образованные отделением e_m от других элементов, $\overline{\omega}_m^+$ - дискретное множество точек на S_m^+ , $\overline{U}_m^+$ - граничные условия на $\overline{\omega}_m^+$. Для определения $\overline{U}_m^+$ рассмотрим выделенный пунктиром на рис.1 фрагмент объекта расчёта в условно разделённом состоянии (рис.2), где S_m^-, S_n^- – “свободные” и $S_{m,n}^+$, $S_{m,m-1}^+$, $S_{n,m}^+$ – “смежные” поверхности, $S_{m,n}$ – поверхность разделения элементов m и n $(m<n)$.

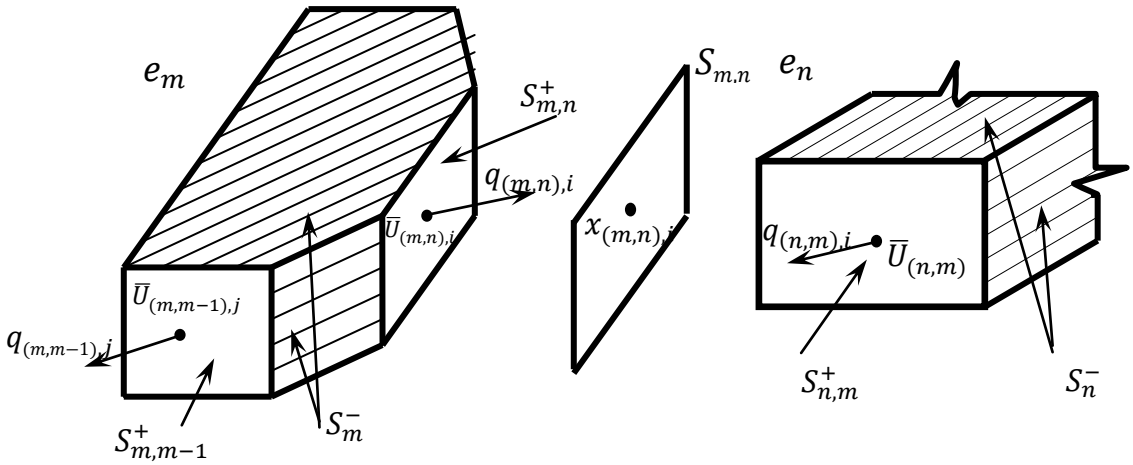


Рис.2. Фрагмент области Ω в разделённом состоянии

Используя общие соображения, обосновывающие применение уравнения Лапласа для описания физических процессов, введём условия сопряжения элементов e_m и e_n в виде уравнений совместности (3) и баланса (4):

$$\overline{U}_{(m,n),i} - \overline{U}_{(n,m),i} = 0\,, \tag{3}$$

$$q_{(m,n),i} + q_{(n,m),i} = 0 \tag{4}$$

для всех $x_{(m,n),i} \in \overline{\omega}_{m,n}$, где $\overline{\omega}_{m,n}$ – дискретное множество точек на поверхности $S_{m,n}$ разделения элементов e_m и e_n (рис.2), $\overline{U}_{(m,n),i}$ и $\overline{U}_{(n,m),i}$ – потенциалы в точке $x_{(m,n),i}$ соответственно на $S^+_{m,n}$ и $S^+_{n,m}$, $q_{(m,n),i} = \lambda \cdot \partial U_{(m,n),i}/\partial \nu_{m,n}$, $q_{(n,m),i} = \lambda \cdot \partial U_{(n,m),i}/\partial \nu_{n,m}$, λ - физическая константа, $\nu_{m,n}$ и $\nu_{n,m}$ – нормали в точке $x_{(m,n),i}$ к поверхностям соответственно $S^+_{m,n}$ и $S^+_{n,m}$. В задаче теплопроводности q … – тепловой поток, λ – коэффициент теплопроводности.

Решим поставленную задачу в 6 этапов.

1. Для всех $e_m = \Omega_m \epsilon E$ определим потенциалы $\beta_{m,j}$ в точках $x_j \epsilon \overline{\omega}^+_m$. Для этого воспользуемся программой p

$$\overline{U}^+_{m,\varphi}(x) = A\begin{pmatrix} \varphi_{(x)} \text{ на } S^-_m \\ q_j = 0 \quad \forall x_j \epsilon \overline{\omega}^+_m \end{pmatrix} \tag{5}$$

и программой аппроксимации D

$$\overline{U}^+_{m,\varphi}(x) \overset{D}{\to} \left(\beta_{m,j}\,, \forall x_j \epsilon \overline{\omega}^+_m\right). \tag{6}$$

2. Для всех $e_m \epsilon E$ вычислим потенциалы $\alpha_{m,i,j}$ в каждой точке $x_i \epsilon \overline{\omega}^+_m$ от единичного тестирования q_j на всех $x_j \epsilon \overline{\omega}^+_m$. Для этого воспользуемся программой p

$$\overline{U}^+_{m,i,j}(x) = A\begin{pmatrix} 0 \text{ на } S^-_m\,, \\ q_j = 1,\ x_j \epsilon \overline{\omega}^+_m\ , \\ q = 0\ \ \forall x_k \epsilon \overline{\omega}^+_m, k \neq j \end{pmatrix} \tag{7}$$

и программой аппроксимации

$$\overline{U}^+_{m,i,j}(x) \overset{D}{\to} \left(\alpha_{m,i,j}\,, \forall\left(x_i \text{ и } x_j\right) \epsilon \overline{\omega}^+_m\right). \tag{8}$$

3. Построим систему уравнений для определения параметров взаимосвязи между элементами. Положим для определённости, что элемент e_m имеет два смежных элемента: $e_{m*}(m *< m)$ и $e_{m**}(m **> m)$. При этом величина потенциала $\overline{U}_{(m,n),i}$ в точке $x_i \epsilon \overline{\omega}^+_{m,n}$ зависит от всех q_j при $x_j \epsilon \overline{\omega}^+_{m,n} \cup \overline{\omega}^+_{m,n*} \cup \overline{\omega}^+_{m,n**}$.

$$\bar{U}_{(m,n),i} = \beta_{m,i} + \sum_{x_j \epsilon \bar{\omega}^+_{m,n}} \alpha_{m,i,j}\, q_{(m,n),j} + \sum_{x_j \epsilon \bar{\omega}^+_{m,m*}} \alpha_{m,i,j}\, q_{(m,m*),j} + \sum_{x_j \epsilon \bar{\omega}^+_{m,m**}} \alpha_{m,i,j}\ . \quad (9)$$

Аналогично для $\bar{U}_{(n,m),j}$ элемента e_n с двумя смежными $e_{n*}(n *< n)$ и $e_{n**}(n **> > n)$ будем иметь:

$$\bar{U}_{(n,m),i} = \beta_{n,i} + \sum_{x_j \epsilon \bar{\omega}^+_{n,m}} \alpha_{n,i,j}\, q_{(n,m),j} + \sum_{x_j \epsilon \bar{\omega}^+_{n,n*}} \alpha_{n,i,j}\, q_{(n,n*),J} + \sum_{x_j \epsilon \bar{\omega}^+_{n,n**}} \alpha_{n,i,j}\, q_{(n,n**),j}\ . \quad (10)$$

Используя условия (3), (4) и формулы (9), (10), получим уравнение, связывающее неизвестные, оказывающие влияние на потенциал в каждой точке x_i на поверхности S_{mn}:

$$\sum_{x_j \epsilon \bar{\omega}^+_{m,n}} (\alpha_{m,i,j} + \alpha_{n,i,j)}\, q_{(m,n,),j} - \sum_{x_j \epsilon \bar{\omega}^+_{m*,m}} \alpha_{m,i,j}\, q_{(m,m**),j} + \\ + \sum_{x_j \epsilon \bar{\omega}^+_{m,m**}} \alpha_{m,i,j}\, q_{(m,m**),j} - \sum_{x_j \epsilon \bar{\omega}^+_{n*,n}} \alpha_{n,i,j}\, q_{(n*,n),j} + \quad (11) \\ + \sum_{x_j \epsilon \bar{\omega}^+_{n,n**}} \alpha_{n,i,j}\, q_{(n,n**),j} = \beta_{n,i} - \beta_{m,i}.$$

Условие (4) позволило вдвое уменьшить в (11) число неизвестных: $q_{(r,l),k}$ при $r > l$ заменено на $-q_{(l,r)k}$. Уравнение (11) символически можно записать в математической форме:

$$A_{m,n}\, Q_{m,n} = B_{m,n}. \quad (12)$$

4. Преобразуем совокупность уравнений (12) в систему алгебраических уравнений стандартного вида

$$A^* Q_N = B_N\ , \quad (13)$$

где A^* – квадратная матрица порядка N, Q_N и B_N – матрицы–столбцы соответственно неизвестных и свободных членов, N – общее число неизвестных. Решаем систему (13).

5. Подставляя вычисленные значения $q_{...}$ в (9), находим потенциалы на дискретном множестве точек взаимодействия всех элементов разделения объекта.
6. Воспользовавшись программой *p*, параллельно решаем задачу Дирихле для каждого элемента e_m (m=1,2,…,M).

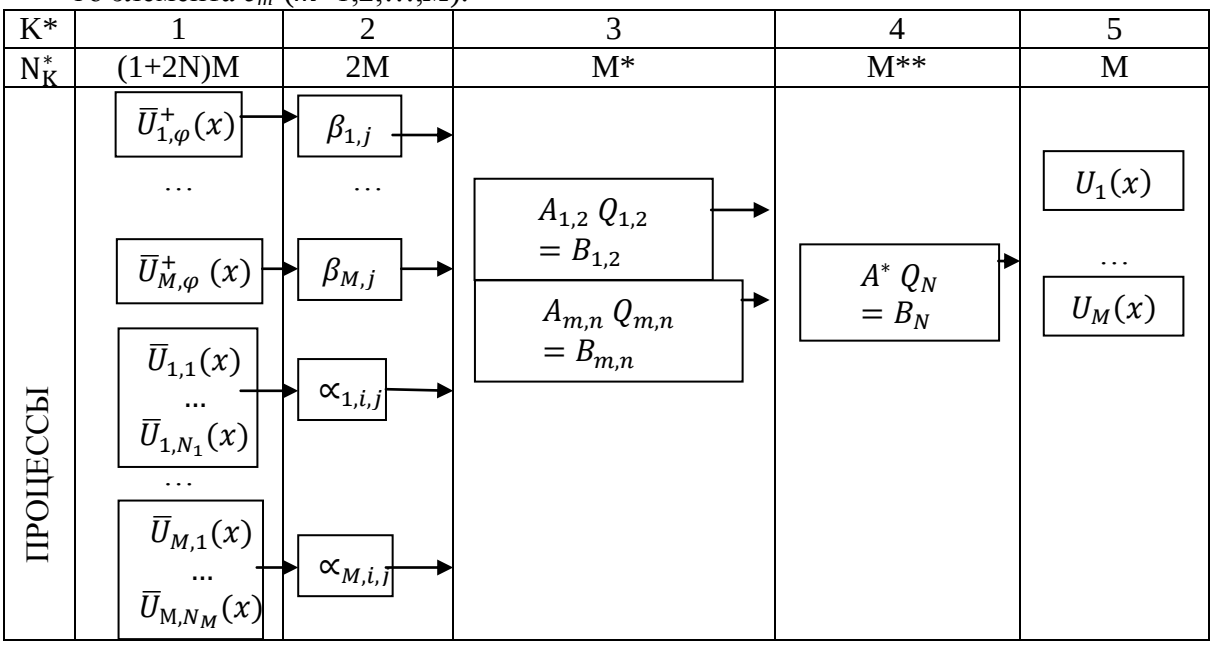


Рис.3. Последовательность решения задачи

Вычислительный процесс решения поставленной задачи можно построить по схеме, указанной на рис. 3, где $K*$ – порядковый номер последовательно выполняемых блоков программы, N^*_K – число параллельных вычислительных процессов в K-ом блоке.

В 1-ом блоке программами p параллельно выполняется $(1+2N)M$ операций вида (5), (7) 1-го и 2-го этапов решения задачи. При этом, используя (1), можно определить время T_1 , затрачиваемое на вычислительные операции блока:

$$T_1 = f(n^*)\,,\ \ n^* = \max_{m\epsilon\overline{M}} n_m\ ,\overline{M} = 1,2,\dots,M\ ,$$

n_m – число неизвестных при решении краевой задачи для элемента e_m .

Во 2-м блоке параллельно выполняется $2M$ операций аппроксимации вида (6), (8). Время выполнения блока

$$T_2 = f_1(N^*)\,, N^* = \max_{m\epsilon\overline{M}} N_m\ ,$$

N_m – количество точек контакта e_m со всеми смежными элементами.

3-й блок (3-й этап решения) в параллельном режиме формирует системы уравнений (12), M^* – число поверхностей разделения элементов. При этом

$$T_3 = f_2(N^{**})\,,\ \ N^{**} = \max_{\forall S_{m,n}} N_{m,n}\ ,$$

$N_{m,n}$ – число точек контакта элементов e_m и e_n .

4-й блок (4-й этап) формирует и решает систему (13) из N линейных алгебраических уравнений. Примем $T_4=f_3(N)$, где f_3 и M^{**} определяются используемой программой решения уравнений.

5-й блок (5-й и 6-й этапы) параллельно выполняет M операций решения задачи Дирихле для элементов $e_1, \dots, e_M$. При этом $T_5=f(n^*)$.

Введём характеристику ускорения

$$\upsilon = \frac{T_0}{T_*},\quad T_0 = f(n),\quad T_* = \sum_{i=1}^{5} T_i\ ,$$

показывающую во сколько раз возрастает (при $\upsilon>1$) производительность вычислений при решении задачи предлагаемым методом.

При разделении Ω не исключён вариант, когда $e_1, \dots, e_M$ будут иметь одинаковые количества неизвестных: $n_1=n_2=\dots=n_M=n/M$. Кроме того, нельзя исключить и такой случай, когда общее число точек связи элементов будет настолько малым, что $T_2+T_3+T_4<<T_1+T_5$ и поэтому $T_*\cong T_1+T_5=2f(n/M)$. Примем $f(\chi) = c\chi^\gamma$, где $c,\gamma>0$ – константы. При этом будем иметь пиковое значение υ:

$$\upsilon_p = \frac{n^\gamma}{2(n/M)^\gamma} = \frac{1}{2}M^\gamma. \qquad (14)$$

При применении расчётной программы “Лира” на ПК (3,2 ГГц, 512 Мб) до 2 млн. неизвестных получена γ=3,8393. Из (14) при M=2 будет υ_p=7. Расчёт на 2-х параллельных ПК при n=2 034 372 уравнений показал υ=5,8.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.
2. Suprun A.N., Kislitsyn D.I. Distributed computing for construction project design by division into project design units // Computing in Civil and Building Engineering, Proceedings of the International Conference, 30 June - 2 July – Nottingham, 2010. P. 325–326.
3. Suprun A.N., Kislitsyn D.I. The multilevel parallelization of structural design calculation in distributed computing environment //14th International Conference on Computing in Civil and Building Engineering (14th ICCCBE), 27 – 29 June – Moscow, 2012. P. 382–383.

ОБМЕН ДАННЫМИ В ПАРАЛЛЕЛЬНОЙ ПРОГРАММЕ РАСЧЕТА СОБСТВЕННЫХ ЧАСТОТ ОБОЛОЧЕК ВРАЩЕНИЯ

В.Л. Тарасов

Нижегородский госуниверситет им. Н.И. Лобачевского

Изложена методика расчета собственных частот оболочек вращения. Предложено распараллеливать вычисления путем раздельного расчета собственных частот для различных значений чисел волн в окружном направлении. Время на пересылку исходных данных на отдельные процессы параллельной программы предложено сокращать за счет предварительного копирования данных из исходного файла в символьный буфер с последующей однократной рассылкой буфера. Приведены результаты численных экспериментов, показывающие сокращение времени вычислений за счет распараллеливания.

Введение

Оболочечные конструкции широко применяются в технике, поэтому различные аспекты их расчета, в том числе и расчета собственных частот неоднократно рассматривались в литературе. В частности, можно указать на работы [1, 2], в которых изложены алгоритмы и приведены программы для расчета оболочечных конструкций. В настоящей работе рассматривается задача расчета собственных частот оболочек вращения с точки зрения ускорения вычислений за счет их распараллеливания.

1. Методика расчета собственных частот

Нахождение собственных частот колебаний осесимметричных оболочечных конструкций сводится к задаче на собственные значения для системы обыкновенных дифференциальных уравнений первого порядка путем разложения в ряд Фурье по окружной координате величин, описывающих напряженно-деформированное состояние оболочки [1]:

$$\boldsymbol{y}' = \boldsymbol{A}(x, \lambda. m)\,\boldsymbol{y}, \quad 0 \le x \le l. \tag{1}$$

Здесь $\boldsymbol{y}$ – вектор из 8 неизвестных, описывающих напряженно-деформированное состояние оболочки, штрих обозначает дифференцирование по меридиональной координате x, $\boldsymbol{A}(x, \lambda, m)$ – квадратная матрица коэффициентов системы, λ – параметр, пропорциональный собственной частоте (безразмерная собственная частота), m – число волн в окружном направлении в форме колебаний, которое может принимать значения 0, 1, 2, … . В качестве основных неизвестных $y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8$ удобно выбрать величины, пропорциональные, соответственно, сдвиговому усилию в срединной поверхности, меридиональному изгибающему моменту, меридиональному усилию, поперечному усилию, перемещению в окружном направлении, углу поворота нормали к срединной поверхности в плоскости меридиана, перемещению в меридиональном направлении и нормальному перемещению точек срединной поверхности.

Граничные условия на «левом» ($x = 0$) и «правом» ($x = l$) торце оболочки можно записать в виде:

$$\boldsymbol{H}_L\,\boldsymbol{y}(0)=0, \quad \boldsymbol{H}_R\boldsymbol{y}(l)=0. \tag{2}$$

Здесь $\boldsymbol{H}_L, \boldsymbol{H}_R$ – матрицы коэффициентов граничных условий размером 4×8.

Система дифференциальных уравнений (1) и краевые условия (2) однородны, поэтому существует тривиальное решение $\boldsymbol{y}(x) \equiv 0$ при любых значениях параметра частоты λ. Собственным частотам соответствуют такие значения λ, при которых однородная краевая задача (1), (2) имеет нетривиальное решение.

Алгоритм поиска собственных значений λ состоит в следующем. Граничные условия слева представляют собой систему из 4-х линейных однородных алгебраических уравнений, в которой неизвестными являются 8 величин $y_i(0)$, $i = \overline{1,8}$. Выберем 4 линейно независимых вектора $\boldsymbol{y}^1(0)$, $\boldsymbol{y}^2(0)$, $\boldsymbol{y}^3(0)$, $\boldsymbol{y}^4(0)$ так, чтобы их компоненты удовлетворяли краевому условию при $x = 0$. Используя выбранные векторы как начальные условия, проинтегрируем численно систему (1) при некотором значении параметра частоты λ, получим 4 линейно независимых частных решения. Составим из этих частных решений линейную комбинацию с произвольными постоянными коэффициентами, которая будет являться общим решением системы (1), удовлетворяющим краевым условиям слева. Это общее решение подставим в краевые условия на правом конце при $x = l$, получим систему уравнений для нахождения произвольных постоянных. В случае расчета собственных частот система для определения постоянных интегрирования является однородной, то есть она будет иметь ненулевое решение при равенстве нулю определителя системы (частотного определителя), величина которого зависит от значения параметра λ, при котором выполнялось численное интегрирование. Значения λ, обращающие в нуль частотный определитель, есть собственные частоты.

Таким образом, определение собственных частот требует многократного интегрирования системы дифференциальных уравнений (1) для различных значений λ, которые надо подбирать так, чтобы частотный определитель обратился в нуль.

Для нахождения собственных частот перебираются значения λ с некоторым шагом до тех пор, пока не будет найден интервал, на концах которого определитель имеет разные знаки. Это признак того, что собственная частота расположена на данном интервале. Уточнение собственной частоты выполняется методом секущих. Чтобы найти интервал, содержащий собственную частоту, приходится выполнять дополнительные предварительные исследования.

2. Распараллеливание вычислений при расчете собственных частот

Для каждого значения параметра частоты λ требуется построить 4 частных решения системы (1). Очевидной является идея строить эти частные решения параллельно, используя 4 процессора. Но рассматриваемая система (1), описывающая напряженно-деформированное состояние тонких оболочек, такова, что такое распараллеливание оказывается невозможным, так как одни частные решения оказываются быстро растущими, а другие быстро убывающими, в результате чего первоначально независимые решения оказываются фактически зависимыми, и частотный определитель обращается в нуль при любых значений λ.

Для борьбы с этим явлением используют метод ортогональной прогонки С.К. Годунова [3], состоящий в том, что частные решения строятся не по отдельности, а совместно, и для преодоления «сплющивания» системы векторов частных решений проводится их ортонормирование через определенное число шагов интегрирования. Таким образом, при реализации метода ортогональной прогонки невозможно распараллеливание для независимого построения частных решений.

Другой возможностью распараллелить вычисления является распараллеливание по волновым числам, так как расчеты собственных частот для различных значений чисел волн в окружном направлении m можно выполнять независимо друг от друга. Данный подход удобен еще и тем, что число волн m явно входит в систему (1). Такое распараллеливание было реализовано в виде программы на языке С с использованием техноло-

гии MPI. Программа позволяет рассчитать собственные частоты оболочки для заданного количества волновых чисел.

3. Пересылка исходных данных на процессы параллельной программы

При разработке программы нужно было решить вопрос о передаче исходных данных, необходимых для расчетов, на все процессы параллельной программы. Геометрию и размеры типичной оболочки вращения описывают около 20 параметров, для которых в программе назначены индивидуальные числовые переменные. Ведущий процесс мог бы прочитать эти параметры из текстового файла и сделать соответствующее число рассылок на остальные процессы программы. Такой подход неудобен тем, что функцию распределения данных пришлось бы вызывать около 20 раз для пересылки небольших порций данных, а это привело бы к значительным затратам времени на рассылку. Уменьшить количество рассылок позволяет следующий алгоритм распределения исходных данных по процессам.

Нулевой процесс открывает входной файл как текстовый и читает количество рассчитываемых частот `MTS`. Если число процессов окажется больше количества частот, программа прерывается. Если число процессов не превышает количества рассчитываемых собственных частот, исходный файл закрывается и вновь открывается как бинарный, что позволяет определить его размер в байтах. Найденный размер файла рассылается всем процессам, участвующим в вычислениях, и используется в качестве размера символьного буфера для временного размещения исходных данных. Нулевой процесс читает входной файл как бинарный в собственный буфер, а затем рассылает его содержимое всем процессам программы функцией MPI_Bcast(…). Процессы создают текстовый входной поток, связывают его с буфером исходных данных и выполняют чтение из буфера в числовые переменные программы.

Набор волновых чисел делится равномерно между процессами на отдельные диапазоны. Каждый процесс вызывает функцию, которая находит собственные частоты для заданного диапазона волновых чисел. Сборка результатов осуществляется на нулевом процессе с помощью функции MPI_Gatherv(…), вызываемой каждым процессом параллельной программы. Нулевой процесс выводит результаты в выходной файл.

Работу программы поясняет табл. 1. Строки таблицы соответствуют шагам работы алгоритма. В ячейках таблицы описаны действия, выполняемые процессами параллельной программы.

Таблица 1. Схема работы программы расчета собственных частот

Процесс 0	Процесс 1	…	Процесс `k`
Чтение `MTS` из `Input.txt`. Проверка `ProcNum <= MTS`			
Чтение `Input.txt` как бинарного в `BUFF`			
Рассылка `BUFF`	Прием `BUFF`	…	Прием `BUFF`
Чтение `BUFF` как текстового потока	Чтение `BUFF` как текстового потока	…	Чтение `BUFF` как текстового потока
Определение диапазона волновых чисел	Определение диапазона волновых чисел	…	Определение диапазона волновых чисел
Расчет собственных частот	Расчет собственных частот	…	Расчет собственных частот
Отсылка результатов процессу 0	Отсылка результатов процессу 0	…	Отсылка результатов процессу 0
Вывод результатов			

4. Результаты расчетов и выводы

На вычислительном кластере ННГУ [5] были выполнены численные эксперименты по расчету собственных частот конической оболочки, сечение которой вдоль оси симметрии приведено на рис. 1. Для тестирования программы рассматривалась оболочка со следующими исходными данными: угол при большем основании $\beta = 60^0$, отношение радиуса меньшего основания к среднему радиусу $r/R_{cp} = 1/3$, отношение толщины оболочки к среднему радиусу $H/R_{cp} = 0.005$, коэффициент Пуассона материала оболочки $\nu = 0.3$. В качестве меридиональной координаты бралось отношение расстояния *s*, отсчитываемое вдоль меридиана, к среднему радиусу: $x = s/R_{cp}$. Торцы оболочки предполагались жестко защемленными.

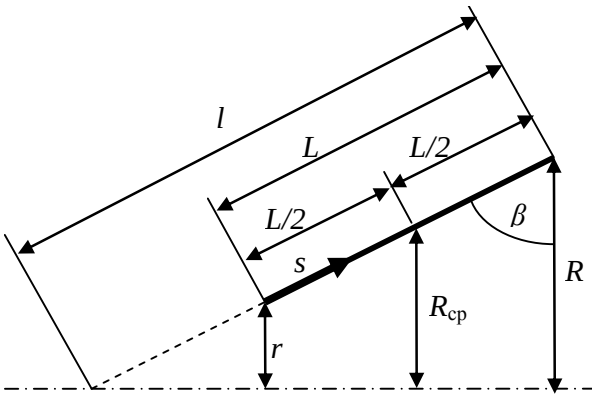


Рис. 1. Геометрические параметры усеченной конической оболочки

В приводимой далее формуле для частоты λ использованы следующие обозначения: *E* – модуль упругости материала оболочки, *ρ* – плотность материала, ω – круговая частота

$$\lambda = \omega R_{cp}\sqrt{\rho\frac{1-\nu^2}{E}}.$$

В табл. 2 проводится сравнение безразмерных собственных частот, вычисленных с помощью обсуждаемой программы, с результатами работы [6] для 6 чисел волн. Это сравнение убеждает в правильности работы программы.

Таблица 2. Сравнение параметра частоты с результатами работы [6]

m	3	4	5	6	7	8
λ	0.1629	0.1137	0.09358	0.09043	0.09373	0.09990
λ [6]	0.1635	0.1145	0.09426	0.09078	0.09387	0.09987
Разница, %	0.37	0.70	0.72	0.39	0.15	0.03

Результаты экспериментов по определению влияния числа используемых процессоров на время вычислений приведены в табл. 3. Видно, что ускорение пропорционально числу задействованных процессоров, но коэффициент пропорциональности, то есть эффективность [5], составляет только 0.5. Тем не менее, можно сделать вывод, что распараллеливание вычислений при расчете собственных частот дает существенное сокращение времени расчетов.

Таблица 3. Время вычислений и ускорение при расчете собственных частот

Количество частот	Время расчета на одном процессоре	Число процессоров	Время расчета	Ускорение
MTS = 6	T1 = 0.21 сек	6	T6 = 0.070 сек	T1 / T6 = 3
MTS = 12	T1 = 0.6 сек	12	T12 = 0.1 сек	T1 / T12 = 6

Литература

1. Кармишин А.В., Лясковец В.А., Мяченков В.И., Фролов А.Н. Статика и динамика тонкостенных оболочечных конструкций.– М.: Машиностроение, 1975. – 376 с.
2. Мяченков В.И., Григорьев И.В. Расчет составных оболочечных конструкций на ЭВМ: Справочник.– М.: Машиностроение, 1981.– 216 с.
3. Годунов С.К. О численном решении краевых задач для систем линейных обыкновенных дифференциальных уравнений // Успехи матем. наук. 1961. Т. 16, № 3. С. 171–174.
4. Тарасов В.Л. Распараллеливание вычислений при расчете собственных частот оболочек вращении // Труды IX Всероссийской научной конференции «Нелинейные колебания механических систем».– Нижний Новгород: Издательский дом «Наш дом», 2012. – С. 905–914.
5. Интегрированная среда высокопроизводительных вычислений ННГУ – [http://cluster.software.unn.ru].
6. Кольман Э.Р., Силкин В.Б. Свободные колебания конической оболочки при различных граничных условиях // Расчеты на прочность.– М.: Машиностроение, 1968, вып. 13, С. 251-273.
7. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. 423 с.

ОБЪЕМНАЯ ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ГАЗОДИНАМИЧЕСКОГО МОДЕЛИРОВАНИЯ МЕТОДОМ ПРОЕЦИРУЕМЫХ ТЕТРАЭДРОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

В.В. Титов

GDT Software Group, Тула

E-mail: titov@cfd.ru

В статье описывается применение метода проецируемых тетраэдров для интерактивной объёмной визуализации результатов газодинамического моделирования. В отличие от предыдущих работ, предложенный в статье метод опирается на использование технологии параллельных вычислений CUDA. Использование вычислительных возможностей графической карты для проецирования ячеек и сортировки позволяет достичь интерактивности при визуализации скалярных данных на структурированной сетке. На текущем графическом оборудовании представленный алгоритм позволяет отображать данные со скоростью 23 млн тетраэдров в секунду.

Введение

Современные методы анализа результатов численного моделирования сложных нестационарных газо/гидродинамических процессов, проблем механики деформируемого твёрдого тела не обходятся без визуального отображения информации. Визуальное представление анализируемых данных может иметь различный характер в зависимости от размерности расчётной области, визуализируемых параметров, требования наглядности представления информации. Наиболее общим и часто используемым графическим представлением информации в трёхмерном пространстве является объёмная визуализация скалярных данных. При этом скалярное поле отражает значение какого-либо конкретного параметра: плотность, давление, температура и т.д. - в каждой точке доступного объёма. Скалярное поле обычно представляется в виде цвета и прозрачности, значения которых определяются с использованием, так называемой, передаточной функции. В представляемой работе рассматривается визуализация скалярного поля, заданного на неструктурированной сетке, каждая ячейка которой представляет собой тетраэдр.

Существует много различных способов создания объёмной визуализации: проецирование ячеек, трассировка лучей, использование заметающей плоскости [1].

Подход, используемый в статье, базируется на методе проецируемых тетраэдров, описанном Shirley и Tuchman [2]. Также были применены методики классификации тетраэдров и расчёта цвета и прозрачности из [3].

1. Метод проецируемых тетраэдров

В основе метода лежит проецирование тетраэдров, составляющих расчётную сетку, в экранную область и их отображение в порядке видимости.

Каждый спроецированный тетраэдр преобразуется в набор треугольников. В процессе проецирования тетраэдр относится к одному из классов в зависимости от его положения в пространстве относительно наблюдателя. Классы разделяются по количеству получаемых треугольников. Среди вершин треугольников выделяют «толстые» и «тон-

кие» по длине луча проецирования, проходящего через первоначальный тетраэдр. «Толстой» является вершина, для которой длина луча максимальна. Для этой вершины определяются координаты точек входа и выхода луча из тетраэдра, длина луча (l) и значения скалярной величины (sf, sb). «Тонкие» вершины попадают в узлы ячейки. Для этих вершин длина луча равна нулю, а значения скалярной величины и координат известны.

Далее выполняется растеризация полученных треугольников и для каждого фрагмента (пиксела) треугольника выполняется интерполяция параметров sf, sb, l, преобразование скалярных величин в цвет с использованием передаточной функции и вычисление цвета и прозрачности фрагмента с использованием интеграла интенсивности излучения. Результирующие цвет и прозрачность помещаются в цветовой экранный буфер для получения конечного изображения с использованием выражения:

Inew = Color + (1 – Alpha) x I,

где I – значение цвета в буфере, Color, Alpha – рассчитанные для текущего фрагмента значения цвета и прозрачности.

2. Реализация метода с использованием CUDA

Разработанная реализация метода была разделена на несколько этапов. На первом этапе для каждого тетраэдра выполняется проецирование. На втором этапе выполняется сортировка тетраэдров по глубине. На третьем формируется массив индексов для OpenGL примитивов. На четвёртом – выполняется отрисовка примитивов с использованием шейдеров.

2.1. Проецирование тетраэдров

На этапе проецирования для каждого тетраэдра осуществляется вычисление параметров «толстой» вершины, глубины центра, класс и количество вершин в проекции. Реализация этапа выполнена с использованием технологии CUDA, при которой для решения задачи используется очень большое количество параллельно выполняемых нитей, при этом обычно каждой нити соответствует один элемент вычисляемых данных. В нашем случае, каждой нити соответствует один обрабатываемый тетраэдр. Набор инструкций, выполняемых нитью, называется ядром.

В нашей реализации этапа проецирования ядро производит следующие действия:

- загрузка координат узлов тетраэдра из глобальной памяти;
- проецирование координат узлов в экранную область и вычисление глубины центра тетраэдра;
- классификация тетраэдра и определение количества вершин в проекции;
- сортировка вершин проекции;
- вычисление параметров пересечения спроецированных граней;
- вычисление координат «толстой» вершины и длины луча в этой точке;
- вычисление скалярной величины для «толстой» вершины;
- формирование выходных данных.

После выполнения вычислений часть данных сохраняется в массивы в памяти GPU для дальнейшей обработки с использованием CUDA, а часть сохраняется в буферный объект для отрисовки.

2.2. Сортировка по глубине

Для правильного отображения полупрозрачных объектов трёхмерной сцены необходима их сортировка по глубине. Может использоваться как сортировка от ближних к дальним, так и обратная – от дальних к ближним. Выбор зависит от установленной схемы смешивания цветов. В нашем случае была выбрана сортировка от ближних к

дальним. Для сортировки была использована функция сортировки по ключу библиотеки Thrust, входящей в состав CUDA Toolkit. Библиотека Thrust позволяет задействовать мощность графической карты для выполнения базовых алгоритмов.

В качестве ключа для функции сортировки стал массив значений глубины центров тетраэдров, а значениями для сортировки стали порядковые номера тетраэдров.

При выполнении сортировки были исключены обмены CPU-GPU, так как все данные были полностью подготовлены на GPU.

2.3. Формирование индексов

Во время визуализации каждый тетраэдр преобразуется в набор треугольников. При этом треугольники удобно заменить на имеющийся в OpenGL примитив GL_TRIANGLE_FUN. Этот примитив задаётся массивом вершин v1, v2, …, vn. Из n вершин формируется n-2 треугольника следующим образом: первый задаётся вершинами v1, v2, v3; второй – v1, v3, v4; *i*-ый – v1, vi+1, vi+2. Для отображения примитивов используется функция glMultiDrawElements, позволяющая выводить последовательности индексных примитивов. Для этой функции необходимо подготовить:

- массив count с количествами выводимых элементов в каждой последовательности, в нашем случае – количество вершин для каждого примитива GL_TRIANGLE_FUN;
- массив pindices с указателями на места расположения индексов для каждой последовательности – у нас этот массив заполняется один раз при загрузке данных, а при каждом выводе изменяются сами индексы.

Подготовка этих массивов перед выводом примитивов была реализована с использованием CUDA. Вычислительное ядро для каждого выводимого примитива выполняет следующие действия:

- выбирается индекс тетраэдра из отсортированного массива;
- считывается информация о классификации тетраэдра;
- в массив count записывается количество вершин для отображаемого вместо тетраэдра примитива;
- в массив indices записываются индексы вершин в соответствии с классом проекции тетраэдра.

После выполнения CUDA-ядра массив count копируется в память CPU, а массив indices отображается в буферный объект в памяти GPU.

2.4. Визуализация примитивов

После того как сформированы массивы индексов можно произвести вывод примитивов. При выводе для хранения информации о вершинах используются буферные объекты. В одном буферном объекте хранится информация о координатах вершин, в другом о значениях скалярного поля и толщине тетраэдров в каждой вершине, в третьем хранятся индексы для ссылки на информацию из первых двух буферных объектов. Для преобразования значений скалярного поля в цвет используется возможность запрограммировать конвейер рисования с использованием языка программирования GLSL. В вершинном шейдере осуществляется обычная обработка по умолчанию, а фрагментный шейдер производит дополнительную обработку для вычисления цвета и прозрачности фрагментов. При этом для текущих значений скалярного поля на передней и задней гранях тетраэдра выполняется выборка цвета из одномерной текстуры палитры, затем производится конечный расчёт цвета и прозрачности в соответствии с предварительно частично рассчитанным интегралом интенсивности излучения.

3. Результаты

Описанный алгоритм был реализован в качестве расширения для пакета научной визуализации Scientific VR на языке C++ с использованием CUDA и GLSL. Для измерения производительности использовалась система с процессором Pentium IV 3.0 ГГц, 4 ГБ ОЗУ и графической картой NVidia GeForce GTX 570 1280МБ. В качестве тестовых наборов данных использовались результаты газодинамического и прочностного моделирования с различным количеством ячеек.

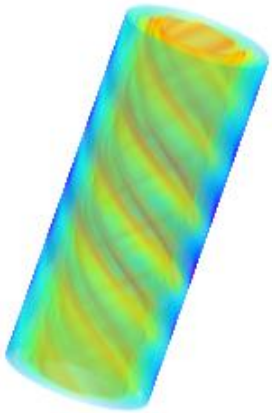


Рис. 1. Объёмная визуализация данных газодинамического моделирования (набор 1)

Рис. 2. Объёмная визуализация данных газодинамического моделирования (набор 2)

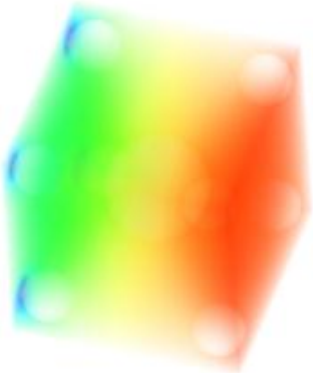


Рис. 3. Объёмная визуализация данных прочностного моделирования (набор 5)

В табл. 1 приведено среднее время исполнения каждого этапа и конечная производительность визуализации.

Таблица 1. Результаты визуализации скалярных данных

	Набор 1	Набор 2	Набор 3	Набор 4	Набор 5
Кол-во узлов, тыс.	768.9	1851	993.5	544.5	129
Кол-во ячеек, тыс.	4374	10764	5760	3198.5	533
1-ый этап, мс	32.35	78.15	40.89	25.81	4.98
2-ой этап, мс	10.16	22.24	12.95	8.74	4.87
3-ий этап, мс	23.94	60.71	30.44	18.10	2.02
4-ый этап, мс	123.13	295.5	158.83	89.73	15.62
Общее время, мс	189.58	456.6	243.11	142.37	27.49
Производительность, млн ячеек/сек	23.072	23.574	23.693	22.465	19.389

4. Заключение

Таким образом, производительность приведённой реализации позволяет формировать изображение объёмной визуализации скалярных данных для сеток с 23 млн ячеек за секунду. Кроме того, было проведено тестирование версии программы, в которой этап отрисовки примитивов выполнен с использованием технологии CUDA: вместо использования конвейера OpenGL и шейдеров на языке GLSL создано CUDA-ядро. Хотя производительность тестовой версии оказалась на порядок ниже - ее предпочтение позволяет реализовать весь алгоритм на CUDA. Такая возможность позволяет использование для визуализации кластера, оснащённого несколькими узлами. Что в свою очередь даст возможность визуализировать сетки со значительно большим количеством ячеек и с большей скоростью.

Литература

1. Johnson C., Hansen C. Visualization Handbook. Academic Press, Inc., Orlando, FL, USA, 2004.
2. Shirley P., Tuchman A. A.: Polygonal approximation to direct scalar volume rendering. In Proceedings San Diego Workshop on Volume Visualization, Computer Graphics. 1990. V. 24, P. 63–70.
3. Marroquim R., Maximo A., Farias R., Esperanca C. GPU-based cell projection for interactive volume rendering. SIBGRAPI: Brazilian Symposium on Computer Graphics and Image Processing. 2006. P. 147–154.

МАКЕТ ТЕЛЕМЕДИЦИНСКОЙ СИСТЕМЫ НОВОГО ПОКОЛЕНИЯ

В.Е. Турлапов, В.П. Гергель

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлены принципы организации и архитектура макета информационной телемедицинской системы нового поколения. Новизна системы определяется тремя главными признаками: 1) организация системы на основе вычислительного кластера, как системы вычислительной медицины или биологии; 2) организация системы на основе большого распределенного долговременного хранилища медицинских и биологических данных, совместимого с HIS, LIS, PACS, RIS; 3) объединение в системе данных и средств медицинской диагностики с системами вычислительного моделирования работы органов человека на клеточном уровне.

Введение

Наиболее актуальным для инновационной медицины сегодня является объединение и медицинских, и биологических исследований вокруг конкретного пациента на основе всестороннего использования ресурсов высокопроизводительных кластеров и организованных при них хранилищ данных, что подтверждают, в том числе, проекты лидера производства суперЭВМ IBM (2008 – использование 3D-модели пациента, как основы его медицинской карты [1], 2012 – моделирование действия лекарств на 3D-аватаре конкретного человека [2]). Дополнительным аргументом является возможность организации на основе такого кластера многочисленных профильных облачных сервисов.

Задаче создания именно такого объединения средств диагностики, вычислительной медицины и вычислительной биологии и созданию на этой основе макета телемедицинской системы нового поколения был посвящен трехлетний (2010-2012) госконтракт.

Архитектура созданного макета телемедицинской системы

Ядром системы является гетерогенный вычислительный кластер с Интернет-доступным распределенным хранилищем медицинских диагностических данных, в состав которого могут входить также базы данных медицинских учреждений (HIS, PACS, RIS,...). Хранилище, объемом до 36 терабайт, развернуто на базе гетерогенного вычислительного кластера ННГУ. Сегодня в хранилище находятся около 10 000 томограмм, в основном МРТ, общим объемом 70 гигабайт.

Обеспечение функциональности базы данных (БД) медицинских изображений, а тем более томограмм, предъявляет существенные дополнительные требования: предварительный просмотр слоев томограммы, без ее полного скачивания; пакетное наполнение хранилища на сервере большими объемами данных в формате DICOM с контролем целостности и автоматическим обезличиванием для обеспечения конфиденциальности медицинских данных; упаковка изображений для повышения скорости передачи; интеграция клиента с системой 2D-3D визуализации томограмм.

Все подсистемы макета телемедицинской системы имеют доступ чтение/запись к хранимым данным. Система способна принимать данные от своих подсистем и диагностических приборов: а) терагерцового анализатора (разработка малого предприятия) для экспресс-анализа по выдыхаемому воздуху; б) ультразвукового анализатора крови и биологических жидкостей; в) модели прибора оптической диффузионной томогра-

фии; г) модели электрической активности сердца с учетом его геометрии и моделей активности клеток (программа «Виртуальное сердце»); д) моделей активности ансамблей клеток мозга.

Устройство серверной части

Серверная часть системы (рис.1) состоит из сервера базы данных, хранящего метаданные об изображениях, и одного или нескольких серверов хранения, на которые непосредственно загружаются изображения. В качестве СУБД использована PostgreSQL. В качестве HTTP-сервера использован Microsoft IIS. Выполнением операций над изображениями на сервере хранения управляет сервер приложений. Он реализован как FastCGI-приложение к HTTP-серверу. Для непосредственного хранения данных используется SAS (Serial Attached SCSI) хранилище. В целях масштабируемости его разделы монтируются как подкаталоги в каталог storage без присвоения латинской буквы каждому разделу. FastCGI-приложение при импорте томограмм упаковывает серии изображений слоев в формате DICOM в единый файл и без сжатия размещает в подкаталогах “volume001”, “volume002” и т.д. в зависимости от наличия свободного пространства. В качестве пути к томограмме на сервере хранения в базу данных заносится путь относительно каталога storage, например «/volume001/tom245».

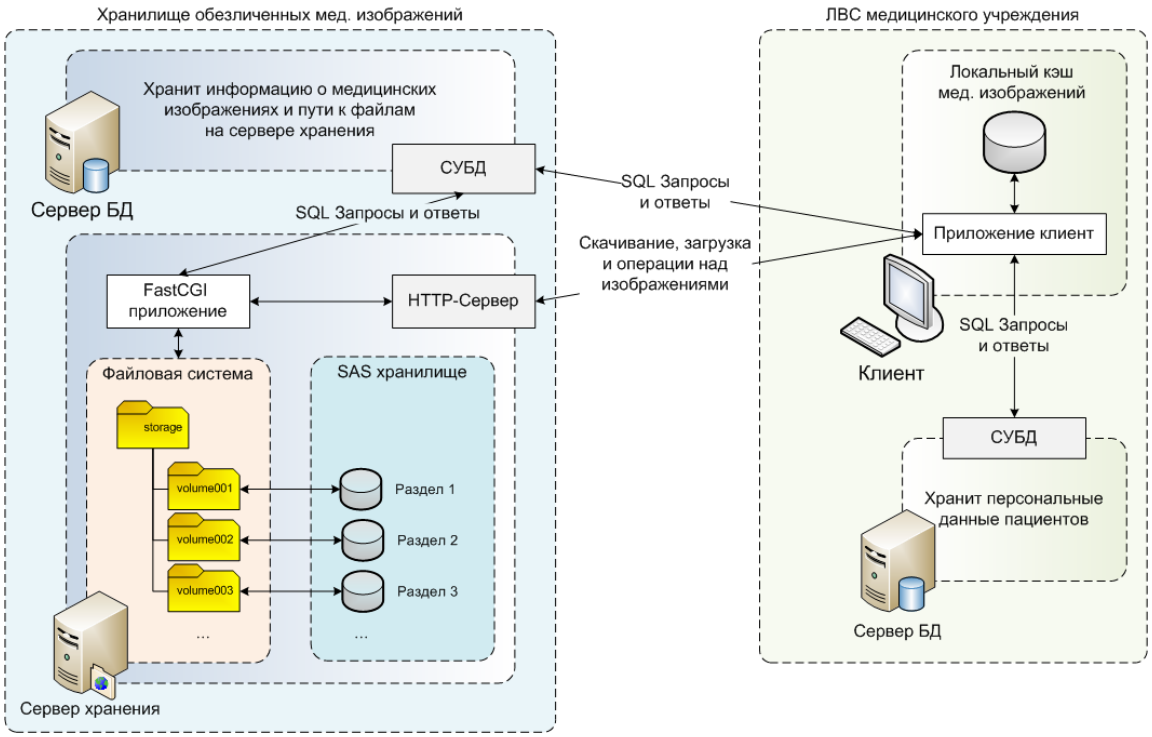


Рис. 1. Архитектура компонентов системы хранения и обработки медицинских изображений

Ускорение скачивания/загрузки изображений

При передаче томограммы целиком производится сжатие без потерь с использованием алгоритма DEFLATE. Сжатие с максимальной степенью компрессии позволяет уменьшить объем передаваемых данных более, чем в два раза, при незначительных (по сравнению со временем передачи) накладных расходах на сжатие/распаковку. Это позволят удвоить скорость передачи снимков между сервером и клиентом. При передаче одиночного слоя для целей предварительного удаленного просмотра слоев томограммы через браузер производится его кодирование в формат PNG (сжатие без потерь тем же

алгоритмом DEFLATE) с уменьшением глубины цвета до 8 бит. Для ускорения повторного доступа реализована подсистема локального дискового кэширования на стороне клиента. Кэш представляет собой специально выделенный каталог, содержащий подкаталоги/файлы. Максимальный объем кэша задается в настройках программы. Кэш работает по принципу вытеснения объектов. Имя каждого подкаталога/файла состоит из префикса, указывающего тип объекта кэша, и уникального числового кода. Это позволит в будущем при необходимости расширить список объектов кэшировании. Для серии изображений одной томограммы в формате DICOM запись в дисковом кэше представляет собой подкаталог с файлами, которые содержат отдельные слои. В качестве имени подкаталога используется префикс “tom” и числовой идентификатор соответствующей записи в базе данных. Такой формат хранения позволяет открывать изображения для чтения непосредственно из кэша.

Интерфейс клиентского приложения и интеграция подсистем

При пакетном импорте кроме записей о самих изображениях в базе данных создаются и все связанные записи (пациент, исследование, часть тела и др.). Чтобы не передавать большое количество данных, записи выводятся постранично, по умолчанию по 20 записей. Возможен переход между соседними страницами и на указанную страницу, сортировка записей по любому полю. Благодаря реализации сервера приложений в клиентскую добавлена выборка и предварительный просмотр отдельных слоев томограммы до того, как скачивать томограмму целиком (см. рис. 2).

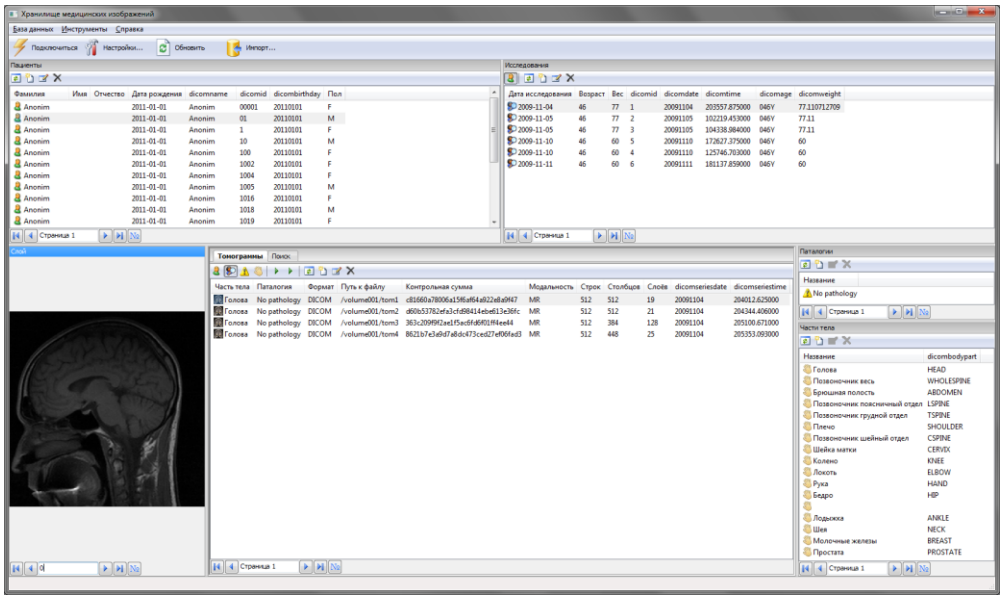


Рис. 2. Копия окна клиентского приложения

3D-визуализатор позволяет просматривать томограммы в различных режимах визуализации и его запуск осуществляется из окна клиента после выбора томограммы в списке (см. пример на рис. 3).

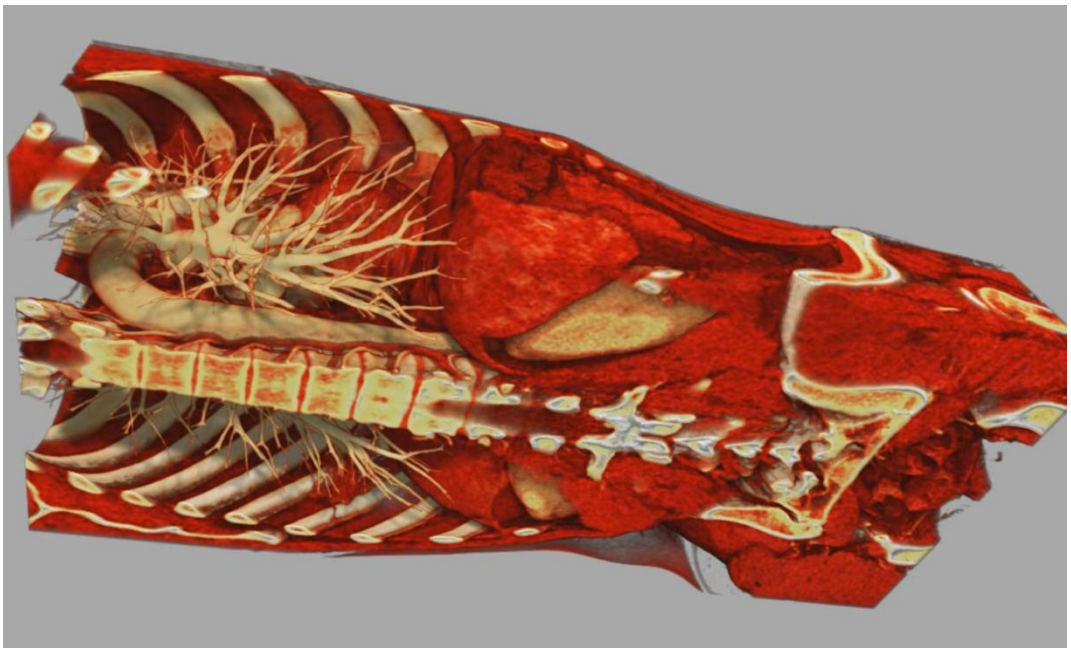


Рис. 3. Пример 3D-визуализации томограммы

Сложность интеграции профильных подсистем сильно разнится от простого запуска процесса до реализации специальных окон диалога условий параллельных вычислений, подготовки входных и анализа выходных данных. Примером сложной интеграции является подсистема оптической диффузионной томографии (программа XMCML). Компонент XMCML активируется как консольное приложение.

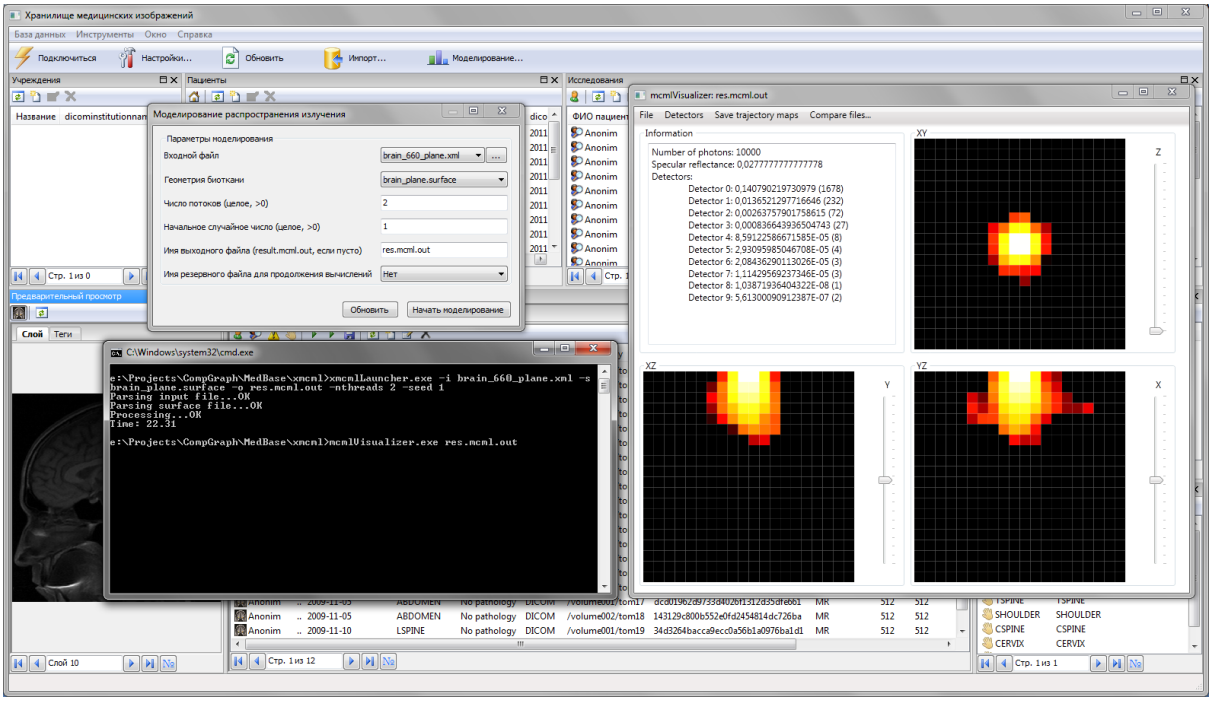


Рис. 4. Подключенная подсистема оптической диффузионной томографии. Результат моделирования распространения зондирующего излучения в мозге

В качестве входа оно принимает XML-файл с описанием параметров задачи. В качестве результата создается сохраняемый выходной файл, содержимое которого отображено (рис. 4) посредством GUI-приложения mcmlVisualizer.exe. Файлы, необходимые приложению, размещены в каталоге ../xmcml (относительно рабочего каталога клиен-

та). Для интегрирования подобных компонентов в программную оболочку клиента создано отдельное окно, позволяющее задавать условия запуска: файл с параметрами задачи; файл, описывающий геометрию зондируемого объекта; число вычислительных потоков на кластере, имя выходного файла и др.

Заключение

Итогом работы является создание действующего макета телемедицинской системы нового поколения со следующим, интересным для врачей-диагностов и клиницистов, набором свойств и характеристик:

- Ядром системы является гетерогенный вычислительный кластер с Интернет-доступным распределенным долговременным хранилищем медицинских диагностических данных, хранящем сегодня около 10 000 томограмм.
- Система способна выполнять также функции центра вычислительной медицины и вычислительной биологии, значительно дополняя и расширяя функции телемедицинской системы, обеспечена необходимым функционалом для ее практической эксплуатации и построения на ее основе телемедицинских и других систем исследовательской и практической медицины.
- Впервые системы прямой медицинской диагностики объединены с системами моделирования и исследования отдельных важнейших органов человека сердца и мозга, использующими вычислительные возможности кластера для постепенного сближения исследования человека на макро- и микроуровнях, раскрытия глубинных причин болезней и методов их лечения и профилактики.
- Система имеет 3D-визуализатор томограмм, не уступающий по качеству, производительности и функциональности зарубежным аналогам [3].

Планируется создание на основе данного макета системы для сопровождения процесса лечения в клинике ядерной медицины: диагностики; назначения на лечение в стационар; собственно графика лечения с наблюдением и анализом его результативности по данным ПЭТ+КТ.

Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839 и гранта Президента РФ НШ-1960.2012.9.

Литература

1. Charette R.N. Visualizing Electronic Health Records With "Google-Earth for the Body" (IBM researchers develop 3-D visualization tool for electronic health records) Jan. 2008. (http://spectrum.ieee.org/biomedical/diagnostics/visualizing-electronic-health-records-with-googleearth-for-the-body)
2. Charette R.N. Using Avatars to Understand Adverse Drug Reactions / March 06, 2012. (http://spectrum.ieee.org/riskfactor/biomedical/diagnostics/using-avatars-to-understand-adverse-drug-reactions/).
3. Гаврилов Н.И., Турлапов В.Е. Подходы к оптимизации GPU-алгоритма volume raycasting для применения в составе виртуального анатомического стола // Научная визуализация. 2012. Кв. 2. Т. 4. № 2. С. 21–56. – [http://sv-journal.com/2012-2/index.php?lang=ru].

КВАНТОВОХИМИЧЕСКИЕ ВЫЧИСЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНО-АППАРАТНОЙ АРХИТЕКТУРЫ CUDA

И.И. Устюгов, Л.В. Парфенова, Л.М. Халилов

Институт нефтехимии и катализа РАН, Уфа

Сочетание теории функционала плотности (DFT) в задачах квантовой химии с возможностью применения мощных графических ускорителей NVIDIA и программно-аппаратной архитектуры CUDA может обеспечить беспрецедентную производительность и точность в расчетах сложных молекулярных систем.

Введение

Современные квантовохимические задачи предъявляют все большие требования к вычислительным возможностям центральных процессоров (Central Processing Unit, CPU). С развитием индустрии видеоигр произошел большой скачок в технологиях производства мощных графических ускорителей NVIDIA. В связи с этим, стало возможным их применение в высокопроизводительных вычислениях (High Performance Computing, HPC).

Графические ускорители, такие как nVidia GeForce 8800GTX, могут быть охарактеризованы как потоковые процессоры [1]. Потоковая обработка – это обобщение архитектуры SIMD (Single Instruction Multiple Data, «одна инструкция – много данных») [2], которая впервые была реализована в суперкомпьютере Cray-1 [3]. Принцип SIMD-архитектуры заключается в том, что приложения объединяются в потоки (streams) и ядра (kernels), представляя собой блоки данных и преобразования кода соответственно. Потоки данных обрабатываются параллельно на нескольких процессорах, используя небольшое количество ядер (порядка 1-2).

В данном примере nVidia 8800GTX имеет 128 потоковых процессоров, объединенных в 16 мультипроцессорных единиц, каждый из которых включает в себе 8 устройств обработки данных. Тактовая частота 1 потокового процессора 1.35 ГГц. Таким образом, суммарная мощность потоковых процессоров nVidia 8800GTX составляет 172.8 ГГц, что сравнимо с традиционными научными расчетными кластерами, в которых используются CPU, например, серверные процессоры AMD Opteron.

На сегодняшний день стоимость графической карты NVIDIA значительно ниже стоимости одного CPU, который используется в научных расчетных кластерах. Это обуславливает экономическую выгоду использования графических ускорителей в квантовохимических расчетах.

Результаты и обсуждение

Ранние попытки использования графических процессоров для «неграфических» вычислений [4-6] были ограничены точностью и сложностью их программирования. Эти проблемы были решены в более современных графических ускорителях, так как они стали поддерживать арифметические операции с 32-битными числами с плавающей точкой (10^{-38}... $\approx 10^{38}$, числа с одинарной точностью). Следующие поколения графических ускорителей и потоковых процессоров NVIDIA и AMD расширили эту под-

держку до арифметики 64-битных чисел с плавающей точкой (10^{-308}... $\approx 10^{308}$, числа с двойной точностью).

Проблему сложности программирования графических процессоров (GPU, Graphical Processing Unit) решила корпорация NVIDIA, представив программно-аппаратную архитектуру CUDA (Compute Unified Device Architecture). Эта архитектура предоставила разработчикам простой программируемый интерфейс для написания исходного кода на языке «Си» (C). С развитием CUDA многие другие языки программирования (такие как Fortran, C#, C++, и т.д.) получили поддержку данной архитектуры.

Результаты расчетов на модельных соединениях (кофеин, холестерол, фуллерен C_{60}, таксол, ваниломицин, CLN025 – «искусственный протеин» [7], олестра) наглядно представлены в работах [3, 8]. В качестве сравнения приводились результаты, полученные в программе *GAMESS v11 Apr 2008 (R1)* на одноядерном CPU Intel Pentium D 3 ГГц.

В расчетах с использованием итеративного алгоритма прямого метода самосогласованного поля (прямой SCF-метод, базис 6-31G) на видеокарте nVidia GeForce 280GTX, nVidia G80 и трех видеокартах nVidia GeForce 280GTX получены впечатляющие результаты [8] (табл. 1).

Таблица 1. Результаты квантовохимического бенчмарка формирования матрицы Фока

		Коэффициент ускорения GPU по сравнению с CPU		
Модельное соединение	**Процессорное время GAMESS, с**	**nVidia G80**	**nVidia 280GTX**	**3 x nVidia 280GTX**
кофеин ($C_8N_4H_{10}O_2$)	7.6	19	28	42
холестерол ($C_{27}H_{46}O$)	113.9	40	56	120
фуллерен (C_{60})	589.9	37	57	155
таксол ($C_{45}NH_{49}O_{15}$)	476.8	46	64	145
ваниломицин ($C_{54}N_6H_{90}O_{18}$)	1226.3	67	90	222
CLN025, «искусственный протеин» ($C_{62}N_{11}H_{97}O_{32}$)	2274.7		92	225
олестра ($C_{156}H_{278}O_{19}$)	14079.2	199	352	696

Результаты приведены для первой, наиболее затратной по времени итерации алгоритма прямого SCF-метода. На GPU использована арифметика 32-битных чисел с одинарной точностью.

Из табл. 1 видно, что при использовании GPU в квантовохимических расчетах дает более чем 100-кратный прирост производительности.

В настоящее время в квантовой химии широкое распространение получила теория функционала плотности (Density Functional Theory, DFT). На данный момент DFT применяется для расчетов энергий связи в молекулах, их минимальной энергии, а так же в физике твердого тела.

Сверхпроводимость, релятивистские эффекты в тяжелых элементах и атомных ядрах, классические жидкости, магнитные свойства сплавов – все это было изучено с помощью теории функционала плотности [9].

Популярность данного подхода обуславливается меньшим временем расчетов (по сравнению с другими квантовохимическими методиками: SCF, MP2, MP3, и т.д.) с незначительным увеличением погрешности вычислений. Следовательно, DFT в сочетании с программно-аппаратной архитектурой CUDA обеспечит беспрецедентную производительность и точность.

Таким образом, на данный момент CUDA представляет собой простую, удобную, быструю и экономически выгодную архитектуру программирования для квантовохимических DFT-вычислений сложных молекулярных систем.

Литература

1. Kapasi U.J., Rixner S., Dally W.J., Khailany B., Ahn J.H., Mattson P., Owens J.D. Programmable Stream Processors // Computer 2003. 36, 54.
2. Russel R.M. The Cray-1 Computer System. Comm // ACM. 1978. 21, 63.
3. Ufimtsev I.S., Martinez T.J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation // J. Chem. Theory Comput. 2008. 4, 2.
4. Fatahalian K., Sugerman J., Hanrahan P. Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication. In Graphics Hardware; Akenine-Moller, T., McCool, M., Eds.; A.K. Peters: Wellesley, 2004.
5. Hall J., Carr N., Hart J. Cache and Bandwidth Aware Matrix Multiplication on the GPU. – University of Illinois Computer Science Department Web Site, 2003. [http://graphics.cs.uiuc.edu/jch/papers/UIUCDCS-R-2003-2328.pdf].
6. Bolz J., Farmer I., Grinspun E., Schroder P. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid // ACM Trans. Graph. 2003, 22, 917.
7. Honda S., Akiba T., Kato Y.S. et al. Am. Chem. Soc. 2008, 130, 15327.
8. Ufimtsev I.S., Martinez T.J. Quantum Chemistry on Graphical Processing Units. Direct Self-Consistent-Field Implementation // J. Chem. Theory Comput. 2009. 5. P. 1004–1015.
9. Klaus C. A Bird’s-Eye View of Density-Functional Theory // Brazilian Journal of Physics, 2006. 36. 4A. P. 1318–1341.

МЕТОД РАСПАРАЛЛЕЛИВАНИЯ ПРОГОНКИ НА ГИБРИДНЫХ ЭВМ

А.А. Федоров, А.Н. Быков, Е.А.Сизов

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Современные тенденции в разработке и создании суперкомпьютеров состоят в том, что качественно более высокая производительность их достигается на путях перехода к ЭВМ с множественными вычислительными процессами на основе параллельной обработки информации. Успешное использование большой потенциальной мощи таких ЭВМ для решения одной задачи возможно лишь после разработки прикладного программного обеспечения, учитывающего параллельную обработку информации.

В институте теоретической и математической физики (ИТМФ) РФЯЦ-ВНИИЭФ для решения научно-технических и технологических задач разработаны различные комплексы программно-ориентированных прикладных программ. Один из них основан на эйлерово-лагранжевой методике РАМЗЕС-КП численного решения трехмерных нестационарных задач газовой динамики с учетом теплопроводности. Конечно-разностная аппроксимация дифференциальных уравнений, используемая в методике, сводит задачу нахождения численного решения к необходимости решения систем линейных алгебраических уравнений (СЛАУ) специального (трехдиагонального) вида. Для нахождения решения таких СЛАУ используется метод прогонки.

Появление в последние годы параллельных ЭВМ с арифметическими ускорителями сделало актуальной задачу реализации методов решения СЛАУ с помощью прогонки на параллельных ЭВМ с такой архитектурой. Здесь мы рассмотрим один из методов решения прогонки на параллельных ЭВМ – метод, предложенный Н.Н. Яненко. Программа, реализующая этот метод на гибридных ЭВМ, написана в комплексе РАМЗЕС-КП на языке программирования ФОРТРАН с использованием функций библиотеки MPI и библиотеки CUDA для языка ФОРТРАН.

Программа тестировалась на ядрах ЦП Intel Xeon X5670 и АрУ NVIDIA Tesla C2050. В качестве тестовой рассматривалась задача о бегущей тепловой волне. На отрезке

$$0\ \text{см} \le x \le 1\ \text{см},$$

на котором находятся два вещества, в начальный момент времени задана температура

$$T(x,0) = 0\ \text{КэВ}.$$

На границах навязываются следующие условия:

$$T(x,t) = 10\sqrt{t},\ x = 0\text{см}$$

$$T(x,t) = 0,\ x = 1\text{см}.$$

Протекающий процесс описывается нелинейным уравнением теплопроводности

$$\frac{\partial E}{\partial t} = \frac{1}{\rho} div(\chi \cdot gradT)$$

со следующими характеристиками:

$$E = T, \chi = 0.5T^2, \quad \text{при } 0\,\text{см} \le x \le 0.5\ \text{см};$$

$$E = 0.1T, \chi = 5T^2, \quad \text{при } 0.5\,\text{см} \le x \le 1\ \text{см}.$$

Задача имеет аналитическое решение. Она решалась на 1, 15, 45, 90 ускорителях.

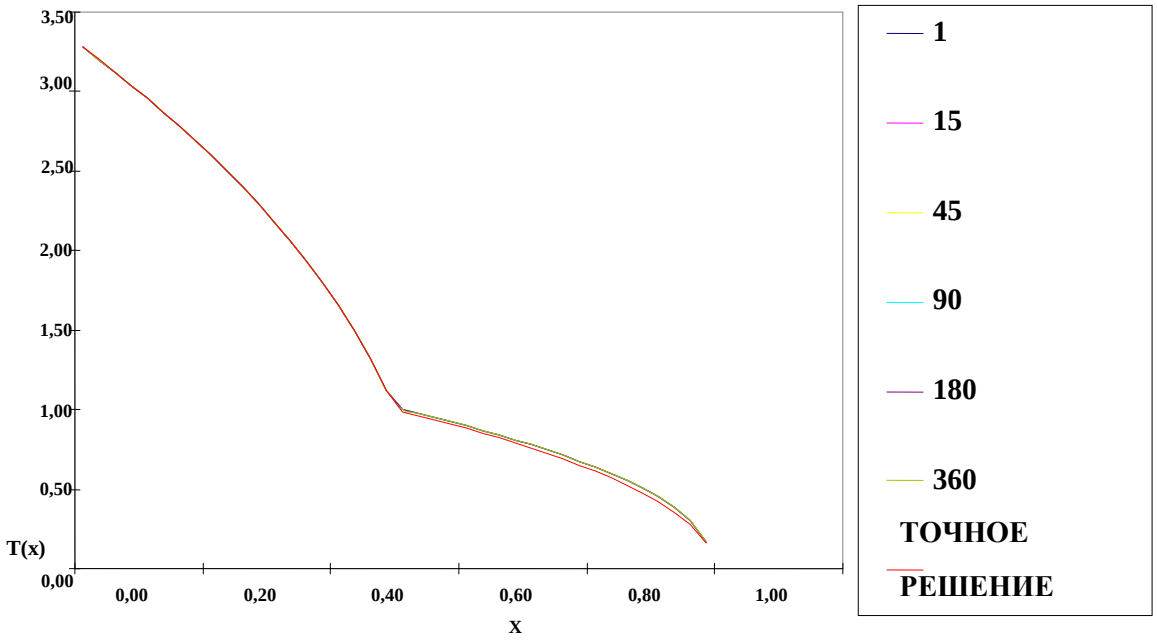


Рис. 1. Графики решений тестовой задачи на разном числе ускорителей и точного решения на момент времени 0.111 сек

При этом исследовались время работы алгоритма, ускорение и эффективность. Также сравнивалось время работы алгоритма на гибридных ЭВМ и ЭВМ без арифметических ускорителей. Из этих данных можно сделать вывод, что использование ускорителей позволяет уменьшить время работы программы до 5 раз. Однако такое ускорение на гибридных ЭВМ недостаточно хорошо. Это связано с тем, что основное время занимает передача данных на АрУ и обратно, но при включении данного алгоритма в комплекс программ РАМЗЕС-КП отпадет необходимость большинства перекачек данных, так как они уже будут храниться в памяти АрУ. Также с увеличением числа ускорителей возрастает время коммуникаций между процессорами, что приводит к снижению эффективности распараллеливания.

В дальнейшем по данному методу предполагается:

1. Попробовать реализовать параллельно-конвейерный метод при решении прогонки на втором этапе.

2. Применить двумерную декомпозицию, что уменьшит число процессоров в прогонке и должно повысить эффективность распараллеливания.

3. Внедрение разработанных программ в эталон комплекса РАМЗЕС-КП, что позволит перенести расчет коэффициентов трехточечных уравнений на АрУ и избавиться от двух этапов перекачки данных между ЦП и АрУ.

4. Кроме этого, предполагается попробовать реализовать на гибридных ЭВМ параллельно-конвейерный метод, используемый сейчас в эталоне комплекса РАМЗЕС-КП.

Литература

1. Быков А.Н., Веселов В.А., Воронин Б.Л., Ерофеев А.М. Методика РАМЗЕС-КП для расчета пространственных движений многокомпонентных теплопроводных сред в эйлеро-лагранжевых координатах. – Сб. Труды РФЯЦ-ВНИИЭФ. Саров, вып. 13, 2008. С. 50–57.

2. Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В. Об организации параллельных вычислений и «распараллеливании» прогонки. – Сб. Численные методы механики сплошной среды. Новосибирск, 1978. Т. 9, № 7. С. 139–146.
3. Рыбакин Б.П. Параллельное программирование для графических ускорителей. М.: НИИСИ РАН, 2011. С. 49–56, 69–70.

УКРУПНЕННАЯ СХЕМА ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ ПРИ РАСЧЕТАХ СЛОЖНЫХ СТРОИТЕЛЬНЫХ КОНСТРУКЦИЙ МЕТОДОМ РАЗДЕЛЕНИЯ НА ПРОЕКТНЫЕ ЕДИНИЦЫ

В.Е. Хромых, Д.И. Кислицын

Нижегородский государственный архитектурно-строительный университет

Автоматизация проектирования конструкций в настоящее время выполняется, в основном, на персональных компьютерах. При этом имеющихся ресурсов ПК при проектировании сложных строительных объектов может не хватить не только на построение автоматизированного согласования проектных решений, но и на выполнение конструкторских работ. Выходом из сложившейся ситуации является применение технологии распределенных вычислений в компьютерных сетях.

В настоящее время в ННГАСУ разработана вторая версия программного комплекса «Решатель», позволяющая в соответствии с методом разделения объекта на проектные единицы (ПЕ) [1, 2] значительно сократить время на расчет конструкции.

«Решатель» работает в двух режимах: «ГИП» и «Проектировщик». Каждому проектировщику соответствует своя ПЕ. ГИП создает проект и делит модель на определенное количество ПЕ, за счет чего реализуется распараллеливание вычислений, основная часть которых выполняется программой «Лира».

Из рис. 1 видно, что укрупненная схема работы «Решатель» представлена в виде трех взаимосвязанных между собой блоков.

В блоке 1 происходит подготовка данных перед проведением расчетов, т.е. сбор необходимой информации от проектировщиков, которые записываются и в дальнейшем хранятся в базе данных (БД).

Когда все необходимые данные собраны, можно переходить к построению модели на исходном языке «Лиры» и запускать ее на расчет (блок 2), которым управляет система «Решатель». После успешно проведенных расчетов клиент автоматически собирает перемещения от единичных и внешних усилий для «своей» ПЕ, полученные в результате проведенных расчетов, и отправляет их на сервер в БД. Сервер на основе полученных данных формирует систему уравнений и решает ее. Результаты записываются в БД для дальнейшего использования.

В блоке 3 выполняется расчет модели от усилий взаимодействия и проверка полученных результатов.

Подробней рассмотрим реализацию данного блока.

На основе решения системы уравнений сервер формирует коэффициенты и рассылает их клиентам с целью проверить правильность полученных результатов, т.е. сервер формирует для каждой ПЕ XML-файл с усилиями взаимодействия и высылает их соответствующим проектировщикам.

<u>Структура данного XML-файла:</u>

```
<?xml version="1.0" encoding="utf-8"?>
<Fkl>
  <Cipher>rama2</Cipher>
  <MePE>1</MePE>
  <Type Element="Стержень">
```

```
    <N CommNode="1">
      <Position>Right</Position>
      <ID>7</ID>
      <FX>-0.671219</FX>
      <FY>0</FY>
      <FZ>-0.592612</FZ>
      <FuX>0</FuX>
      <FuY>0.00259403</FuY>
      <FuZ>0</FuZ>
    </N>
  </Type>
</Fkl>
```

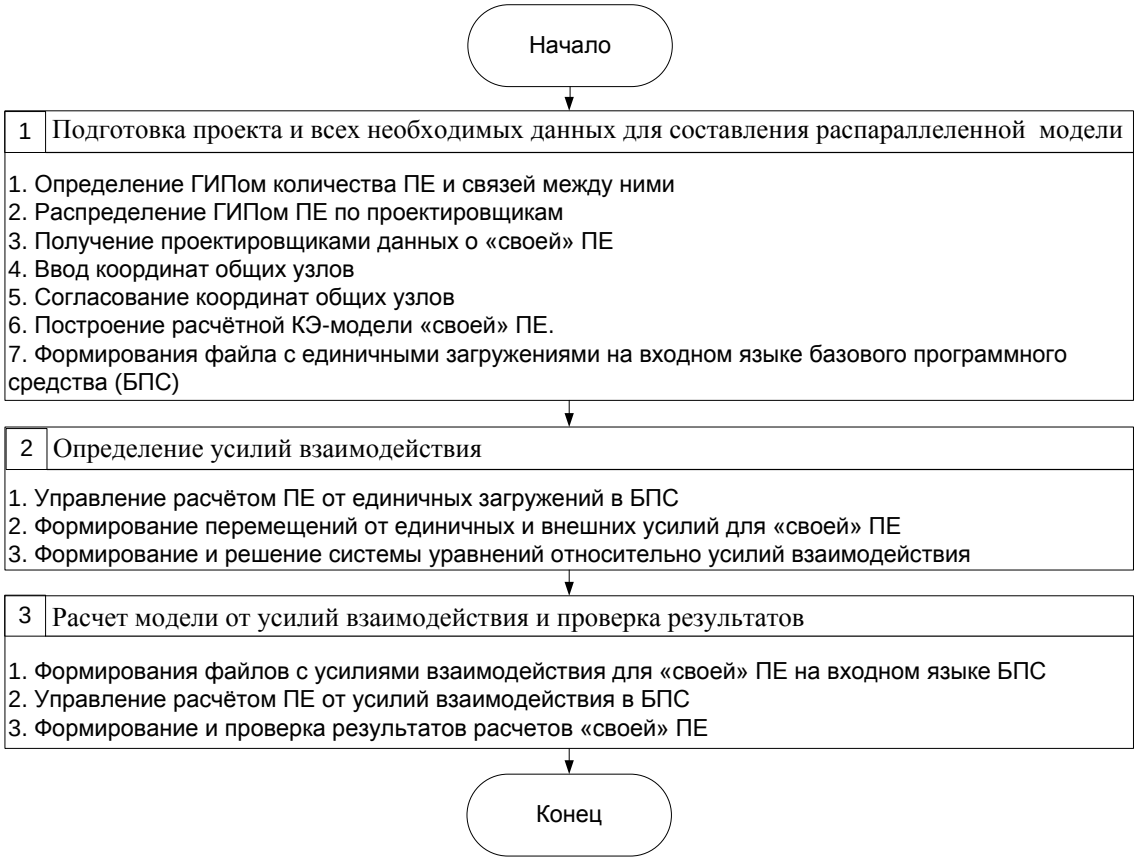



Рис. 1. Укрупненная схема работы «Решателя»

Данный XML-файл содержит шифр, номер ПЕ, а также стержни, содержащие узлы, в которых расположены координаты.

Клиенты на основе полученных данных формируют файлы модели на исходном языке «Лиры» и запускают на расчет, которым управляет система «Решатель».

По структуре файл исходных данных «Лиры» состоит из восьми блоков:

- блок 0 содержит списки единиц измерений и загружений, используемых в модели;
- блок 1 – характеристики стержневых элементов;
- блок 2 – не используется;
- блок 3 – типы и описания жесткостей;
- блок 4 – координаты узлов;
- блок 5 – характеристики связей в узлах;
- блок 6 – узловые нагрузки;

- блок 7 – типы и значения нагрузок.

Для запуска на расчет полученной модели «Лирой», «Решатель» использует конфигурационный файл запуска «Лиры», «MgParm.cfg», содержащий в себе все необходимые параметры. «Решатель» изменяет данный конфигурационный файл и запускает «Mirage.exe», за счет чего производится расчет модели. При этом в локальной папке «Лиры» создаются файлы с результатами расчетов. Затем при успешно выполненных расчетах «Mirage.exe» запускается «Bars32.exe». В локальной папке «Лиры» формируются файлы с перемещениями по данной модели.

Клиент автоматически формирует XML-файл с перемещениями в общих узлах от усилий взаимодействия для ПЕ. Данный XML-файл передается на сервер, после чего сервер обрабатывает полученный XML-файл и записывает данные в БД.

Структура данного XML-файла:

```
<UnitDisplacementFromEfforts>
  <Cipher>rama2</Cipher>
  <MePE>1</MePE>
  <Type Element="Стержень">
    <N FEM_Node="1">
      <Load Name="Загружение1">
        <RX>2.0854</RX>
        <RY>0</RY>
        <RZ>-0.2487</RZ>
        <RuX>0</RuX>
        <RuY>-1.0772</RuY>
        <RuZ>0</RuZ>
      </Load>
    </N>
  </Type>
</UnitDisplacementFromEfforts>
```

Данный XML-файл содержит шифр, номер ПЕ, стрежни, содержащие узлы, в которых расположены номер загружения и перемещения по соответствующим координатам.

Сервер производит вычисление разницы значений перемещений от усилий для парных значений, записывает эту разницу в БД, сравнивает полученные значения с допустим отклонением, составляет отчет и отправляет клиенту.

Литература

1. Suprun A.N., Kislitsyn D.I. Distributed computing for construction project design by division into project design units // Computing in Civil and Building Engineering, Proceedings of the International Conference, 30 June – 2 July – Nottingham, 2010.
2. Suprun A.N., Kislitsyn D.I. The multilevel parallelization of structural design calculation in distributed computing environment // 14th International Conference on Computing in Civil and Building Engineering (14th ICCCBE), 27 – 29 June – Moscow, 2012.

ВЕБ-СИСТЕМА РАЗРАБОТКИ И ВНЕДРЕНИЯ РАСПРЕДЕЛЕННЫХ ВИРТУАЛЬНЫХ ИСПЫТАТЕЛЬНЫХ СТЕНДОВ

Е.С. Худякова, Г.И. Радченко

Южно-Уральский госуниверситет, Челябинск

Для суперкомпьютерного моделирования сложных технологических процессов требуется организация потока задач, объединяющего несколько систем инженерного моделирования. Непосредственное решение такой задачи требует от инженера знаний по организации расчетов в распределенной вычислительной системе (РВС). Технология распределенных виртуальных испытательных стендов (РаВИС) позволяет скрыть от пользователя особенности реализации РВС, обеспечивая прозрачное моделирование инженерных задач. Предлагается веб-приложение для разработки РаВИС. С его помощью разработчик может описать параметры РаВИС, определить потоки задач в виде визуальных диаграмм, осуществить шаблонизацию файлов постановки инженерной задачи.

Введение

На сегодняшний день для решения задач, связанных с разработкой сложных технологических процессов, широкое применение нашли системы компьютерного моделирования (CAE - Computer Aided Engineering). Благодаря таким системам стало возможным проводить виртуально те эксперименты, которые нельзя или трудно осуществить реально (высокая стоимость, небезопасность и др.) [7]. Однако использование систем CAE также сопровождается рядом трудностей. Как правило, технологический цикл решения задачи включает в себя несколько этапов, которые соединены в нетривиальный нелинейный поток задач, выполняемый в рамках распределенной вычислительной системы. Процесс решения инженерной задачи можно разбить на следующие типичные этапы: формирование геометрии задачи, генерация вычислительной сетки, определение граничных условий, проведение компьютерного моделирования, визуализация, анализ результатов решения. Таким образом, для рядового инженера описание и запуск процесса инженерного моделирования с использованием одного или нескольких инженерных пакетов в распределенной вычислительной системе (РВС) может потребовать глубоких знаний в области системного программирования [5, 6].

В грид-сообществе было разработано два основных метода описания потоков задач: скриптовые языки и графы [8]. Например, в системе CLAVIRE [1] для задания потока задач используется предметно-ориентированный язык EasyFlow. Интерфейс системы представляет поле для ввода скрипта на данном языке, в другой части автоматически генерируется диаграмма потока задач. Такой подход является не очень удобным, поскольку для его использования необходимо изучение нового языка описания процессов. В системе MathCloud [2] пользователю предоставляется возможность непосредственного изображения потока задач в виде графа. И хотя граф изначально является чисто математическим понятием, он обладает очень интуитивным способом визуализации, с которым может работать каждый. Недостатком этой системы можно назвать неудачное решение представления данных в виде отдельных вершин графа, что перегружает диаграмму. Существуют и другие подходы к заданию потоков задач. В проекте Taverna [9] поток задач представляется для пользователя в трех видах (граф, xml-код,

иерархическая структура), однако редактирование потока задач возможно только в блоке представления потока задач в виде иерархической структуры. Во всех описанных системах разработчики ограничивают пользователей данными изначально сервисами, отсутствует возможность модификации и параметризации исходных файлов постановки задач, что позволило бы существенно увеличить гибкость систем.

Технология внедрения ресурсов CAE-пакетов в распределенные вычислительные среды, предлагаемая в работе [3], основывается на концепции распределенного виртуального испытательного стенда. Распределенный виртуальный испытательный стенд (РаВИС) – это программное решение, обеспечивающее проведение работ инженерного моделирования задачи в распределенной вычислительной среде с грид-инфраструктурой. РаВИС регламентирует процесс декомпозиции задачи в иерархию подзадач; поиск вычислительных ресурсов; сопоставление задачам соответствующих базовых компонентов CAE-систем; мониторинг хода решения задач; передачу результатов решения задач пользователю.

В данной статье описывается веб-приложение DiVTB Developer, обеспечивающее разработку распределенных виртуальных испытательных стендов: визуальное проектирование потока задач для проведения моделирования, шаблонизацию файлов постановки задачи, формирование набора параметров моделируемой задачи. Обеспечивается экспорт разработанного РаВИС в систему DiVTB Server для предоставления его конечным пользователям.

1. Архитектура DiVTB Developer

На рис. 1 изображена диаграмма вариантов использования системы DiVTB Developer. Пользователь системы, разработчик РаВИС, может создать РаВИС с нуля на базе имеющихся в системе сервисов, описав необходимый поток задач. Разработчик может импортировать РаВИС в систему, загрузив XML-файлы его описания. Сохранив РаВИС, пользователь может в любой момент открыть его для редактирования. Кроме того, инженер может экспортировать проект в систему DiVTB Server [4], откуда возможен запуск расчета задачи в распределенной вычислительной среде.

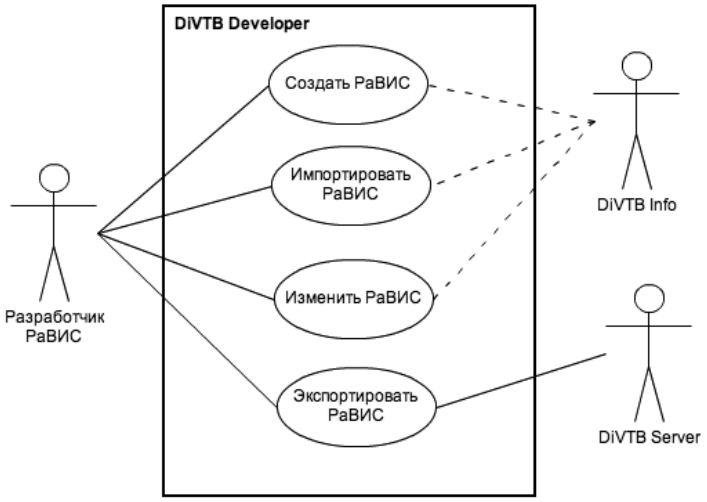


Рис. 1. Диаграмма вариантов использования системы DiVTB Developer

Для создания, импорта и изменения РаВИС необходима информация о существующих сервисах, их параметрах. Данную информацию предоставляет служебная система DiVTB Info.

Рассмотрим архитектуру системы DiVTB Developer (рис. 2). Разработчик РаВИС взаимодействует с *модулем пользовательского интерфейса*, при помощи которого со-

здает диаграмму потоков задач РаВИС. В *модуле управления проектом* поток задач обрабатывается и сохраняется в *хранилище проектов*. Файлы, необходимые для запуска расчета в конкретном инженерном пакете, загружаемые пользователем во время формирования потока задач, сохраняются в *хранилище файлов постановки задач*. *Модуль взаимодействия с DiVTB Server* позволяет инженеру загрузить проект на сервер, с которого возможен запуск проекта. Система *DiVTB Info* предоставляет системе DiVTB Developer информацию об имеющихся в РВС доступных сервисах (установленных на узлах РВС пакетах инженерного моделирования), имеющихся лицензиях на CAE-пакеты, а также о параметрах каждого сервиса.

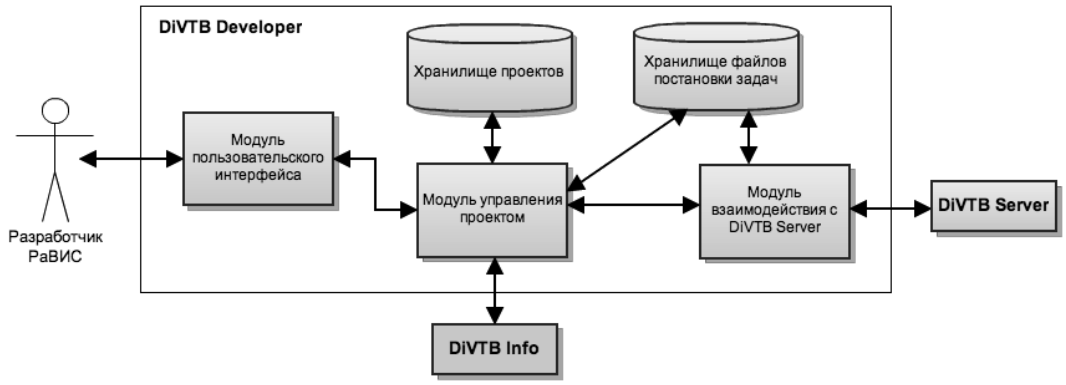


Рис. 2. Архитектура системы DiVTB Developer

Пользовательский интерфейс системы состоит из трех логических блоков (рис. 3): блок определения потока задач, блок управления процессом, блок редактирования файлов постановки задач.

Блок определения потока задач (рис. 4а) необходим для построения диаграммы потока задач – набора процессов, основанных на существующих сервисах, и дуг управления, их соединяющих. В *блоке управления процессом* (рис. 4б) происходит определение процесса, а именно:

- задание имени, словесное описание семантики процесса;
- выбор вычислительного сервиса (как системного, так и пользовательского);
- загрузка файлов постановки задачи для конкретного сервиса (CAE-пакета);
- задание значений входных параметров процесса.

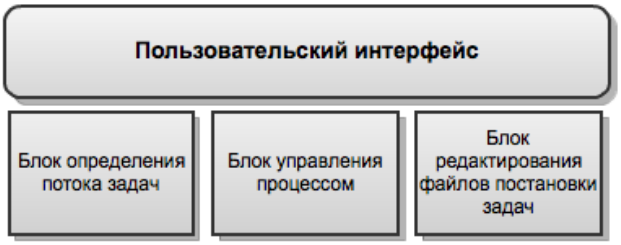
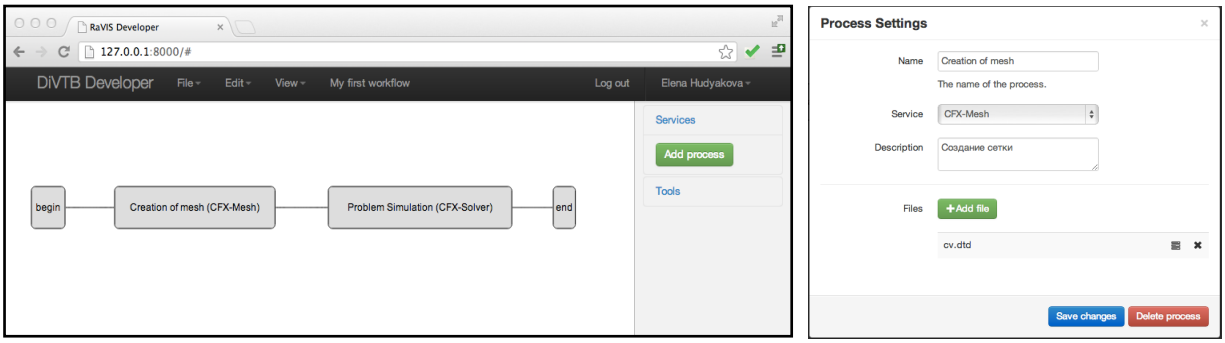


Рис. 3. Логические блоки пользовательского интерфейса системы

Под пользовательским сервисом подразумевается РаВИС, созданный инженером ранее. Он, наряду с системным сервисом (добавленным в систему администратором), может выступать в качестве блока потока задач, таким образом, реализуя идею вложенного процесса.

Блок редактирования файлов постановки задач позволяет изменять *проблемно-ориентированные файлы*. Данные файлы описывают задачу и необходимы для запуска расчета в определенном CAE-пакете. Пользователь может динамически менять файл, вставляя в текст специальный код с использованием входных параметров текущего

процесса в качестве переменных. Такой процесс назовем *шаблонизацией* проблемно-ориентированного файла. Таким образом, создав проблемно-ориентированный файл один раз для определенной задачи, пользователь может описать РаВИС, решающий целый класс задач.



(а) (б)

Рис. 4. Интерфейс системы: (а) основное окно редактирования потока задач, (б) окно редактирования процесса

2. Реализация системы

Система реализована в виде веб-приложения. Серверная часть приложения реализована на языке программирования Python с использованием фреймворка Django [10]. Данный фреймворк обеспечивает объектно-реляционное отображение данных, а его система шаблонов, разделяющая логику и представление информации, позволяет быстро и эффективно реализовывать необходимый функционал.

Клиентская часть реализована на стеке технологий HTML/CSS/JavaScript. Поскольку DiVTB Developer обеспечивает быструю работу с РаВИС, пользовательский интерфейс играет важную роль в данной системе. Поэтому при разработке интерфейса особое внимание уделялось его интуитивности и наглядности. Для этого использовался ряд библиотек. Для создания элементов пользовательского интерфейса используется JavaScript-библиотека Twitter Bootstrap [11]. Для рисования диаграммы потока задач используется JavaScript-библиотека Raphael [12].

В качестве шаблонизатора используется Java Minimal Template Engine (JMTE) [13]. JMTE – шаблонизатор, позволяющий вставлять значения переменных в текст шаблона при помощи простого синтаксиса, подобного String.format. JMTE поддерживает базовые операции с переменными (циклы, условия).

Заключение

В данной статье рассмотрена архитектура веб-приложения DiVTB Developer, обеспечивающего разработку распределенных виртуальных испытательных стендов. Разработчику предоставляется возможность визуального проектирования потока задач для проведения моделирования, шаблонизации файлов постановки задачи, формирования набора параметров моделируемой задачи. Также система предоставляет функцию экспорта РаВИС в систему DiVTB Server, откуда возможен запуск расчета задачи.

Работа выполнялась при поддержке Совета по грантам Президента РФ (проект МК-1987.2011.9), Российского фонда фундаментальных исследований (проект № 11-07-00478-а) и Министерства образования и науки РФ (государственное задание 8.3786.2011).

Литература

1. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Известия вузов. Приборостроение. 2011. Т. 54, № 10. С.36-43.
2. Лазарев И.В., Сухорослов О.В. Реализация распределенных вычислительных сценариев в среде MathCloud // Проблемы вычислений в распределенной среде. Труды ИСА РАН, Т. 46. М.: КРАСАНД, 2009. С. 6-23.
3. Радченко Г.И. Распределенные виртуальные испытательные стенды: использование систем инженерного проектирования и анализа в распределенных вычислительных средах // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". 2011. № 37(254). В. 10. С. 108-121.
4. Федянина Р.С. CAEBeans Server: среда выполнения проблемно-ориентированных оболочек над инженерными пакетами // Параллельные вычислительные технологии (ПаВТ’2010): Труды международной научной конференции (Уфа, 29 марта - 2 апреля 2010 г.). Челябинск: Издательский центр ЮУрГУ, 2010. С. 621-628.
5. Buriola T.M., Scheer S. CAD and CAE Integration Through Scientific Visualization Techniques for Illumination Design // Tsinghua Science & Technology. V. 13, N. 1. 2008. P. 26-33.
6. Fox G.C., Gannon D. Workflow in Grid Systems // Concurrency and Computation: Practice & Experience. V. 18. N. 10. 2006. P. 1009-1019.
7. Palloni F. A New Tool for Multidisciplinary CAE Simulations // NAFEMS 2005 World Congress, May 17-20, 2005, Malta.
8. Hoheisel A. User Tools and Languages for Graph-based Grid Workflows // Workflow in Grid Systems Workshop in GGF10, at Berlin, Germany, March 2004.
9. Oinn T., Addis M., Ferris J., Marvin D., Senger M., Greenwood M., Carver T., Glover K., Pocock M.R., Wipat A., Li P. Taverna: a tool for the composition and enactment of bioinformatics workflows // Bioinformatics, 2004. Vol. 20, No. 17. P. 3045-3054.
10. Фреймворк Django – [https://www.djangoproject.com/].
11. Библиотека Twitter Bootstrap – [http://twitter.github.com/bootstrap/].
12. Библиотека Raphael – [http://raphaeljs.com/].
13. Шаблонизатор Java Minimal Template Engine (JMTE) – [http://code.google.com/p/jmte/].

АНАЛИЗ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ GPU В ЗАДАЧЕ ПОИСКА В ШИРИНУ НА ГРАФЕ

М.А. Черноскутов

Институт математики и механики УрО РАН

В работе описана реализация алгоритма поиска на графе в ширину с использованием технологии CUDA на GPU, а также с использованием технологии OpenMP на CPU. Проведено сравнение быстродействия обеих реализаций с использованием графов с различным числом вершин и ребер. Показано, что GPU опережает CPU по производительности на всем диапазоне тестов.

Введение

В последнее время растет популярность использования GPU в качестве вычислителей общего назначения, с помощью которых потенциально можно запрограммировать решения самых разнообразных задач. Сейчас GPU используются в основном для приложений, которые принято называть вычислительно-интенсивными («computing intensive tasks»): механика сплошных сред, молекулярная динамика, трассировка лучей и т.д. Однако растет потребность и в задачах совершенно другой природы – «data intensive tasks», которые характеризуются работой с большими массивами данных и активным использованием нерегулярного доступа в память [1]. На данный момент уже появился специализированный рейтинг Graph500 (аналог рейтинга Top500) [2], целью которого является оценка производительности вычислительных систем при обработке больших массивов данных.

Одним из представителей задач класса «data intensive» является обход графа в ширину. Основная идея алгоритма заключается в разметке всех вершин в графе, начиная с одной заранее заданной, в определенном порядке. Более подробно о нем можно прочитать в [3]. Он используется и в качестве ядра теста Graph500 (пока реализация доступна только для CPU) [4]. Данный выбор обусловлен следующими обстоятельствами [2]:

- алгоритм обхода графа в ширину используется в различных научных и технических приложениях (анализ социальных сетей, информационная безопасность, биоинформатика, нефтедобыча);
- структура графов отображает реальные наборы данных, которые встречаются в задачах;
- результаты масштабирования задачи обхода графов должны хорошо соответствовать масштабированию реальных приложений.

В данный момент известно множество работ по обработке больших графов на CPU и GPU и сравнению их производительности [5-9].

1. Постановка задачи

Целью работы является анализ эффективности использования GPU в задаче обхода графов в ширину. Сравнение проводится с многоядерным CPU. Распараллеливание алгоритма обхода графа осуществляется по всем доступным ядрам GPU и CPU. В качестве входных данных для алгоритма используются графы одинакового размера. Результатом работы программы является время и скорость обхода заданного графа (измеряет-

ся в количестве пройденных ребер в секунду). Ранжирование результатов осуществляется по скорости обхода графа (так же, как и в рейтинге Graph500 [2]).

2. Графы, используемые в экспериментах

В каждом эксперименте использовался ориентированный граф, с заранее заданным числом исходящих из каждой вершины ребер. При этом вершины, которые является противоположными концами ребер, выбирались случайным образом (не исключая возможности образования петель и несвязных компонент). Графы с подобной структурой часто встречаются в реальных задачах, таких как анализ социальных сетей или биоинформатика. Размер графов выбирался таким образом, чтобы заполнить всю доступную DRAM память на GPU. Используемые в работе графы приведены в табл.1.

Таблица 1. Описание используемых графов

Количество исходящих из вершины ребер, шт.	Количество вершин, шт.	Размер графа, ГБ
5	66 060 288	2.21
10	48 234 496	2.52
15	35 651 584	2.52
20	28 311 552	2.53
25	23 068 672	2.49
30	19 922 944	2.52
35	16 777 216	2.44
40	14 680 064	2.41
45	13 631 488	2.49
50	12 582 912	2.53
55	11 534 336	2.54
60	10 485 760	2.50
65	9 437 184	2.43
70	8 388 608	2.31

3. Алгоритм обхода графа

Для поиска в ширину с использованием GPU реализовано два CUDA-ядра. Первое производит перебор вершин в текущей итерации (в порядке обхода в ширину) и заносит данные об их соседях в специальный массив, а второе, анализируя этот массив, формирует список вершин для следующей итерации алгоритма. Каждому GPU-потоку ставится в соответствие одна вершина. В текущей реализации в каждом блоке задействовано 1024 потока. Количество блоков варьируется в зависимости от количества исходящих ребер для каждой вершины.

Поиск в ширину на графе с помощью CPU производится аналогичным образом – используются две функции, распараллеленные на все доступные ядра CPU: первая обходит вершины на текущей итерации, а вторая формирует список вершин для следующей итерации.

4. Описание и результаты эксперимента

Эксперименты проводились на материально-технической базе суперкомпьютерного центра Института математики и механики УрО РАН. Для тестирования использовались GPU Nvidia Tesla M2050, имеющая 448 вычислительных ядер и 3 ГБ DRAM памяти (в действительности, доступным для использования оказалось около 2.5 ГБ из-за включенного ECC и других накладных расходов) и CPU Intel Xeon X5675, имеющий 6 вычислительных ядер и 48 ГБ оперативной памяти. Оба вычислителя располагаются в сервере HP ProLiant SL390s G7 4U.

Программа для GPU скомпилирована с использованием nvcc, входящего в комплект CUDA Toolkit 4.0. Для CPU использовался компилятор icc версии 11.1 и технология OpenMP. Для распраллеливания обхода графа в ширину на все процессоры в CPU и потоки в GPU использовались технологии OpenMP и CUDA. Для генерирования ребер в графах использовался генератор случайных числе rand(), входящий в стандартную библиотеку языка C.

Программам на GPU и CPU подавались графы с одинаковым числом вершин и одинаковым количеством ребер, исходящих из каждой вершины. Эксперименты проводились с графами, имеющими от 5 до 70 исходящих из каждой вершины ребер. Количество вершин в графе варьировалось от 8 388 608 до 66 060 288, в зависимости от объема доступной памяти и числа исходящих из каждой вершины ребер.

Как видно из рис.1, GPU заметно опережает CPU во всех экспериментах. Наибольшее преимущество GPU имеет при обходе графов с малым числом исходящих ребер. При увеличении числа исходящих ребер, преимущество GPU постепенно уменьшается, вследствие роста количества запросов в память, производимых множеством параллельных нитей. В CPU, наоборот, параллельных потоков намного меньше и вычислительные ядра имеют большую скорость работы по сравнению с нитями GPU.

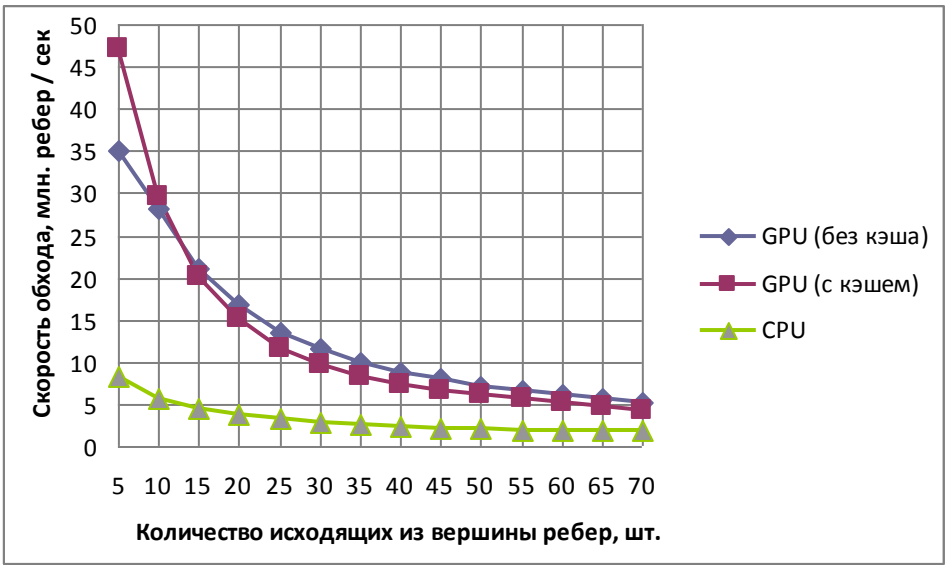


Рис. 1. Зависимость скорости обхода графов на CPU и GPU от количества исходящих ребер из каждой вершины в графе

Достигнутое на GPU ускорение показано на рис.2. Интересной особенностью является влияние кэша L1 в GPU на скорость обхода графа. При малом числе исходящих ребер кэш GPU оказывает положительное влияние на производительность, т.к. может захватить больше данных о вершинах графа, а с ростом их числа, наоборот, показывает ухудшающиеся результаты. К тому же с ростом количества исходящих ребер снижается пространственная и временная локальность [10] и степень промахов кэша из-за этого только увеличивается.

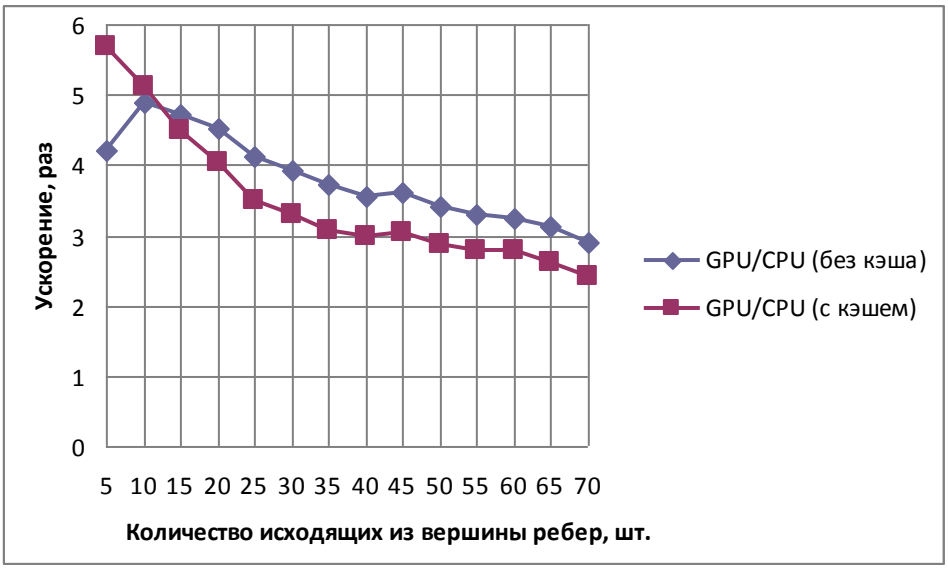


Рис.2. Ускорение, достигнутое на GPU

5. Выводы

В результате эксперимента выявлено, что поиск на графах в ширину при использовании GPU проводится с большей скоростью, по сравнению с аналогичной задачей на CPU, при условии, что размер графа не превышает объем памяти в GPU. Установлено, что кэш L1 в GPU в большинстве случаев не дает выигрыша в производительности.

В качестве дальнейших направлений исследований интерес представляет:

- обход графов, полученных с помощью генераторов (таких как Kronecker Graph Generator [11]), а также графов, полученных в результате решения реальных задач (к примеру графы из Sparse Matrix Collection [12] или DIMACS Implementation Challenge [13]);
- анализ эффективности обхода больших графов в multi-GPU системах, а также в кластерных системах, использующих GPU.

Работа поддержана грантом УрО РАН РЦП-12-П13.

Литература

1. Furht B., Escalante A. Handbook of Data Intensive Computing, Springer, 2011.
2. Murphy R., Wheeler K., Barrett B., Ang J. Introducing the Graph 500 // Cray User's Group (CUG), May 5, 2010.
3. Кормен Т., Лейзерсон Ч., Риверс Р., Штайн К. Алгоритмы. Построение и анализ – М.: «Вильямс», 2011.
4. Graph 500 Benchmark 1 (“Search”) – [http://www.graph500.org/specifications].
5. Harish, P. and Narayanan, P.J. Accelerating large graph algorithms on the GPU using CUDA // Proceedings of the 14th International Conference on High performance computing (Berlin, Heidelberg, 2007). P. 197–208.
6. Hong S. et al. Accelerating CUDA graph algorithms at maximum warp // Proceedings of the 16th ACM Symposium on Principles and practice of parallel programming (New York, NY, USA, 2011). P. 267–276.
7. Merrill D., Garland M., Grimshaw A. Scalable GPU graph traversal // Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP’12). P. 117-128.

8. Hong S., Oguntebi T., Olukotun K. Efficient Parallel Graph Exploration on Multi-Core CPU and GPU // Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques (Galveston, TX, USA, October 10-14, 2011). P. 78–88.
9. Корж А.А. Масштабирование Data Intensive приложений с помощью библиотеки Dislib на суперкомпьютерах BlueGene/P и Ломоносов // Материалы Всероссийской научной конференции «Научный сервис в сети ИНТЕРНЕТ», Новороссийск, 19-24 сентября 2011 г. Изд-во МГУ. С. 126–131.
10. Murphy R., Kogge P. On the Memory Access Patterns of Supercomputer Applications: Benchmark Selection and Its Implications // IEEE Transactions on Computers. 56(7), July 2007. P. 937–945.
11. Seshadhri C., Pinar A., Kolda T. An In-Depth Study of Stochastic Kronecker Graphs // Proceedings of the 2011 IEEE 11th International Conference on Data Mining (ICDM). P. 587–596.
12. Davis T., Hu Y. The University of Florida Sparse Matrix Collection // ACM Transactions on Mathematical Software. 2011. Vol. 38. P. 1–28.
13. 10th DIMACS Implementation Challenge – [http://www.cc.gatech.edu/dimacs10/index.shtml].

МОДЕРНИЗАЦИЯ ПАРАЛЛЕЛЬНОГО СЕТЕВОГО СИМУЛЯТОРА ПСС

П.Ю. Шамин, В.В. Прокошев

Владимирский госуниверситет им. А.Г. и Н.Г. Столетовых

Рассмотрена доработка параллельного сетевого сетевого симулятора ПСС, разработанного в ВлГУ, которая обеспечивает переход с фиксированного шага по времени, использовавшегося в исходном варианте симулятора, к переменному, обусловленному частотой наступления событий в модели. Выполненная доработка позволяет более адекватно моделировать некоторые процессы, однако снижает производительность процесса моделирования, поэтому в статье также рассматривается возможный подход к повышению производительности симулятора.

Введение

В ВлГУ на кафедре физики и прикладной математики в ходе выполнения нескольких госбюжентых начно-исследовательских работ был разработан параллельный сетевой симулятор ПСС, позволяющий производить имитационное моделирование различных процессов, связанных с взаимодействием в среде из сети узлов. Причём речь идёт не только о информационных сетях, возможно моделирование сетей любой природы, например, социальных взаимодействий в обществе. На базе ПСС также создан параллельный наносимулятор, позволяющий изучать процессы в наноструктурированной среде (содержащей углеродные нанотрубки).

Симулятор ПСС широко применяется как инструмент имитационного моделирования сотрудниками кафедры в научной работе, в частности, задействован в нескольких НИР. С использованием моделей ПСС, осуществлена защита нескольких кандидатских диссертаций. Кроме того, ПСС активно используется в учебном процессе, в частности, как инструмент для курсового проектирования по дисциплине «Компьютерное моделирование» и дипломного проектирования студентов специальностей «Прикладная математика и информатика», «Математическое обеспечение и администрирование информационных систем».

Подробнее об устройстве и принципах использования ПСС можно узнать в [1].

1. Принципы функционирования ПСС

Физически параллельный сетевой симулятор ПСС представляет собой распределённое приложение, предназначенное для работы в среде операционной системы Linux с использованием технологии MPI. При запуске симулятора создаётся один управляющий процесс (УП) и несколько (как, минимум, один) моделирующих процессов (МП). Модель сети, представленная в виде определённой иерархии объектов, разбивается на сегменты, каждый из которых поддерживается своим МП. Взаимодействие между узлами моделируемой сети в ПСС осуществляется путём передачи объектов «пакет данных». Логика модели реализуется путём вызова на каждом шаге для каждого узла моделируемой сети модулей расширения, которые пользователь пишет в соответствии со своей задачей. Модули расширения физически представляют собой динамические библиотеки Linux и могут: порождать новые пакеты, обрабатывать и перенаправлять пакеты, поступившие на узел, и собирать статистику работы. Отсчёт модельного времени

осуществляется циклами. Доставка пакетов данных в моделируемой сети осуществляется по различным алгоритмам в зависимости от того, требуется ли выполнить доставку пакета узлу, моделируемому тем же МП (локальная доставка) или другим МП (удалённая доставка). Локальная доставка производится копированием объекта «пакет данных» из буфера исходящих пакетов объекта узла-отправителя в буфер входящих пакетов узла-получателя. Удалённая доставка выполняется путём передачи MPI сообщения в адрес МП, которым поддерживается узел-получатель. Логика основного цикла работы МП ПСС приведена на рис. 1. Синхронизация модельного времени на всех МП осуществляется естественным образом процедурой удалённой доставки.

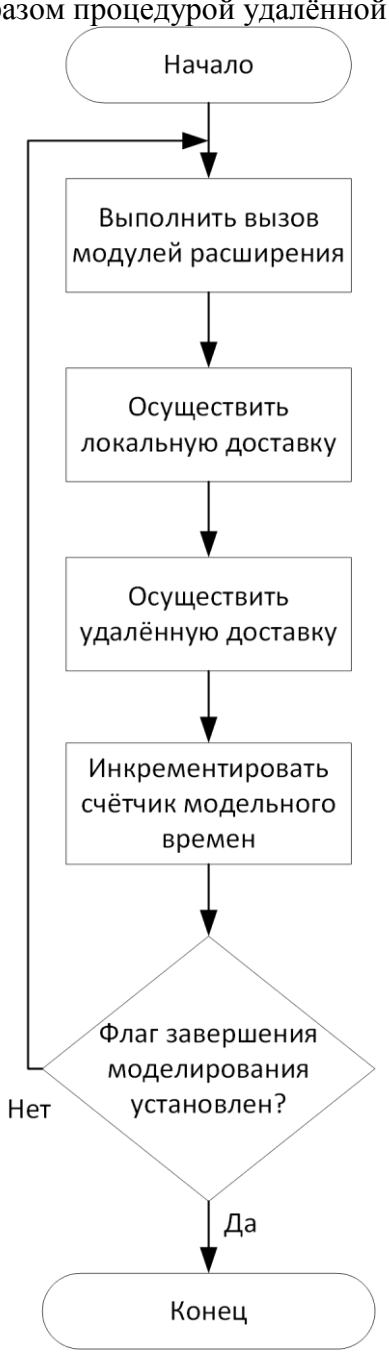


Рис. 1. Основной цикл работы МП ПСС

Следует отметить, что УП в ходе основного цикла работы ПСС используется только для ведения файла-журнала и досрочного принудительного завершения процесса моделирования. Основные свои функции он выполняет до начала моделирования (за-

грузка и передача конфигурационной информации) и после его завершения (агрегация и сохранение результатов моделирования).

2. Описание задачи

Как можно заметить из приведённого описания, изменение модельного времени в ПСС происходит с дискретным шагом фиксированного размера, равным единице. Это не всегда удобно, а иногда и вовсе неприемлемо. Кроме того, передача пакетов от узла к узлу считается происходящей всегда за один шаг, то есть пакет, переданный на текущем шаге, в начале следующего оказывается на узле-получателе.

С учётом сказанного выше, при модернизации ПСС ставились следующие задачи.

- Введение переменного шага по времени, приращение которого определяется временем наступления ближайшего события в модели.
- Реализация возможности задержки доставки отправленного пакета узлу-получателю до указанного момента времени.

Кроме того, поскольку модельное время в рамках всей модели должно быть синхронизировано, возникает задача о выборе «следующего» момента модельного времени, единого для всех МП. Данная процедура наиболее естественно организуется с использованием УП в качестве своеобразного «арбитра».

3. Описание решения

С целью решения данной задачи, логика работы ПСС была существенно переработана. Так, например, в исходной версии каждый узел моделируемой сети имел два буфера – для входящих и исходящих пакетов. Для реализации возможности доставки пакетов за произвольное время потребовалось дополнительно оснастить каждый узел буфером пакетов, находящихся в процессе доставки данному узлу, но ещё не поступивших на него. Для определения момента фактического поступления, каждый объект пакета был дополнен полем метки времени, в которое он должен достигнуть адресата. Типичная цепочка перемещений пакета приведена на рис. 2.

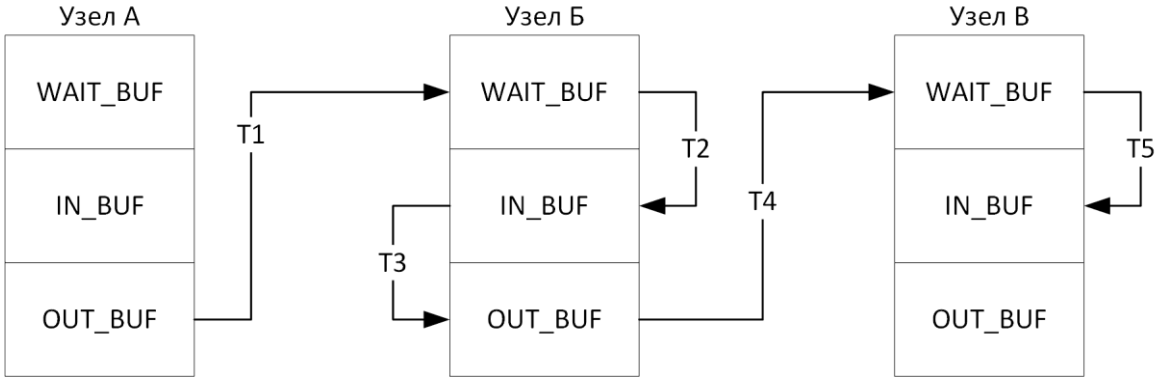


Рис. 2. Типовой порядок перемещения пакета в модернизированной версии ПСС

На рис. 2 представлены три узла – А, Б, В (в реальности цепочка может иметь произвольную длину, пока не истечёт время жизни пакета – TTL). Процесс обработки пакета происходит следующим образом:

- Пакет порождается в некотороый момент времени T1 одним из модулей расширения ПСС и помещается в буфер исходящих пакетов (OUT_BUF) узла-отправителя А.
- Ядро ПСС перемещает пакет в буфер транспортируемых пакетов (WAIT_BUF) узла-получателя Б с прибавлением некоторого времени к метке времени следующей обработки пакета, что позволяет смоделировать задержку при передаче (новая метка времени пакета – T2).

- При наступлении момента времени T2 пакет помещается в буфер входящих пакетов узла Б. Предположим, что узел Б должен ретранслировать пакет. Тогда один из модулей расширения ПСС, отвечающий за маршрутизацию, устанавливает для пакета метку времени T3, имитируя затрату времени на его обработку, и помещает его в буфер исходящих пакетов узла Б.
- При наступлении момента времени T3 ядро ПСС перемещает пакет из буфера исходящих пакетов узла Б в буфер ожидания узла В. Пакет вновь получает увеличенную метку времени (новое значение T4), чтобы имитировать затрату времени на транспортировку.
- Наконец в момент времени T4 пакет попадает в буфер входящих пакетов узла В, где один из модулей расширения ПСС обнаруживает, что узел В является окончательным узлом-получателем, обрабатывает содержимое пакета и уничтожает его. Время уничтожения фиксируется как T5, с учётом имитируемых затрат времени на обработку пакета.

Другой проблемой явился единый отсчёт модельного времени. Данная задача была решена следующим образом:

- Выполнив все операции по моделированию для текущего момента модельного времени, каждый МП определяет следующий момент модельного времени, в который у него должно произойти следующее событие (событием в модели является наступление времени обработки какого-либо пакета).
- Каждый МП пересылает свой «локальный» ближайший момент времени УП.
- Получив «локальные» ближайшие моменты времени от всех МП, УП выбирает из них «глобальный» ближайший момент в пределах всей модели и рассылает его каждому МП.
- МП осуществляют обработку всех пакетов, которые должны быть обработаны, в новый текущий момент времени (в случае если таковые есть в сегменте сети, моделируемом данным МП).
- Цикл повторяется до установки флага завершения моделирования, принудительного прерывания процесса моделирования или наступления некоторого (определённого параметрами настройки ПСС) момента модельного времени.

С учётом описанной логики изменения модельного времени, основной цикл работы МП ПСС приобрёл вид, приведённый на рис. 3.

4. Дальнейшие работы по проекту

Выполненная доработка позволила существенно повысить функциональность ПСС за счёт более гибкого управления модельным временем и возможности учёта различных расходов времени на обработку и транспортировку пакетов. Однако при неудачном стечении обстоятельств возможно существенное снижение производительности (много моментов времени, когда один МП работает, остальные простаивают). Возможным решением проблемы было бы внедрение оптимистического алгоритма управления модельным временем вместо консервативного. В этом случае каждый МП обрабатывал бы пакеты для своего ближайшего момента времени, а при возникновении коллизий (имеются пакеты, предназначенные данному МП, но с меткой времени в прошлом), осуществлялся бы откат изменений до данной метки времени и повторный расчёт. Поскольку преимущество в производительности зависит от частоты «откатов», а реализация подхода требует значительного расхода оперативной памяти, данный подход требует дополнительной проработки и экспериментальной оценки его эффективности.

Рис. 3. Основной цикл работы МП ПСС после модернизации

Работа выполнена в рамках НИР по госзаданию «Наука» (регистрационный номер: 8.3534.2011 от 23.11.2011).

Литература

1. Шамин, П.Ю. Параллельный сетевой симулятор: концепция и перспективы развития / П.Ю. Шамин, А.С. Алексанян, В.В. Прокошев // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. – СПб., 2009. – № 3. – С. 18–24.

ПОДСИСТЕМЫ УПРАВЛЕНИЯ И КОНТРОЛЯ РАБОТОСПОСОБНОСТИ РЕШАЕМОЙ ЗАДАЧИ И АППАРАТУРЫ ГЕТЕРОГЕННОГО ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

Е.А. Шамов

Волгоградский государственный технический университет

Рассматриваются методы и подходы по организации возможности управления и контроля работоспособности высокопроизводительных вычислительных комплексов. Показано, что использование командной логики и организация контроля работоспособности аппаратуры и задач дает возможность динамически управлять с необходимым функционалом не только кластером в целом, но и каждой конкретной задачей в частности. Кроме того, такой подход позволяет организовывать надежные вычисления, как на уровне потока задач, так и на уровне отдельно решаемой задачи. В результате, применение этого подхода апробировано и показало эффективность при работе систем моделирования динамики электронных потоков и динамики жидкости.

Введение

В современном мире существует большое количество задач, требующих значительных вычислительных затрат. Задачи данного класса не представляется возможным решать при помощи обычных ЭВМ с последовательными методами, алгоритмами и программами за приемлемое время, а иногда и вовсе нет возможности их решить по причине нехватки ресурсов. Единственная возможность решать такие задачи – применять специальные аппаратно-программные комплексы (кластеры, гриды, облачные системы). К сожалению, при создании, программировании, управлении и, в общем, работе таких комплексов существует ряд трудностей и проблем, которые необходимо эффективно решать.

Одни из ключевых направлений развития высокопроизводительных вычислительных комплексов – управление, надежность и отказоустойчивость. Причем стоит особо отметить, что развитие подсистем управления и организация надежных вычислений относится не только непосредственно к комплексу, но и к решаемым на нем задачам. По сути, современная ситуация обстоит таким образом, что нет возможности управлять комплексом в полной мере или в той степени и теми функциями, которые необходимы. Кроме того, насколько бы ни был комплекс высоконадежным, у него все равно может отказывать оборудование, что, как правило, приводит к завершению корректного выполнения задачи или даже к «зависанию» выполнения задачи, которое нет возможности или сложно обнаружить извне задачи.

Управление комплексом

Современные высокопроизводительные вычислительные комплексы, как правило, имеют подсистемы управления с ограниченным функционалом. В основном такие подсистемы позволяют лишь управлять очередью задач и отображают текущее состояние вычислительных узлов (например, MS HPC Cluster Manager). Такие подсистемы базируются на жесткой логике и обладают соответствующими достоинствами и недостатками.

Стоит отметить, что большинство программ, разрабатываемых для высокопроизводительных комплексов, также построены на жесткой логике и, как правило, не облада-

ют внешними функциями управления (нет возможности контролировать извне). Однако такие программы удобны тем, что работают по принципу «запустил и забыл», но для ряда случаев это неприемлемо. Простейшим примером являются задачи, работающие на пета- и экзафлопсных (в перспективе) комплексах. При работе таких комплексов отказы оборудования – достаточно частое явление, а значит, каждая программа должна корректно и эффективно работать при соответствующих отказах, что очень сложно предусмотреть при использовании жесткой логики. Еще одним примером служит класс задач, требующих организации эффективной динамической нагрузки. Динамическая нагрузка при использовании жесткой логики требует значительных усилий при проектировании программы, и к тому же практически нет возможности реализовывать нагрузку эффективно при смене аппаратной платформы и изменении вычислительной сложности задачи или подзадач и т.д. По сути, жесткая логика и называется жесткой, так как не обладает свойством приспосабливания под какие-либо изменения.

Для организации универсальной системы управления предлагается использовать командную логику – подход, при котором вычислительный комплекс и каждая отдельно решаемая задача выполняют соответствующий поток инструкций. По причине того, что поток можно динамически менять, появляется возможность автоматически перестраивать ход работы комплекса и/или задач для их корректного функционирования и повышения эффективности.

Командная логика позволяет:

- управлять любыми аспектами работы комплекса, а именно его очередями, аппаратурой, индивидуально каждой задачей (если требуется) и т.д.;
- полностью управлять ходом выполнения каждой задачи и ее подзадачами.

В итоге командная логика предоставляет возможность полностью управлять всем комплексом, как единым целым, и каждой, отдельно взятой, его частью.

К сожалению, за использование командной логики приходится зачастую платить производительностью, но потери можно свести к минимуму при помощи специальных подходов считывания и обработки команд. В целом основной проблемой применения командной логики является сложность проектирования и кодирования, которые и без нее сложно реализовать по причине использования огромного количества различных методов, технологий и подходов.

Высокая надежность комплекса

Организация надежных вычислений на высокопроизводительном вычислительном комплексе – трудная и ресурсоемкая задача. К тому же эффективность современных систем повышения надежности оставляет желать лучшего.

Проблема надежности приводит к тому, что необходимо контролировать не только аппаратуру комплекса, но и ход выполнения задачи и отдельно взятых ее частей. Для корректного функционирования комплекса необходимо не только вовремя находить отказы и сбои оборудования, но и динамически модифицировать логику работы задач и комплекса.

В итоге для высоконадежных вычислений необходимо динамически отлавливать отказы и сбои, а соответственно лучше всего использовать командную логику, которая позволяет проводить диагностику комплекса, «на лету» перераспределять подзадачи, предназначенные для расчета на вышедшем из строя оборудовании, скорректировать задачи, находящиеся в очереди и т.д. Без организации такого контроля зачастую решение задачи попадает в тупиковую ситуацию, что сложно обнаружить и, как результат, будут впустую потрачены ресурсы.

Простейшим способом повышения надежности является организация опроса подсистем и оборудования комплекса, а также каждого узла (потока) решаемой задачи.

При грамотной реализации такой способ позволяет обнаруживать отказы и, автоматически направлять ход решения задач в нужное русло.

Апробация

Проверка практической применимости и эффективности подсистем управления и контроля работоспособности решаемой задачи и аппаратуры комплекса была частями реализована в системах моделирования динамики жидкости и электронных потоков. Отказы систем эмитировались при помощи специальных функций прекращения корректной работы потока узла, а также физическим отключением линий сети. Подсистемы управления и контроля работоспособности экспериментально подтвердили свою эффективность, так как обнаруживали все отказы и эффективно перераспределяли нагрузку между оставшимися ресурсами, а в итоге задачи корректно завершали свое функционирование и выдавали правильные результаты.

Далее коротко рассмотрим решение задачи падения куба воды на ограниченную наклонную поверхность при помощи созданной системы моделирования динамики жидкости. Данная система базируется на межмолекулярном взаимодействии на основе потенциала Леннарда-Джонса. При моделировании динамики жидкостей на молекулярном уровне важны такие свойства реальных жидкостей, как вязкость и сжимаемость. На рис. 1 в области 1 выделены центры масс, не подвергшиеся явлениям сжимаемости и вязкости, а в области 2 выделены центры масс, подвергающие или подвергшиеся явлению вязкости.

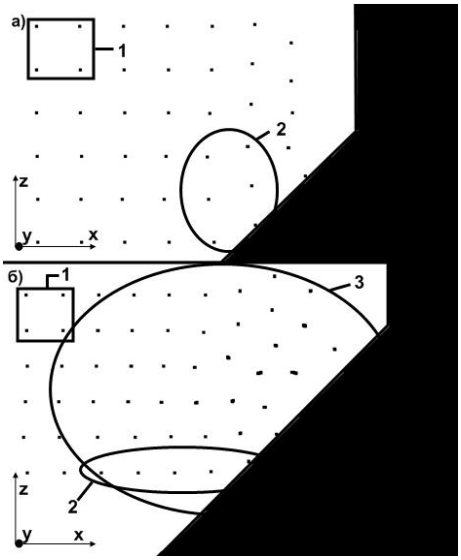


Рис. 1. Явления сжимаемости и вязкости при ударе куба воды об ограниченную наклонную поверхность

В области 3 на рис. 1 выделены комплексные явления сжимаемости и вязкости. Изучение рисунка 1 дает возможность наглядно убедиться в том, что центры масс отдельных частей куба воды смещаются, а соответственно проявляются явления вязкости и сжимаемости жидкости, что, в свою очередь, подтверждает адекватность метода и модели.

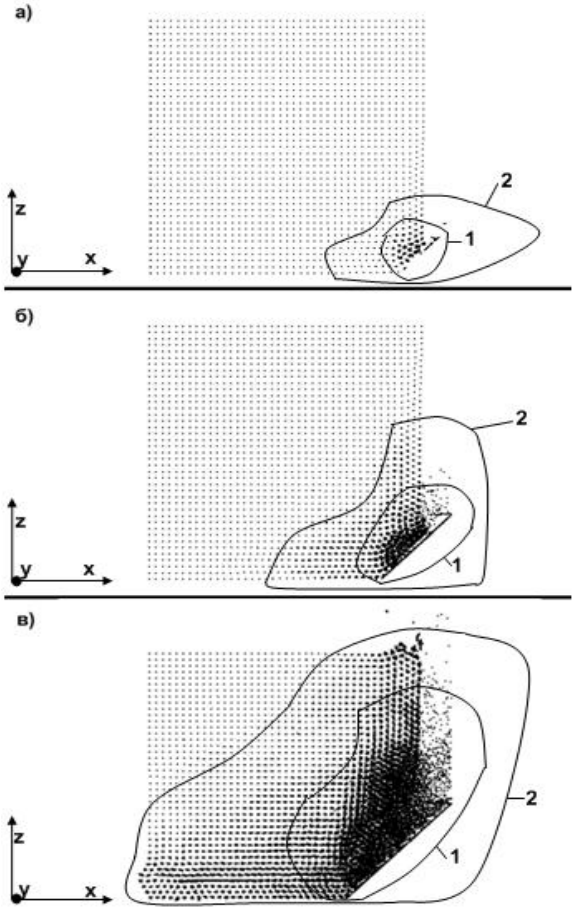


Рис. 2. Явление зарождения и динамики волн внутри куба воды

На рис. 2 изображены картины зарождения и динамика волн внутри куба воды при столкновении с ограничительной поверхностью. Области 2 на рис. 2 показывают передовой край волны, а области 1 показывают край волны, приводящий к сильным «деформациям».

Рис. 2а, 2б, 2в отличаются лишь временем моделирования – 0.4, 0.41 и 0,45 секунды соответственно. Как видно из рис. 2в, волны давления жидкости приводят даже к отделению нескольких капель жидкости от основной массы не по причине соударения, а по причине воздействия высокоэнергетической волны. Рисунки 1 и 2 подтверждают адекватность используемых методов, применяемых для моделирования задач гидродинамики.

Рассмотрим теперь эффективность применения технологии параллельного программирования на кластере центральных и графических процессорных устройств (ЦПУ и ГПУ). В рамках исследования на базе Волгоградского государственного технического университета (ВолгГТУ), кафедры ЭВМ и систем, был собран кластер ЦПУ и ГПУ с пиковой мощностью в 8 TFlops.

Таблица 1. Эффективность распараллеливания задач гидродинамики

Число контролируемых центров масс, штук	**Число итераций, штук**	**Аппроксимированное время расчета на одном ядре ЦПУ, с (дни)**	**Время расчета на кластере ЦПУ и ГПУ, с (дни)**	**Ускорение, разы**
1331	10000	37891 (0.43855)	137 (0.0015856)	276
10648	10000	2847951 (32.96239)	5456 (0.06315)	522
68921	10000	45224657 (523.4336)	72943(0.8443)	620

Из табл. 1 можно сделать заключение о том, что распараллеливание задачи падения куба воды на ограниченную наклонную поверхность при помощи технологий MPI, OpenMP и Nvidia CUDA является эффективным, а соответственно, методы, примененные для задач гидродинамики, также эффективны.

Заключение

Моделирование задач динамики жидкостей и электронных потоков на кластере ЦПУ и ГПУ позволяет на несколько порядков сократить время расчета, получать более точные данные и проводить качественный анализ. Благодаря использованию параллельного программирования появилась возможность анализа сложных электромагнитных и гидродинамических систем и устройств, что крайне затруднительно при его отсутствии. В результате, использование командной логики и организация контроля работоспособности аппаратуры и задач дало возможность динамически управлять не только кластером в целом, но и каждой конкретной задачей в частности. Кроме того, такой подход позволил организовывать надежные вычисления, как на уровне потока задач, так и на уровне отдельно решаемой задачи.

Литература

1. Шеин А.Г., Шамов Е.А. Стохастическая модель динамики плоского электронного потока в скрещенных статических электрическом и магнитном полях // Известия ВолгГТУ. Сер. Электроника, измерительная техника, радиотехника и связь. Вып. 3. 2009. – С. 48–53.
2. Шамов Е.А. Проблемы, возникающие при моделировании динамики произвольных объектов на кластере центральных и графических процессорных устройств. // Барышникова С.С., Жариков Д.Н., Попов Д.С. – Пермь: Издательство ПГТУ, 2010.
3. Шамов Е.А., Лукьянов В.С., Жариков Д.Н., Попов Д.С. Технологии достижения параллелизма mpi, cuda, openmp и моделирование динамики электронного потока в скрещенных полях с применением гибрида технологий mpi и openmp. // Известия ВолгГТУ. Сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. Вып. 8. 2010.
4. Шамов Е.А. Применение параллельного программирования при моделировании динамики плоского электронного потока в скрещенных статических электрическом и магнитном полях // Известия ВолгГТУ. Сер. Электроника, измерительная техника, радиотехника и связь. 2011.
5. Шамов Е.А., Шеин А.Г., Андреев А.Е., Барышникова С.С. О применении системы моделирования динамики произвольных объектов для решения задач гидродинамики на примере моделирования падения куба воды на ограниченную наклонную поверхность. // Известия ВолгГТУ. Сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. Вып. 11. 2011. С. 37–41.
6. Шамов Е.А. Моделирование динамики электронного потока в скрещенных полях на кластере центральных и графических процессорных устройств // Шестнадцатая Всероссийская научная конференция ВНКСФ-16. Волгоград – 2010.
7. Shamov E.A., Shein A.G. Introduction to the method of space miniaturization / First International Conference "Innovative Information Technologies" – Czech Republic, Prague – 2012.

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ПРИ РАСЧЕТЕ ОПТИЧЕСКИХ СВОЙСТВ ПОЛУПРОВОДНИКОВЫХ НАНОКРИСТАЛЛОВ В РАМКАХ ТЕОРИИ ФУНКЦИОНАЛА ПЛОТНОСТИ

А.В. Швецов, Д.В. Осокин

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача расчета оптических свойств многоэлектронных систем на основе динамической версии метода функционала плотности. Предлагается параллельный алгоритм расчета для систем с графическими ускорителями, который может быть использован для определения спектров как изолированных, так и взаимодействующих кремниевых нанокристаллов (квантовых точек). Приведены результаты вычислительных экспериментов.

Введение

Расчет оптических свойств многоэлектронных атомных и молекулярных систем представляет огромный интерес для спектроскопии, лазерной физики и оптоэлектронных приложений. В современной оптоэлектронике актуальной проблемой является исследование воздействия лазерного поля на отдельные кремниевые нанокристаллы (квантовые точки) и ансамбли взаимодействующих нанокристаллов, в которых состояния оккупируют большое число электронов. Интерес также представляют сложные гибридные структуры, в которых проявляется резонансная связь дискретных состояний экситонов в полупроводниковых квантовых точках с плазмонами в металлических наночастицах, приводящая к усилению фотолюминесценции полупроводниковых квантовых точек.

Для описания возбуждений в наноструктурах необходимо решить уравнение Шредингера для многоэлектронной волновой функции в многомерном пространстве и времени. В случае систем, состоящих из десятков и более частиц, подход, основанный на прямом решении уравнения Шредингера для многочастичной волновой функции, является крайне затратным. Упрощение возможно, если ограничиться кругом явлений, где проявляются свойства наблюдаемых, определяемых только электронной плотностью. В основе теории Кона-Шэма (теории функционала плотности) лежит тот факт, что наблюдаемые многоэлектронные системы могут быть выражены через электронную плотность и для этого не требуется находить многоэлектронную волновую функцию. Теория Кона-Шэма расширена Рунге и Гроссом на случай нестационарных систем и позволяет рассчитывать динамический отклик многоэлектронных систем.

Расчеты даже в рамках теории функционала плотности в случае большого числа частиц требуют большого объема вычислений. Одним из выходов является разработка параллельной реализации данного метода на графических процессорах.

1. Моделирование спектров электронных возбуждений в нанокристаллах

Ограничим рассмотрение взаимодействием сильного лазерного поля с электронной подсистемой в кремниевой квантовой точке; расчет отклика проведем в рамках нестационарной теории функционала плотности.

Отправной точкой теории функционала плотности является уравнение Шредингера для многоэлектронной функции $\psi(t)$ в присутствии нестационарного поля. Гамильтониан системы $H=T+V_{ext}+W_{ee}$ представляет собой сумму кинетической энергии электронов, потенциала взаимодействия электронов с ядрами решетки и внешними полями, а также энергии электрон-электронного взаимодействия. Одноэлектронный гамильтониан в квантовой точке берется в приближении эффективной массы. С учетом сказанного, гамильтониан многоэлектронной системы записывается в виде

$$\hat{H}(t) = -\frac{\hbar^2}{2m_e^*}\sum_{i=1}^{N_P}\nabla_i^2 + \sum_{i=1}^{N_P}\upsilon(\vec{r}_i,t) + \hat{W}_{ee},$$

где m_e^* – эффективная масса электронов, которая для конкретных материалов известна из экспериментальных работ.

Далее выбрана атомная система единиц: $e=\hbar=m_0=1$, где m_0 – масса свободного электрона. В приведенном выше выражении фигурирует внешний потенциал $\upsilon(\vec{r},t)$, который согласно теореме Рунге–Гросса однозначно связан с электронной плотностью $n(\vec{r},t)$ для многоэлектронной системы, эволюционирующей из начального состояния $\psi(t=0)$. Следовательно, если известна лишь электронная плотность системы, эволюционирующей из заданного начального состояния $\psi(t=0)$, то она однозначно определяет внешний потенциал, который ее создает. Внешний потенциал полностью определяет гамильтониан системы, тогда нестационарное уравнение Шредингера может быть решено, и получены наблюдаемые характеристики системы. Для этого выбирается система невзаимодействующих электронов, называемая KS системой, определенная таким образом, что точно воспроизводит электронную плотность истинной взаимодействующей системы. Тогда свойства истинной системы могут быть выражены через электронную плотность невзаимодействующей системы. Поскольку однозначное соответствие между зависящей от времени электронной плотностью и внешним потенциалом установлено для любого $\hat{W}_{ee}$, то оно справедливо и для $\hat{W}_{ee} \equiv 0$, то есть и для KS системы. Следовательно, внешний потенциал $\upsilon_{KS}(\vec{r},t) \equiv \upsilon_{KS}[n;\Phi_0](\vec{r},t)$ невзаимодействующей системы, воспроизводящий электронную плотность $n(\vec{r},t)$, эволюционирующую из состояния Φ_0, однозначно определен. KS состояние Φ_0 невзаимодействующих электронов выбирается в виде детерминанта одночастичных орбиталей $\varphi(\vec{r},0)$. При этом KS состояние должно соответствовать заданной начальной электронной плотности, воспроизводить ее саму и ее первую производную по времени. Электронная плотность взаимодействующей системы может быть определена как

$$n(\vec{r},t) = \sum_{j=1}^{N_p}\left|\varphi_j(\vec{r},t)\right|^2,$$

где $\varphi_j(\vec{r},t)$ являются решениями уравнения Шредингера с потенциалом $\upsilon_{KS}(\vec{r},t)$:

$$i\hbar\frac{\partial}{\partial t}\varphi_j(\vec{r},t) = \left[-\frac{\nabla^2}{2m_e^*} + \upsilon_{KS}[n;\Phi_0](\vec{r},t)\right]\varphi_j(\vec{r},t)$$

$$\upsilon_{KS}[n;\Phi_0](\vec{r},t) = \upsilon[n;\psi_0](\vec{r},t) + \int d^3\vec{r}'\frac{e^2 n(\vec{r}',t)}{|\vec{r}-\vec{r}'|} + \upsilon_{XC}[n;\psi_0,\Phi_0](\vec{r},t).$$

Здесь $\upsilon(\vec{r},t)$ – потенциал внешнего поля, второе слагаемое представляет собой потенциал Хартри $\upsilon_H(\vec{r},t)$, третье слагаемое – обменно-корреляционный потенциал. Функ-

циональная зависимость $\upsilon(\vec{r},t)$ определяется тем фактом, что внешний потенциал $\upsilon(\vec{r},t)$ является заданной функцией (складывается из потенциала, связанного с барьером на границе нанокристалла со средой, в которой он находится, и потенциальной энергии во внешнем электромагнитном поле волны). Для аппроксимации $\upsilon_{XC}(\vec{r},t)$ используется адиабатическое приближение локальной плотности, которое хорошо зарекомендовало себя при расчетах спектров многоэлектронных систем. В этом случае υ_{XC} зависит лишь от значения электронной плотности в данной точке в данный момент времени. Таким образом, задача сводится к решению N_p нестационарных уравнений Шредингера с потенциалом $\upsilon_{KS}(\vec{r},t)$ для орбиталей $\varphi_j(\vec{r},t)$.

2. Численный метод решения

Для демонстрации общей методики исследуем динамику электронных состояний в сферической потенциальной яме с бесконечными стенками. Данная модельная система должна отражать свойства сферической квантовой точки с достаточно большим потенциальным барьером (в реальных системах барьер составляет несколько электронвольт).

На первом шаге происходит расчет электронной плотности, после чего решением уравнения Пуассона определяется потенциал Хартри для данной $n(\vec{r},t)$

$$\Delta\upsilon_H(\vec{r},t) = -4\pi\left(n(\vec{r},t) - n_+(\vec{r})\right).$$

Здесь $n_+(\vec{r})$ – плотность положительных зарядов, которая выбирается в виде равномерно размазанного по всему объему нанокристалла положительно заряженного желе.

После этого определяется обменно-корреляционный потенциал по аппроксимирующей формуле

$$\upsilon_{XC}(\vec{r},t) = -\frac{1{,}222}{r_S(n)} - 0{,}066\ln\left(1+\frac{11{,}4}{r_S(n)}\right),\quad r_S(n) = \sqrt[3]{\frac{3}{4\pi n(\vec{r},t)}}.$$

Далее решаются нестационарные уравнения Шредингера для N_p орбиталей (для KS системы) на одном шаге по времени. В результате находятся орбитали KS системы на новом шаге по времени, по которым рассчитывается электронная плотность на новом шаге по времени и происходит переход к расчету $n(\vec{r},t)$ на следующем шаге по времени.

В настоящей работе для решения самосогласованных уравнений Кона-Шэма выбраны итерационный метод сопряженных градиентов и метод Рунге-Кутта, поскольку допускают блочное разбиение пространства, и, следовательно, будут хорошо сочетаться в параллельной реализации метода Кона-Шэма для расчета многоэлектронных задач.

3. Результаты вычислительных экспериментов

Предложенный алгоритм расчета оптических свойств полупроводниковых нанокристаллов в рамках теории функционала плотности на основе блочного разбиения пространства был реализован для гетерогенных систем с распределенной памятью (вычислительных кластеров с графическими процессорами на узлах).

Вычислительные эксперименты проводились на следующей платформе:

- язык программирования: C;
- используемые технологии: OpenMP, CUDA, MPI;
- операционная система: Microsoft Windows Server 2008 HPC Edition SP2 x64;
- среда разработки: Microsoft Visual Studio 2010;
- компилятор: Intel C++ Compiler XE 12.1;

- 16 узлов: 2 CPU Intel Xeon L5630 2.13 ГГц, 4 ядра; 24 ГБ оперативной памяти; NVidia Tesla M2050.

Результаты проведенных экспериментов (рис. 1) демонстрируют эффективность применения графических процессоров для решения задачи. Видно, что GPU-реализация быстрее однопоточной CPU-версии более, чем в 30 раз при размерах трехмерной пространственной сетки от 60 узлов в каждом направлении. Рисунок 2 показывает зависимость ускорения параллельной GPU-реализации метода Кона-Шэма для систем с распределенной памятью в зависимости от числа используемых графических процессоров.

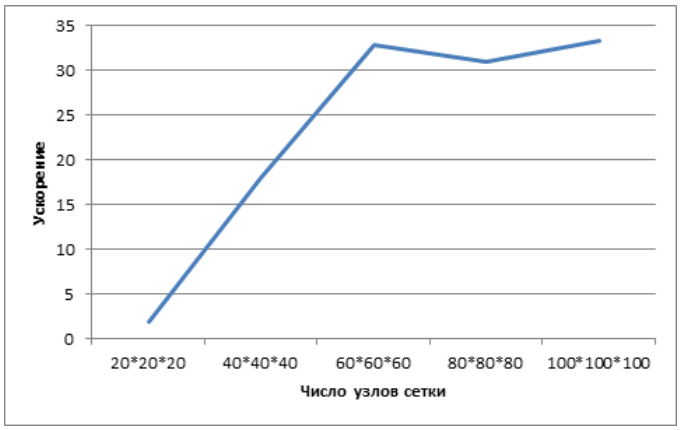


Рис. 1. Ускорение GPU-реализации метода Рунге-Кутта относительно многопоточной CPU-реализации

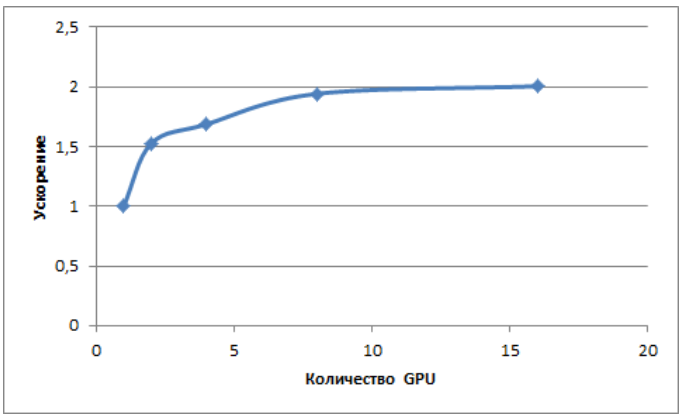


Рис. 2. Масштабируемость параллельной GPU-реализации метода Кона-Шэма

На рис. 3 приведены результаты следующего набора экспериментов: рассматривается параллельная GPU-версия для систем с распределенной памятью, и производится ее запуск на кластере с графическими ускорителями на каждом из узлов. Осуществляются параллельные запуски алгоритма на разном числе узлов кластера, причем количество процессов (узлов) соответствует размеру пространственной сетки, используемой для расчетов. Эффективность распараллеливания при использовании 68 GPU здесь составляет примерно 31.4%.

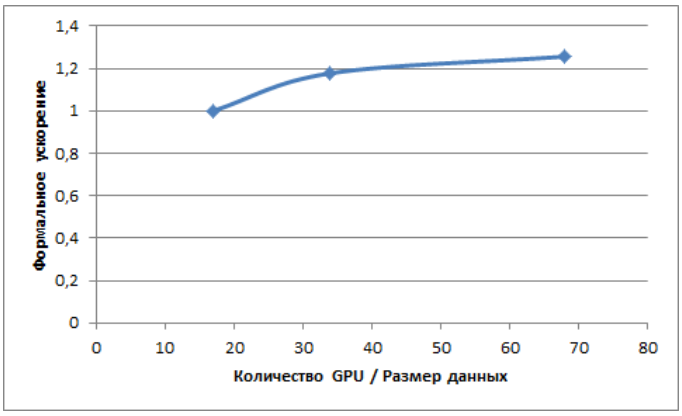


Рис. 3. Масштабируемость параллельной GPU реализации метода Кона-Шэма при пропорционально увеличивающемся объеме данных

Заключение

Предлагаемый алгоритм решения динамических уравнений Кона-Шэма на основе блочного разбиения пространства может давать существенное ускорение при реализации на гетерогенных системах с распределенной памятью. Использование дополнительных графических процессоров в данной задаче наиболее полезно при расчетах с пропорционально увеличивающимся объемом данных. Эффективность распараллеливания продемонстрирована на вычислительных системах при использовании 68 GPU и составляет приблизительно 31.4%.

Реализация была использована при разработке методики расчета оптических свойств многоэлектронных систем – полупроводниковых квантовых точек. Также, алгоритм может быть использован для определения спектров как изолированных, так и взаимодействующих кремниевых нанокристаллов (квантовых точек) различной формы.

Литература

1. Shimizu K.T., Woo W.K., Fisher B.R., Eisler H.J., and Bawendi M.G. // Phys. Rev. Lett. 2002. V. 89. P. 117401.
2. Runge E., Gross E.K.U. // Phys. Rev. Lett. 1984. V. 52. P. 997-1000.
3. Ferconi M., Vignale G. // Phys. Rev. B. 1994. V. 50. P. 14722.
4. Лундквист С., Марч Н. Теория неоднородного электронного газа. М.: Мир, 1987.
5. Broin C.O., Nicolopoulos L.A.A. An OpenCL implementation for the solution of TDSE on GPU and CPU architectures // Computer Physics Communications 183, 2012. P. 2071-2080.
6. Маркус Е.Д. Исследование параллельной реализации метода расщепления для уравнения теплопроводности на кластерных вычислительных системах // Вестник Томского государственного университета 1(14), 2011. С. 73–78.
7. Игнатьев А.А., Затевахин М.А. Параллельный метод для решения уравнения Пуассона // Параллельные вычислительные технологии: Доклад Международной научной конференции, Нижний Новгород, Россия, 30 марта – 3 апреля 2009. С. 491–495.

СУПЕРКОМПЬЮТЕРЫ И ВОПРОСЫ ФУНДАМЕНТАЛЬНЫХ ОСНОВ ИНЖЕНЕРНОГО ОБРАЗОВАНИЯ

Ю.Я. Болдырев, К.Ю. Замотин, Е.П. Петухов

Санкт-Петербургский государственный политехнический университет

Введение

Сегодня в России повсеместно наблюдается парадоксальная и тревожная для развития высокотехнологичных секторов промышленности ситуация – при увеличивающейся армии выпускников вузов растет нехватка молодых, владеющих современными технологиями инженеров. Несмотря на то, что в вузах на гуманитарных и управленческих специальностях фактически нет бюджетных мест, соотношение между «физиками и лириками» по-прежнему склоняется в пользу последних. Статистика свидетельствует, и вакансии крупных российских кадровых агентств подтверждают, что изо дня в день растет спрос на специалистов таких отраслей, как информационные технологии (IT), машино- и приборостроение, энергетика, строительство. А прогноз рынка труда на 2015 год и вовсе неутешительный: тотальный дефицит специалистов технических (инженерных) специальностей.

Задача высшего образовательного учреждения всегда была, и сегодня это особенно актуально, в том, чтобы не только выучить, но и правильно сориентировать молодых людей: они должны быть востребованы и иметь возможность самореализации. «Ахиллесовой пятой» высшего технического образования также всегда была, хотя и в разные времена в разной же степени, его серьезная оторванность от производства. Здесь центральной проблемой является гармоничное сочетание сильных фундаментальных основ инженерных знаний с современным существом будущей специализации. Отсюда появляется значительный пробел между подготовкой кадров в вузе и реализацией их знаний на практике. Принципиально это проблема не только каждого отдельного вуза, но и всей системы образования, которая усугубляется не всегда внятно проводимыми реформами и определенными деформациями отношений высшей школы и предприятий, порожденных рынком. В то же время современная российская промышленность, как-то: машиностроение, радиоэлектроника, строительство, другие отрасли – остро нуждается в специалистах-практиках, сочетающих специфические профессиональные знания с умением работать в команде, знанием современных IT-технологий.

Современная инновационная деятельность в сфере инженерного образования также сильно затруднена и рядом серьезных глобальных проблем, стоящих перед ним, даже если это образование признается качественным. Как отмечают многие зарубежные исследователи, противоречия между современной инженерной практикой на производстве и преподаванием дисциплин в вузе с каждым годом накапливаются все больше. Современный студент сильно отличается от своего предшественника, но преподаватели продолжают учить так, как 30-40 лет тому назад учили их самих. Проблема серьезно усугубляется отсутствием молодых преподавателей в вузе, а инертность образовательных программ и курсов – также одна из главных проблем сегодняшнего инженерного образования.

Процесс инженерного образования должен рассматриваться как освоение наиболее характерных для инженерного сообщества компетенций: овладение важнейшим для той

или иной группы данного сообщества инструментарием, существенейшим из которых, практически для всех групп, сегодня является использование вычислительных систем, а также умение работать с технической литературой. Все это способствует приобретению необходимых инженерных навыков.

Сущностью инженерной деятельности являются не только расчеты, но и разработка изделий (например, дизайн), важными элементы которой – творческий подход и открытость в смысле возможности множественных технических решений для данной проблемы. Как правило, грамотные выпускники вузов «находятся в предположении» существования единственного того или иного численного решения при анализе какой-либо инженерной проблемы. Однако современное программное обеспечение в связке с суперкомпьютерами позволяет быстро и эффективно решать поставленные задачи в весьма широком круге возможностей, освобождая инженера на творческую самореализацию: работа в ключе «что, если», модификация деталей и, что пожалуй, особенно важно, оптимизация конструкции (процесса), которая как раз обеспечивает многовариантность получаемого решения в зависимости от тех или иных параметров, и т.д.

В современном производстве компьютер из вспомогательного средства превращается в некий «центральный пункт», вокруг которого вращается вся творческая деятельность инженера. Однако «зацикливание» последнего только на одном направлении деятельности, только на персональном компьютере для вычислений приводит к своеобразной одномерности инженерного мышления, это означает тупик развития и невозможность решить сложную задачу. У современного инженера должен быть широкий кругозор как в методах решения задач, так и в средствах, с помощью которых эти решения разыскиваются.

Развитие суперкомпьютерных технологий с позиций чисто вычислительных мощностей (ресурсов) привело в наши дни научное и инженерное сообщество к возможности ставить и решать такие классы естественно-научных и инженерных задач, которые были немыслимы для инженеров и исследователей даже 10 лет тому назад. При этом в части инженерных задач главным инструментом здесь являются CAE (Computer-Aided Engineering) технологии и их параллельные реализации для суперкомпьютеров. CAE – общее название для программ и программных пакетов, предназначенных для решения различных инженерных задач: расчётов, анализа и симуляции физических процессов. Сегодня CAE является основным инструментом математического моделирования в инженерной области.

Математическое моделирование и инженерное знание

Все современное естественно-научное и инженерное знание представляет собой совокупность многотысячелетнего человеческого опыта. Причем в своем подавляющем большинстве все бесчисленные составляющие этого опыта есть продукт физического эксперимента. Именно многовековой экспериментальный опыт служил основой инженерному сообществу с незапамятных времен, примерно до конца XVIII века, в создании всех машин и механизмов, служивших человеку.

В начале XIX, когда стал массово развиваться железнодорожный транспорт и паровой флот, стала все более и более проявляться необходимость создания теоретических основ того, что мы сегодня понимаем под инженерным анализом и проектированием. Полномасштабная потребность и значимость инженерного анализа окончательно сформировались во второй половине – конце XIX века, а с началом XX века, когда очень быстрыми темпами стали появляться принципиально новые механизмы и машины, начали формироваться и новые требования к самому существу инженерного анализа. В связи с последним важно будет отметить, что именно глубокое понимание этих проблем и стремление вывести Россию на передовые мировые научно-технические и

экономические позиции лежали в основе самой идеи создания Санкт-Петербургского политехнического института.

В течение первой половины XX века многовековая практика физического эксперимента подошла к практическому исчерпанию своих возможностей при создании новых типов машин и систем, поскольку при решении все большего числа инженерных, да и многих естественно-научных проблем вообще, не позволяла провести сам физический эксперимент либо из-за его сложности, либо по причине принципиальной его невозможности. Последние обстоятельства привели инженерное и научное сообщество к необходимости использования все более и более мощных методов математики или, говоря современным языком, методов математического моделирования. Таким образом, в течение XX века произошло революционное изменение инструментария исследования природы и всего того, что создается человеком, т.е. многовековая практика физического эксперимента стала активно дополняться и заменяться экспериментом математическим. Этому способствовали два обстоятельства: во-первых, новейшие достижения в фундаментальной и вычислительной математике, как инструменте количественного описания явлений и процессов, и, во-вторых, возникновение и мощное развитие вычислительной техники.

Активное внедрение математического моделирования [1-3] в инженерную практику способствовало переходу на качественно новый уровень в инженерных расчетах и проектировании всего множества машин и систем во всех отраслях промышленности. Особенно сильно это новое качество стало проявлять себя в последние десятилетия в связи с широким применением суперкомпьютеров. Но прежде чем раскрывать существо этого качества, необходимо понять, как и в каких направлениях компьютерные технологии повлияли на развитие промышленности. При этом сразу же отметим, что сфера применения компьютеров в промышленности, практически одновременно с их внедрением в нее, стала быстро расширяться и, конечно, не исчерпывалась инженерными расчетами.

Суперкомпьютерные технологии

«Современный уровень развития вычислительной техники и методов математического моделирования дает уникальную возможность для перевода как промышленного производства, так и научных исследований на качественно новый уровень. Цифровые модели сложных конструкций, точное описание и воспроизведение природных явлений и процессов, тонкая многопараметрическая оптимизация — все это становится реальным сегодня. Для индустрии — это повышение конкурентоспособности на мировом рынке, для науки — это завоевание лидирующих позиций нашими учеными, и все это вместе составляет один из важных элементов перехода к инновационной экономике. Именно поэтому суперкомпьютерные технологии отнесены руководством страны к приоритетным направлениям модернизации экономики и технологического развития», – ректор Московского университета, академик В.А. Садовничий [4].

Центральной проблемой, обуславливающей трудности и вопросы, встречающиеся на пути широкого внедрения суперкомпьютерных технологий в инженерное образование, является их универсальный и всеобъемлющий характер. Сущность этой универсальности заключается в том, что суперкомпьютерные технологии позволяют нам в максимальном объеме реализовать технологии математического моделирования. А это, в свою очередь, дает возможность ставить и решать во всей полноте междисциплинарные задачи, которые, как указывалось, дают возможность весьма близко подойти к описанию реального физического мира. При этом проходится вся универсальная цепочка от постановки задачи, выбора эффективных вычислительных схем, обеспечивается требуемая быстрота получения результата и, что особенно важно, инженер и ис-

следователь получают возможность полномасштабной визуализации результатов вычислений.

Но такое высокое и все более нарастающее в своих возможностях качество описания явлений природы и мира техники требует и все более высокого уровня освоения инженерным сообществом самих фундаментальных основ инженерного знания. При этом сами эти основы становятся малоразличимыми со знанием естественно-научным. Именно здесь и кроется проблема слабого внедрения суперкомпьютерных технологий в инженерное образование – высшая техническая школа, в своем подавляющем большинстве, не готова и во многих случаях принципиально не способна к радикальной и быстрой перестройке учебного процесса на такие технологии. Еще раз подчеркнем, что суперкомпьютерные технологии революционно изменили весь процесс инженерной деятельности, а это требует радикальной перестройки образовательного процесса подготовки инженерных кадров.

Какими же видятся пути решения обозначенных проблем? Во-первых, это полномасштабное внедрение как CAE (Computer-Aided Engineering) технологий, так и суперкомпьютерных технологий, которые придают совершенно новый характер деятельности инженера, – она все более и более приобретает характер деятельности исследовательской, опирающейся на фундаментальное знание. Во-вторых, в недалеком будущем произойдет смена самой парадигмы инженерной деятельности. Что имеется в виду? Рассмотрение всей последовательности этапов работы инженеров – разработчиков новой техники и систем (конструкторов, расчетчиков, проектировщиков и т.д.), т.е. тех категорий, которые заняты созданием новых машин, механизмов и систем, ведет к выводу – эта деятельность в ближайшем будущем «обречена» на все более и более творческий характер, при которой сама проектно-конструкторская, рутинная, расчетно-технологическая работа будет возложена на расчетно-технологические вычислительные среды. Роль инженера разработчика будет определяться такой творческой деятельностью, как разработка концепции нового изделия, включая эффективность и быстроту его производства, а также определение его эксплуатационно-технических характеристик.

Но использование CAE-технологий, даже в их полномасштабной связке с CAD (Computer-Aided Design) и CAM (Computer-Aided Manufacturing) технологиями, есть только один из сегментов того, что в недалеком будущем станет основой нового содержания инженерного образования. Как нам видится с позиций сегодняшнего дня, само содержание инженерного образования должно включать в себя преподавание реализованной в суперкомпьютерном варианте группы дисциплин, в основе которых лежит триада CAD/CAE/CAM-технологий. И здесь задача ведущих технических вузов состоит в том, чтобы, во-первых, предложить концепцию инженерного образования на базе суперкомпьютерных технологий и, во-вторых, определить те группы дисциплин, которые будут составлять некоторое ядро – основу такого образования.

На сегодняшний день Санкт-Петербургский государственный политехнический университет подготовил проект создания Суперкомпьютерного центра на базе вуза, который подержан Минэкономразвития, Минобрнауки и Минкомсвязи России и будет реализован в 2013 – 2014 гг. Важнейшим аспектом работы Суперкомпьютерного центра станет переход всей системы учебной работы на его основу, что позволит вузу готовить кадры мирового уровня с использованием наиболее передовых технологий.

Литература

1. Самарский А.А., Михайлов А.П. Математическое моделирование: Идеи. Методы. Примеры. – 2-е изд., испр. М.: Физматлит, 2001. 320 с.

2. Болдырев Ю.Я. Суперкомпьютерные технологии как современное воплощение междисциплинарного подхода в научно-образовательной деятельности // СПб. Научно-технические ведомости СПбГПУ. № 4, 2010. С. 99–106.
3. Болдырев Ю.Я., Петухов Е.П. Суперкомпьютерные технологии и их приложения. Учебное пособие. СПбГПУ, 2010. 92 с.
4. Суперкомпьютерные технологии в науке, образовании и промышленности / Под редакцией: академика В.А. Садовничего, академика Г.И. Савина, чл.-корр. РАН Вл.В. Воеводина. М.: Изд-во МГУ, 2009. 232 с.

СОЗДАНИЕ ВИРТУАЛЬНЫХ ЛАБОРАТОРНЫХ ПРАКТИКУМОВ С ПРИМЕНЕНИЕМ ОБЛАЧНЫХ ТЕХНОЛОГИЙ ВТОРОГО ПОКОЛЕНИЯ

А.В. Богачева, А.В. Духанов

НИИ наукоемких компьютерных технологий
Санкт-Петербургского национального исследовательского университета
информационных технологий, механики и оптики

Рассматривается применение облачных технологий в образовательной сфере на примере многопрофильной инструментально-технологической платформы CLAVIRE. Представлены подходы к междисциплинарным виртуальным лабораторным практикам для развития составляющих компетенций, основанные на использовании распределённых вычислительных ресурсов для пользовательских приложений и организации и управления их выполнением.

Введение

Развитие образовательной сферы требует применения современных технологий для оптимизации рабочего процесса. Использование облачных технологий актуально во многих аспектах, что позволяет автоматизировать деятельность во многих областях. Применение облачных решений в образовании решает ряд проблем в обеспечении учебного процесса: снижение накладных расходов, доступ из любого места, где доступна глобальная сеть, и т.д. Для решений комплексных задач в междисциплинарных науках используется комплексная интеграция разных программных комплексов [1]. Другими не менее важными приоритетами, согласно вышеуказанному источнику, являются: автоматизация, ориентированность на клиента, безопасность, простота и эффективность.

1. Модель обучения

Комплексная интеграция разных программных комплексов находится в одном из приоритетов развития облачных технологий на ближайшее будущее. Здесь формируются особенности и требования к облачной инфраструктуре для автоматизации учебного процесса подготовки специалистов и учёных всех направлений, в которых применяются информационные технологии:

1. Поддержка исполнения композитных приложений (КП), состоящих из отдельных вычислительных сервисов распределенной среды, объединенных в целях решения общей задачи.
2. Ориентация на технологии удаленного взаимодействия с пользователем через web-браузер, без использования специализированных клиентских приложений, в рамках модели SaaS или AaaS облачных вычислений.
3. Обеспечение интеллектуальной поддержки пользователей, как при выполнении ВЛП.
4. Обеспечение информационной безопасности ВЛП в рамках модели системы с открытым контуром.

5. Возможность эффективного совокупного использования ресурсов существующих суперкомпьютеров, Грид-сред и сред облачных вычислений.
6. Поддержка «горячего» подключения в ВЛП новых прикладных пакетов и вычислительных ресурсов с минимальными усилиями со стороны их разработчиков или владельцев.

Для обеспечения перечисленных выше требований к инфраструктуре ВЛП требуется развитие методической составляющей организации учебного процесса с использованием КП. Наиболее распространенным подходом к представлению КП является формализм потока работ, или workflow (WF), который позволяет описать связи между отдельными операциями в распределенной среде в виде ориентированного графа. В целом WF-представление может рассматриваться как основа для формализации процесса изучения междисциплинарных явлений, на основе которого может быть построен весь цикл обучения в рамках выполнения ВЛП.

Можно определить три подхода к использованию междисциплинарных ВЛП для развития составляющих компетенций:

1. Обучаемый получает непосредственный доступ к нему без возможности изменения внутренней структуры в процессе выполнения ВЛП.
2. Обучаемый имеет возможность пошагово исследовать процессы, происходящие в «междисциплинарной» системе, анализируя выходные данные каждого из пакетов и интерпретируя их влияние на результаты последующих вычислений.
3. Обучаемый изучает применение отдельных прикладных пакетов, включая подготовку входных данных, запуск на исполнение, визуализацию и интерпретацию результатов расчетов.

2. Инфраструктура поддержки виртуальных лабораторных практикумов

Виртуальный лабораторный практикум основан на подходе использования распределённых вычислительных ресурсов для пользовательских приложений и организации и управления их выполнением. Такого подхода придерживается многопрофильная инструментально-технологическая платформа CLAVIRE, которая активно использует workflow. Она предназначена для создания, исполнения и предоставления сервисов доступа к предметно-ориентированным высокопроизводительным композитным приложениям, функционирующим в облаке неоднородных вычислительных ресурсов корпоративного уровня, уровня центров компетенции, центров обработки данных, инфраструктур экстренных вычислений и распределенных хранилищ данных. К преимуществам CLAVIRE можно отнести универсальность технологии, обеспечивающую создание и поддержку облачных сред различного назначения, включая корпоративный уровень, уровень центров компетенции, центров обработки данных, а также коллаборативных и публичных распределенных сред и сред экстренных вычислений. Также она содержит поддержку технологий интерактивного управления облачными приложениями, обеспечивающую возможность создания распределенных систем реального времени, систем интерактивной визуализации и виртуальной реальности, а также инфраструктуры ситуационных центров. Обеспечение информационной безопасности пользователей и провайдеров ресурсов и сервисов происходит за счет инновационной модели защиты от нелегитимного пользователя в облачных системах с неопределенным контуром.

В основе функционирования композитных приложений лежит библиотека вычислительных пакетов различного назначения: моделирование атмосферных явлений и волнения, расчёты при движении судна на поверхности воды. Библиотека пакетов является пополняемой и расширяемой. Благодаря встроенному инструментарию описания пакетов – EasyPackage [2], один и тот же пакет можно представить по-разному в зависимости от целевой аудитории, например теоретические исследователи или специали-

сты-предметники. Кроме того, в CLAVIRE встроены возможности визуализации результатов работы пакетов, которые реализованы встроенными средствами или специализированными внешними пакетами, например «GraphVis». Таким образом, можно утверждать, что при наличии необходимого для обучения вычислительного программного обеспечения можно с помощью CLAVIRE практически полностью обеспечить соответствующую сторону учебного процесса. Исключением являются визуальные средства разработки приложений и программные средства с интерактивной визуализацией, которые при необходимости устанавливаются на локальный компьютер пользователя. Однако в CLAVIRE существует возможность ими управлять, если предоставлен соответствующий доступ и сценарий.

Архитектура подсистемы информационного взаимодействия (ПИнфВ) на основе технологии Web 2.0 представлена на рис.1. Все действия с подсистемой пользователь выполняет в основном через компонент «Сайт»: регистрация; авторизация; просмотр страниц; добавление, редактирование и удаление как личной информации, так и разнообразных файлов. Для запуска пакетов и приложений «Сайт» вызывает либо компонент «Доступ», либо компонент интеллектуального интерфейса удаленного доступа.

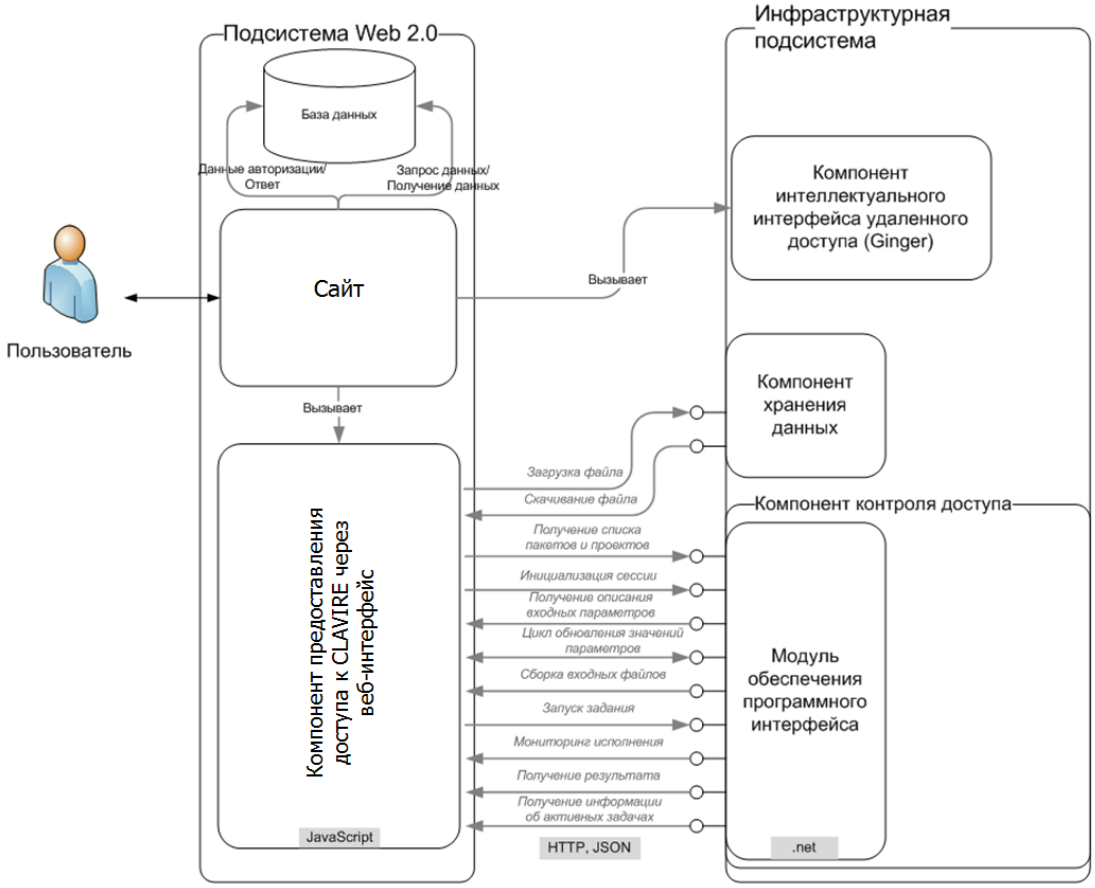


Рис. 1. Основные операции взаимодействия компонента «Доступ» с ПИнфВ

Благодаря такой архитектуре, логика ввода, бизнес-логика и логика интерфейса разделяются, но достаточно тесно взаимодействуют друг с другом. Это разделение позволяет работать со сложными структурами при создании сайта, так как обеспечивает одновременную реализацию только одного аспекта.

Заключение

В рамках данного исследования разработана и обоснована архитектура системы для организации виртуальных лабораторных практикумов в рамках модели облачных вычислений с использованием доступа через web-интерфейс на основе платформы CLAVIRE. В целом это обеспечивает функционирование систем дистанционного обучения с частичным устранением эффектов эрозии знаний, иллюзии интерактивности и информационной избыточности [3].

Литература

1. Intel's Vision of the Ongoing Shift to Cloud Computing. Портал Корпорации Intel, 2010. – [http://www.intel.ru/content/www/ru/ru/cloud-computing/cloud-computing-intel-cloud-2015-vision.html].
2. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Известия высших учебных заведений. Приборостроение. 2011. №10. С. 36-43.
3. Болгова Е.В. и др. Инфраструктурное обеспечение виртуальных лабораторных практикумов для междисциплинарных образовательных программ в рамках концепции облачных вычислений // Современные проблемы науки и образования. 2012. №5 (43). – [http://www.science-education.ru/105-6977].

ИНФРАСТРУКТУРА СТАЖИРОВКИ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ МОЛОДЫХ СПЕЦИАЛИСТОВ И УЧЁНЫХ В ОБЛАСТИ НАУЧНОЙ КОМПЬЮТЕРНОЙ ГРАФИКИ И ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ НА БАЗЕ МНОГОПРОФИЛЬНОЙ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКОЙ ПЛАТФОРМЫ CLAVIRE

М.А. Балахонцева, А.А. Безгодов, А.З. Билятдинова, Е.В. Болгова, А.В. Духанов

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

E-mail: dukhanov@niuitmo.ru

Представляется комплекс программных средств и сервисов, обеспечивающий в облачной среде процесс повышения квалификации и стажировки молодых учёных и специалистов в области научной компьютерной графики и виртуальной реальности. Данный процесс проходил в рамках проекта «Облачные технологии высокопроизводительных вычислений в задачах интерактивной 3D-визуализации сложных процессов и систем» в рамках ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы. В процессе работы приглашённые участники представили прикладные расчётные пакеты, разработали средства визуализации на основе предложенного шаблона и интегрировали программные средства в составе композитного приложения МИТП CLAVIRE.

Введение

Современные технологии высокопроизводительных вычислений порождают экстремально большие объемы расчетных данных. Их анализ и интерпретация формальными методами в большинстве случаев весьма затруднена; как следствие, это требует использования разного рода когнитивных методов интерпретации, в том числе – с использованием средств научной визуализации и виртуальной реальности. Несмотря на то, что сопряжение суперкомпьютерных систем и систем виртуальной реальности в настоящее время не представляет научного интереса, интенсивное развитие глобальных сред распределенных вычислений в сетях Интернет в рамках концепции облачных вычислений предъявляет качественно новые требования к развитию данного направления, что и определяет актуальность исследования.

Освоение приглашёнными специалистами технологий виртуальной реальности в связке с облачными вычислениями требует создания соответствующей инфраструктуры, которая обеспечит доступ к облачной среде высокопроизводительных вычислений и возможность интеграции прикладных расчётных пакетов и средств визуализации.

1. Инфраструктура

Участники проекта представляли различные города, расположенные на очень большой территории. Ввиду их большой загруженности по основному месту работы собрать их в НИУ ИТМО хотя бы на неделю было очень сложно. Поэтому работа с ними осуществлялась дистанционно с применением информационно-телекоммуникационных технологий, включая МИТП CLAVIRE с соответствующими веб-интерфейсами (рис. 1).

Каждый из привлечённых коллег представил вычислительный пакет, решающий прикладную задачу. Все пакеты работают в режиме командной строки и выполняются в операционной системе либо семейства Windows, либо Linux. Ввиду того, что пакеты нескольких привлечённых участников могли корректно работать только на их ресурсах, возникла необходимость присоединения этих ресурсов к вычислительной инфраструктуре CLAVIRE. Оставшаяся часть пакетов была временно установлена на вычислительных ресурсах НИУ ИТМО.

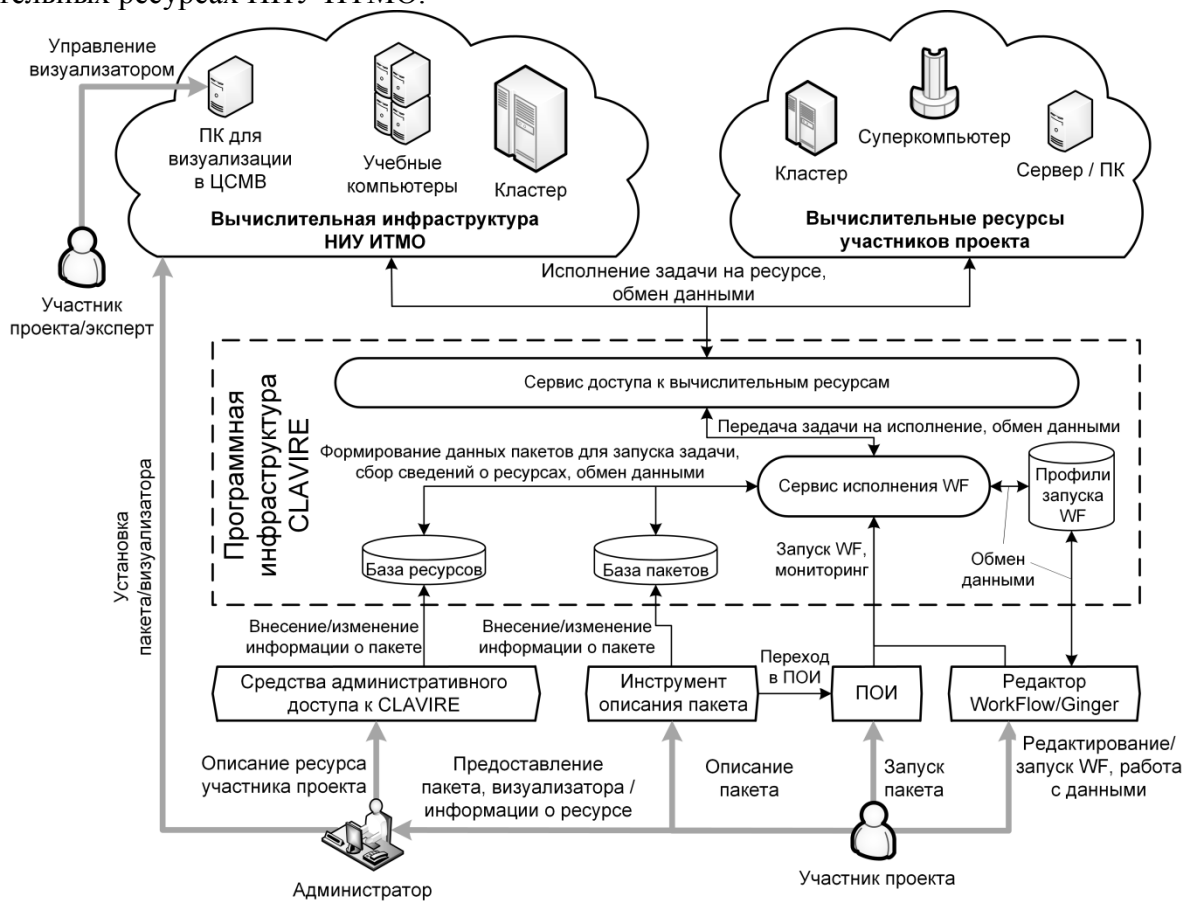


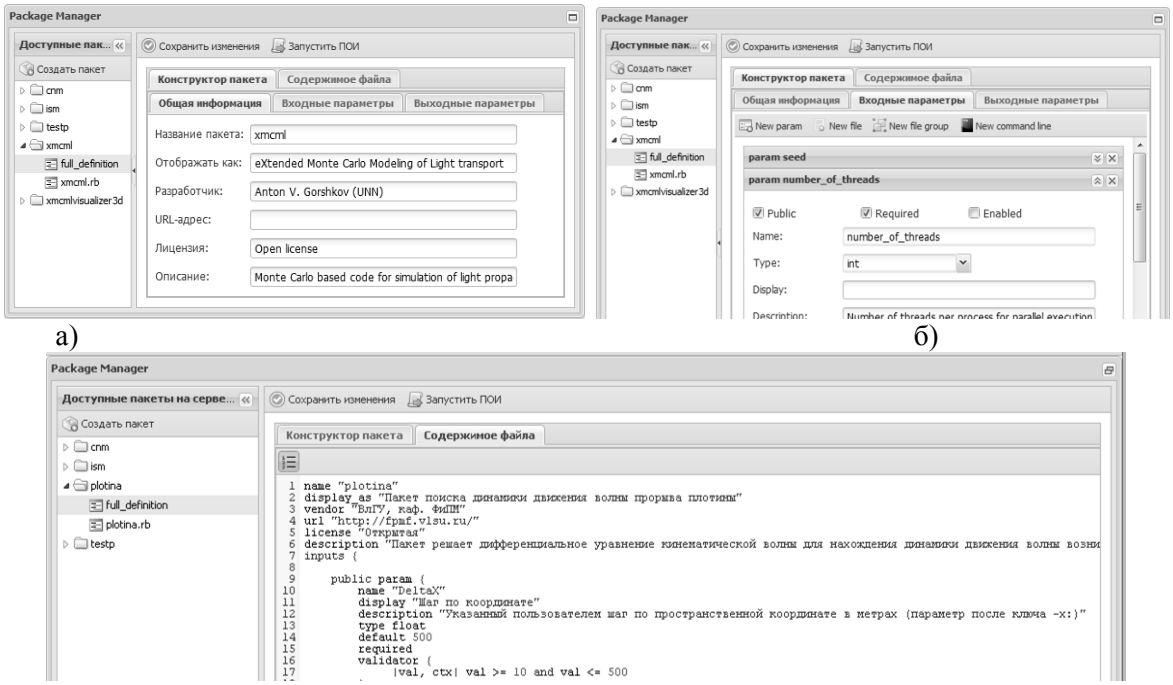
Рис. 1. Схема использования МИТП CLAVIRE участниками проекта

Процесс взаимодействия приглашённого участника с CLAVIRE проходил следующим образом. После предоставления участником прикладного пакета (архива или инсталляционного дистрибутива) администратор устанавливал его на вычислительные ресурсы инфраструктуры НИУ ИТМО и вносил необходимые записи в базу пакетов. В случае если участник использовал свой ресурс, то для его присоединения к CLAVIRE администратор вносил соответствующие записи в базу ресурсов.

После установки пакета / присоединения ресурса привлечённый участник выполнял описание пакета с помощью специализированного средства «Инструмент описания пакета» с веб-интерфейсом. В нём заполнялись общие сведения о пакете (рис. 2, а), его исходных и выходных данных (рис. 2, б), а также командной строке его запуска.

После завершения описания пакета пользователь может перейти к соответствующему сгенерированному скрипту (рис. 2, в) на совместимом с Ruby языке EasyPackage [1]. Здесь он может добавить классы для конвертирования как входных, так и выходных данных, изменить описание параметров, включая «защиту от дурака», значение по умолчанию. Отметим, что с помощью данного средства приглашённый участник может существенно изменить форматы входных и выходных данных, определяя работу пакета в необходимых для него рамках (защита от нелегитимного пользователя).

Проверить корректность описания пакета пользователь может с помощью запуска пакета из проблемно-ориентированного интерфейса (ПОИ). ПОИ предложит ввести/загрузить исходные данные и запустить пакет, затем покажет ход его выполнения (рис. 3). Если пакет отработает успешно, ПОИ после запуска предоставит возможность получить выходные данные и последние будут соответствовать входным данным при обычном локальном запуске пакета, то описание выполнено корректно.



а) б)

в)

Рис. 2. Инструмент описания пакетов: а – общее описание пакета, б – описание входных параметров, в – представление описания пакета на языке Easy Package

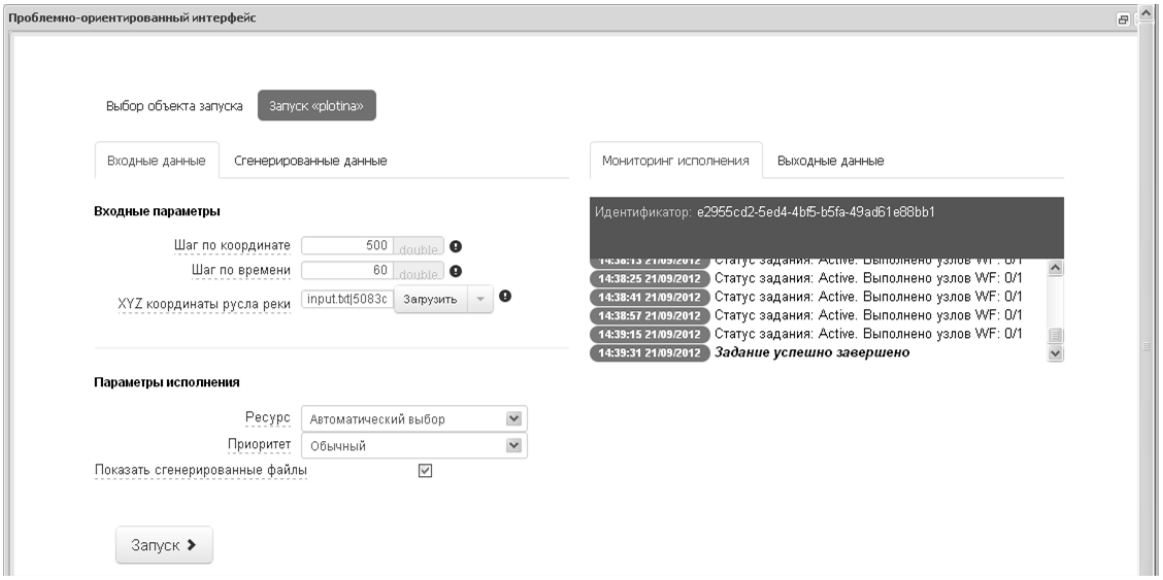


Рис. 3. Проблемно-ориентированный интерфейс

После завершения работ по описанию пакета и проверки корректности пользователь получает возможность работы с пакетом как с элементом композитного приложения в среде CLAVIRE через редактор WorkFlow/Ginger (рис. 1).

2. Обеспечение разработки визуалазатора

Второй важной составляющей процесса привлечённого участника с CLAVIRE являлась разработка визуализатора результатов работы прикладного пакета с последующей процедурой встраивания в CLAVIRE. Участникам предложено создавать визуализатор не с нуля, а на основе специально созданного «Базового визуализатора», который представляет собой класс *base_demo*, который должен являться базовым для конкретных реализаций предметных визуализаторов. В нём заложены необходимые абстрактные методы, которые обеспечивают необходимые условия для построения картинки, сформированной пользователем. При этом расчёт проекций, матриц поворота для обеспечения интерактивности визуализатора с пользователем уже встроены в класс, благодаря чему приглашённый участник имел больше возможностей акцентирования внимания на прикладной задаче.

При установке и подключении библиотек *Kinect for Windows SDK* и *Kinect for Windows Developer Toolkit* в проект можно добавить возможности управления картинкой с помощью потребительской системы захвата движений *MS Kinect* (круговой взмах руками от себя – удаление сцены, взмах одной рукой – вращение сцены и т.д.).

Если видеокарта компьютера и монитор (проекционная система) поддерживают режим стереовывода, то визуализатор может выводить на экран стереокартинку, например поляризационного типа. Для погружения в виртуальную реальность пользователь использует соответствующие очки.

На рис. 4 представлен пример вывода изображения на экран с помощью визуализатора в центре ситуационного моделирования и визуализации (ЦСМВ).

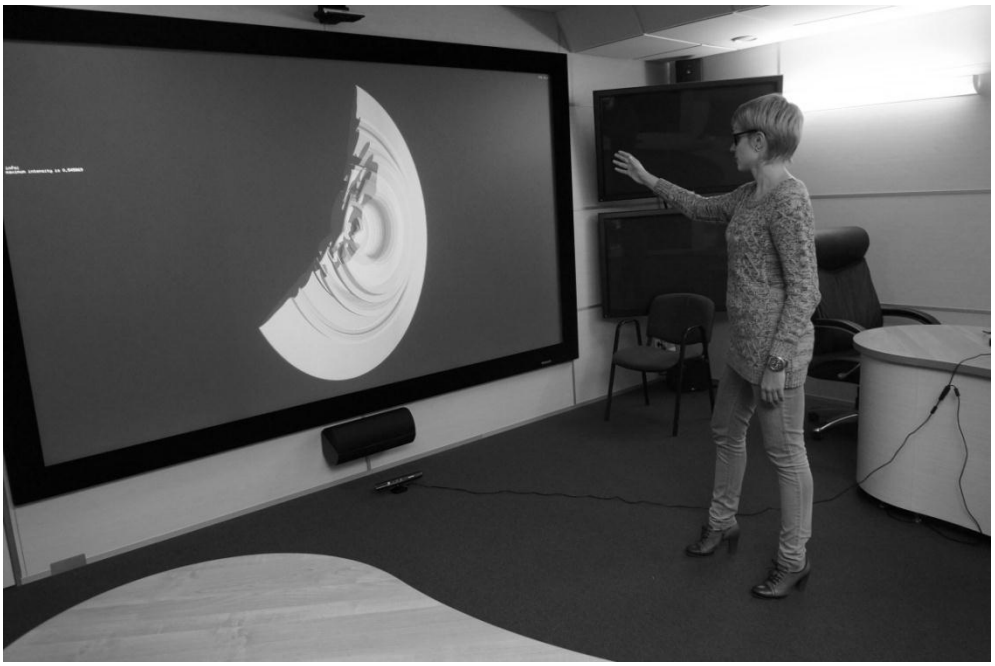


Рис. 4. Управление визуализатором в ЦСМВ НИУ ИТМО

3. Интеграция прикладного пакета и визуализатора в облачную вычислительную среду CLAVIRE

После разработки визуализатора привлечённый исполнитель передавал его администратору либо устанавливал его на своём ресурсе. Чтобы иметь возможность запуска визуализатора в составе композитного приложения, пользователь описывал его тем же путём, как и для прикладного пакета.

Композитное приложение представляет собой скрипт на специализированном языке EasyFlow [4]. Для его разработки и запуска участник использовал редактор WF (рис. 1). После команды пользователя на запуск приложения сервис исполнения WF проверял корректность скрипта на основе данных базы пакетов и в случае отсутствия ошибок ставил его на очередь выполнения. При наступлении очереди задействованные пакеты запускались на заранее определённых по данным базы ресурсов вычислительных устройствах. Предоставление входных и получение выходных данных пользователь-участник осуществлял с помощью редактора WF/Ginger [3] (рис. 5).

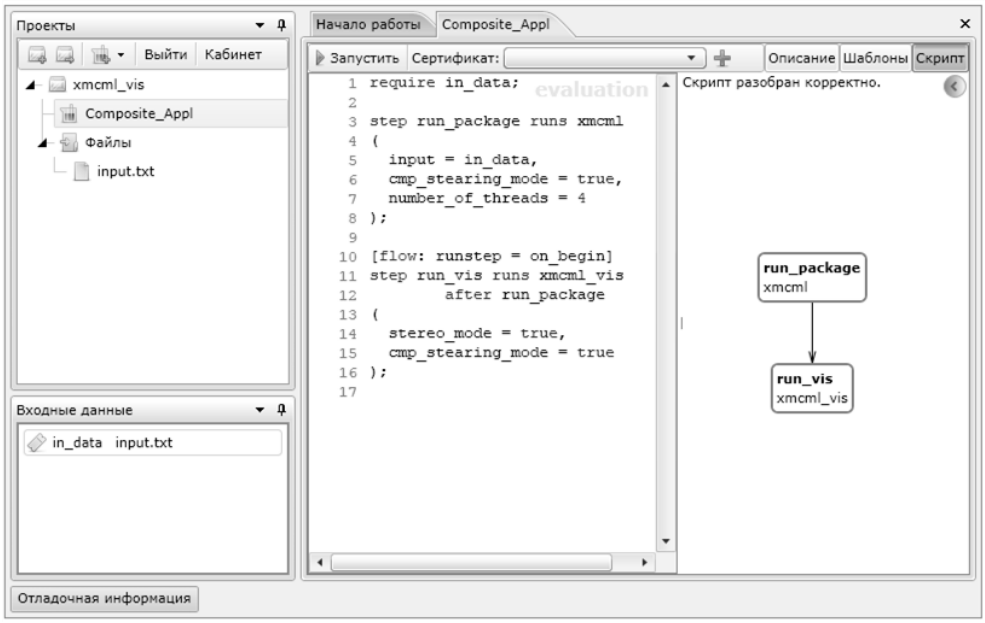


Рис. 5. Интерфейс разработки композитных приложений (WorkFlow)

Заключение

В настоящей работе представлен комплекс программных средств и сервисов, обеспечивающий в облачной среде процесс повышения квалификации и стажировки молодых учёных и специалистов в области научной компьютерной графики и виртуальной реальности. В процессе работы приглашённые участники представили прикладные расчётные пакеты, разработали средства визуализации на основе предложенного шаблона и интегрировали программные средства в составе композитного приложения МИТП CLAVIRE.

Литература

1. Болгова Е.В. и др. Инфраструктурное обеспечение виртуальных лабораторных практикумов для междисциплинарных образовательных программ в рамках концепции облачных вычислений // Современные проблемы науки и образования. 2012. №5 (43) – [http://www.science-education.ru/105-6977].

2. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Известия высших учебных заведений. Приборостроение. 2011. №10. С. 36–43.
3. Ковальчук С.В. и др. Организация человеко-компьютерного взаимодействия в средах компьютерного моделирования на базе облачной инфраструктуры // Прикладная информатика. 2012. № 5 (41). С. 89–102.

СИСТЕМА АКТИВНОГО МОНИТОРИНГА ПАКЕТОВ И РЕСУРСОВ ПЛАТФОРМЫ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

В.А. Карбовский

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

E-mail: dalamarik@gmail.com

Предлагается подход применения предметно-ориентированного тестирования корректности выполнения пакетов и работоспособности ресурсов, использующихся облачными платформами, на примере системы активного мониторинга пакетов и ресурсов МИТП CLAVIRE.

Введение

Непрерывно возрастающая сложность современных платформ облачных вычислений связана с увеличением числа вычислительных ресурсов данных платформ, а также прикладных пакетов. Каждый вычислительный пакет имеет свой собственный формат как входных, так и выходных данных, часть пакетов может завершить свое выполнение без ошибки, однако результат, по тем или иным причинам, будет некорректен. В связи с этим традиционные методы тестирования не могут решить задачу проверки правильности выполнения пакета или работоспособности ресурса, обслуживающего данный пакет. Для решения поставленной задачи следует применять подход предметно-ориентированного тестирования, т.е. тестирование, направленное на предметное содержание объекта.

1. Предметно-ориентированное тестирование

Тестирование ресурсов, входящих в состав МИТП CLAVIRE [1], направлено на две группы предметов, предполагая выполнение следующих категорий проверок [2]:

1. Тестирование вычислительных ресурсов. Предполагает определение доступности подключенных ресурсов, корректной их настройки для работы в среде. Проверяется тестовым запуском тестового пакета.
2. Тестирование вычислительных пакетов. Позволяет определить, корректно ли развернуты вычислительные пакеты на ресурсах за счет тестового запуска специально сформированного задания.

Тестирование основано на исполнении заранее подготовленных сценариев, позволяющих проверять работоспособность среды. Сценарии подготавливаются экспертами пакета, обладающими необходимым опытом. Эксперт формирует входные данные, а также указывает ожидаемые выходные данные, основываясь на априорном знании. Результатом работы сценария является выходной файл, который проверяется на соответствие ожидаемому. Таким образом, проверка каждого сценария формируется индивидуально.

Формализованный сценарий (рис. 1) состоит из трех основных частей. Первая часть – описание входных данных (элемент <data>), в которой задается соответствие между файлами и параметрами. Основная часть (<efscript>) – непосредственно скрипт выполнения задания, который формализуется в виде скрипта на специальном предметно-ориентированном языке программирования – EasyFlow [3]. Такой подход является пред-

почтительным, так как обладает гибкостью, выразительностью, позволяет поддерживать тесты в понятном и читаемом виде. И, наконец, блок выходных данных (<checkblock>), в котором заданы соответствия ожидаемых значений и выходных файлов.

```
<?xml version="1.0" encoding="UTF-8"?>
<taskscript>
	<data>
		<file name="in0" efname="input0"/>
		<file name="in1" efname="input1"/>
		<param name="in_format" value="short"/>
	</data>
	<efscript>
		require in0, in1;
		step a runs testp
		(
			inf0 = in0,
			inf1 = in1,
			operation = "plus"
		);
	</efscript>
	<checkblock>
		<file name="test.out" value="successful"/>
	</checkblock>
</taskscript>
```

Рис. 1. Формализованный сценарий

Таким образом, тестирование сводится к успешному или неуспешному (с выводом информации об ошибке) выполнению заранее подготовленного сценария.

2. Реализация системы

Система мониторинга является веб-приложением, доступным из браузера, реализующим трехуровневую архитектуру [2], и была разработана в рамках проекта «Создание высокотехнологичного производства комплексных решений в области предметно-ориентированных облачных вычислений для нужд науки, промышленности, бизнеса и социальной сферы» при реализации Постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства».

3. Заключение

Таким образом, разработанная система активного мониторинга пакетов и ресурсов позволяет проводить необходимое предметно-ориентированное тестирование вычислительных пакетов и ресурсов в рамках платформы CLAVIRE.

Литература

1. Бухановский А.В., Васильев В.Н., Виноградов В.В., Смирнов Д.Ю., Сухоруков С.С., Яппаров Т.Г. CLAVIRE: Перспективная технология облачных вычислений второго поколения // Изв. вузов. Приборостроение. Т. 54. №10. Современные тенденции развития распределенных вычислений, 2011. С. 7–14.
2. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Изв. вузов. Приборостроение. 2011. Т. 54, № 10. С. 36–43.
3. Barnett M., Grieskamp W., Nachmanson L., Schulte W., Tillmann N., Veanes M. Model-Based Testing with AsmL.NET // Proc. 1st European Conference on Model-Driven Software Engineering. December 2003. P. 43.
4. Fowler M. Patterns of Enterprise Application Architecture. Addison Wesley, 2003. P. 45–67.

ОСОБЕННОСТИ РАЗРАБОТКИ ИНТЕРАКТИВНЫХ КОМПОЗИТНЫХ ПРИЛОЖЕНИЙ

К.В. Князьков, С.В. Ковальчук

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики
E-mail: constantinvk@gmail.com

Рассматриваются принципы проектирования и разработки интерактивных композитных приложений для их исполнения в распределенной вычислительной среде. Интерактивные композитные приложения могут использоваться в качестве основы для создания систем визуализации, системы реального времени, а также для обеспечения технологии computational steering. Апробация разработанного прототипа рассматривается на примере сервиса моделирования флэшмоб-акций.

Введение

Формализм workflow широко распространен для представления сложных научных композитных приложений. Описание комплексной задачи в виде графа связанных блоков позволяет автоматически учитывать зависимости между блоками и распараллеливать исполнение на несколько распределенных ресурсов. Во многих существующих системах исполнения workflow используется модель запуска в пакетном режиме, когда сам workflow и его отдельные подзадачи после запуска переходят в режим непрерывного исполнения без возможности взаимодействия с ними [1]. Такой подход прост в реализации и оправдан для решения задач, не связанных с необходимостью работы в реальном времени. Но существуют задачи, которые сложно адаптировать к пакетному режиму работы, либо, если это возможно, адаптация приводит к значительной потере эффективности использования ресурсов. Это задачи: а) требующие обработки данных непрерывным потоком; б) требующие постоянного взаимодействия с внешними объектами и источниками данных; в) где критически важно время получения результата; г) где требуется возможность интерактивного взаимодействия с пользователем.

Вопросы перехода систем, основанных на модели workflow, от пакетного режима функционирования к интерактивному в настоящее время активно исследуются. На текущий момент существует несколько систем исполнения композитных приложений, которые поддерживают интерактивный режим работы. Например, система WS-VLAM [2] позволяет исполнять интерактивные приложения, а также поддерживает потоковую передачу данных. Помимо этого, существует ряд исследований, не связанных напрямую с workflow, но затрагивающих интерактивный режим работы задач в распределенных средах. В том числе, применение потоков данных для связи задач в Грид рассматривается в [3]. В работе [4] рассмотрены вопросы, связанные с реализацией системы на основе Грид, позволяющей планировать запуск и исполнять интерактивные задачи. В [5] приведена модель workflow с применением конвейеризации, основанная на сетях Петри. Публикация [6] посвящена вопросу сравнения моделей пакетного и потокового исполнения для задачи обработки данных с датчиков для определения опасных погодных явлений.

Таким образом, применение модели workflow для описания комплексных вычислительных задач ограничено пакетным режимом исполнения. Данная работа посвящена

вопросу использования общей модели workflow для описания как пакетных, так и сетевых, и интерактивных приложений. В работе представлена расширенная модель workflow — модель интерактивных композитных приложений, которая ориентирована на описание информационно-управляющих систем.

Интерактивные композитные приложения

Использование классической модели workflow для представления информационно-управляющих систем осложнено из-за отсутствия возможностей: интерактивного взаимодействия с пользователями для привлечения экспертов и операторов в работу композитного приложения; непрерывного режима функционирования; работы в реальном масштабе времени; управления реальными объектами.

Для удовлетворения представленным требованиям и для преодоления ограничений вследствие пакетного исполнения вводится расширение модели workflow — *интерактивные workflow*, основанные на нескольких ключевых принципах:

1. поддержка workflow и отдельных блоков, исполняющихся долгое время;
2. поддержка механизмов управления извне поведением исполняющихся заданий и их жизненным циклом;
3. поддержка коммуникации между блоками workflow во время исполнения;
4. возможность изменения workflow во время исполнения за счет сценария workflow, а также за счет внешнего управления.

Первый принцип снимает ограничение пакетного исполнения и позволяет встраивать пакеты, работающие долгое время (например, серверы). Второй принцип позволяет управлять такими пакетами, например останавливать их или менять конфигурацию. Коммуникация необходима для объединения нескольких интерактивных приложений в одно распределенное. Четвертый принцип говорит о возможности структурного изменения композитного приложения во время исполнения, например за счет добавления блоков. Реализация перечисленных принципов позволяет строить системы взаимодействия с пользователем за счет возможностей управления и коммуникации, строить системы визуализации и управления внешними объектами за счет коммуникации, а также системы обработки данных с внешних источников данных. Наличие возможности управления блоками позволяет реализовать принцип *computational steering* для интерактивного управления процессами моделирования в реальном времени.

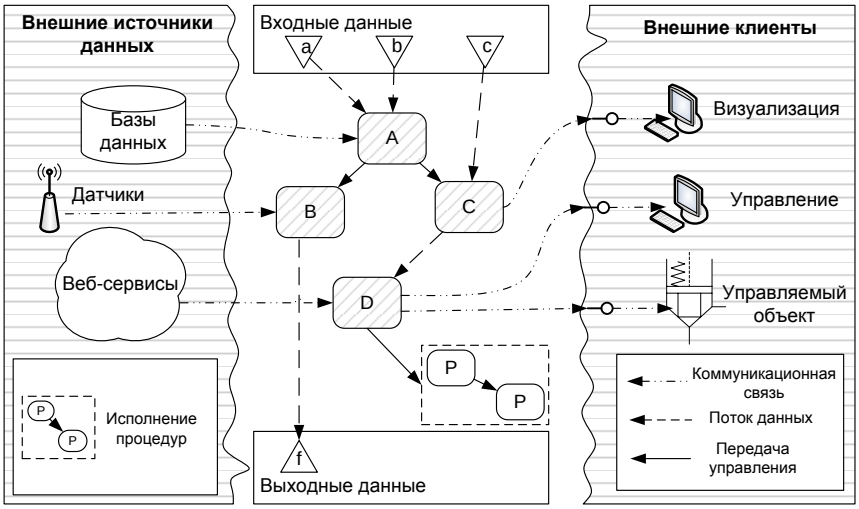


Рис. 1. Модельное интерактивное приложение

На рис. 1 представлена схема модельного интерактивного композитного приложения, на которой представлены коммуникационные связи, а также возможность запуска

пакетных процедур из интерактивных блоков (согласно принципу №4). Из схемы видно, что блоки workflow во время своей работы могут обращаться к внешним, по отношению к платформе, ресурсам для получения информации, а также являться источниками информации для внешних клиентов.

Реализация

Для воплощения принципов модели интерактивных композитных приложений в рамках платформы CLAVIRE [7] был выбран и реализован метод коммуникации узлов, метод передачи управляющих воздействий, а также реализован режим непрерывного исполнения интерактивных блоков. В качестве метода коммуникации был выбран механизм обмена сообщениями между блоками workflow по сети. Для установления коммуникационных зависимостей между блоками workflow используются порты, основной характеристикой которых является протокол, обозначающий способ передачи данных. В ходе реализации прототипа было выявлено, что эффективной абстракцией для коммуникационной связи узлов является парадигма *реактивного программирования*, ориентированная на потоки данных и распространение изменений по портам. Согласно этой парадигме, выходной порт является параметром, который может изменяться в ходе исполнения приложения, при этом присоединенные к нему входные порты других узлов будут синхронно менять свое значение. Для управления интерактивными блоками используется механизм команд, а для взаимного влияния в ходе исполнения введен механизм событий.

В рамках реализации представленной выше модели синтаксис языка описания композитных приложений EasyFlow [8] был расширен за счет новых конструкций: добавлена возможность связывания узлов по портам и объявления интерактивных узлов. Компонент интерпретации workflow модифицирован за счет добавления логики обработки коммуникационной связи, а также динамического конфигурирования запускаемых интерактивных пакетов за счет передачи им конфигурационного файла, содержащего коммуникационные настройки среды. В качестве среды коммуникации и передачи команд был выбран программный сетевой транспортный уровень — ZMQ (zero message queue), в качестве формата сериализации данных — формат BSON. При этом для встраивания пакета длительного исполнения в полученную среду необходимо подключить к нему специальную программную библиотеку (см. рис. 2).

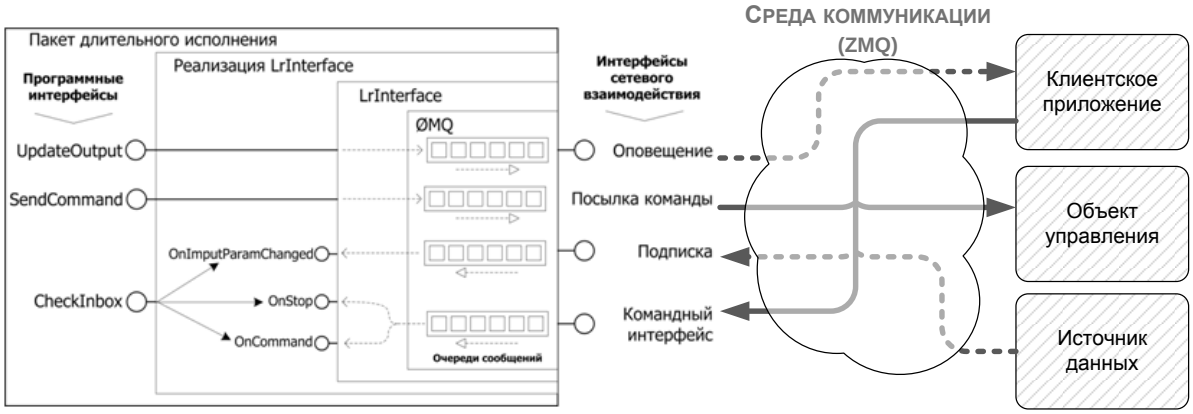


Рис. 2. Структура разработанной программной библиотеки

Пример использования

В рамках апробации предложенной модели и разработанного решения было реализовано композитное приложение — программный сервис моделирования флэшмобакций [9]. Композитное приложение предназначено для исследования критических ситуаций, связанных с террористическими действиями в условиях несанкционированных

флешмоб-акций, организованных с использованием социальных сетей. Композитное приложение представляет собой линейный граф из следующих последовательных шагов. По определенным характеристикам (полученным на основе краулинга) генерируется модельная сеть (шаг 1), на основе которой исследуется распространение информации (оповещение об акции). Для того чтобы оценить неопределенность прогноза распространения информации, используется *бутстрепирование* – формируется выборка на основе текущей сети с удалением части случайно выбранных узлов (шаг 2). Далее осуществляется моделирование распространения информации (шаг 3), чтобы в итоге определить интервальную оценку числа оповещенных (шаг 4). По характерным статистическим сценариям (минимальное, максимальное, среднее значение числа участников) выполняется анализ картины паники и эвакуации (шаг 5). На рисунке 3 изображен результат моделирования паники.

Рис. 3. Статическое определение зон с критическим значением плотности агентов

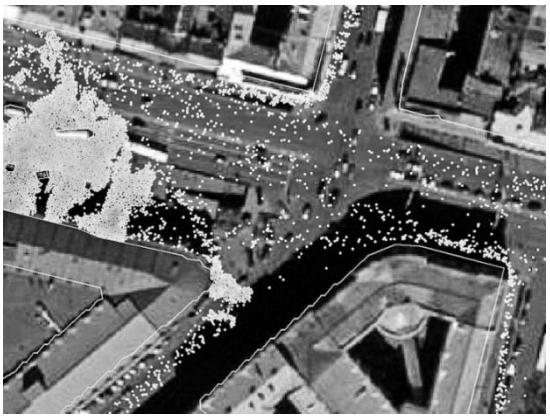


Рис. 4. Окно визуализации во время моделирования

Последний шаг выполняется на основе параллельных процессов моделирования и визуализации (рис. 4). Использование модели интерактивных композитных приложений позволило разделить эти процессы: в данном случае моделирование выполнялось на вычислительно мощном оборудовании, а визуализация осуществлялась в специализированном визуализационном оборудовании. При этом за счет выбранной технологии коммуникации визуализация запускалась параллельно на нескольких компьютерах. Указанные возможности позволяют более эффективно использовать различную вычислительную технику, а также сократить объемы передаваемых презентационных данных и обеспечивают возможность многопользовательского доступа к результатам расчетов.

Заключение

Результатами исследования, представленного в данной работе, являются: принципы построения интерактивных композитных приложений, программная реализация модели в рамках платформы CLAVIRE, апробация решения на сервисе моделирования флэшмоб-акций. Полученные в результате апробации и оценки эффективности данные позволяют говорить о применимости модели интерактивных композитных приложений и разработанной программной инфраструктуры для создания систем визуализации, систем поддержки принятия решений, для задач обработки больших объемов данных, где хранение промежуточных файлов недопустимо из-за их большого объема, для создания интеллектуальных систем управления сложными динамическими объектами.

В то же время часть вопросов, связанных с областью интерактивных приложений, остались нерассмотренными. Вопрос взаимодействия интерактивных программных модулей, работающих в различных вычислительных средах, осложняется их неоднородностью: на текущий момент нет унифицированных способов связи приложений в рам-

ках Грид и сред облачных вычислений. Планирование интерактивных приложений не может использовать в качестве основы применяемый на данный момент в платформе CLAVIRE аппарат параметрических моделей производительности [10], т.к. задачи длительного исполнения по определению могут работать бесконечно долго. К тому же необходимо исследование вопроса взаимодействия нескольких композитных приложений в общей среде, так как в реальных задачах может возникнуть необходимость взаимодействия с уникальными реальными объектами.

Литература

1. Deelman E., Gannon D., Shields M., Taylor I. Workflows and e-science: An overview of workflow system features and capabilities // Future Generation Computer Systems. 2009. V. 25, № 5. P. 528–540.
2. Wibisono A., Korkhov V., Vasunin D., Guevara-Masis V., Zhao Z., Belloum A. Workshop on Workflow Systems in e-Science // Lecture Notes in Computer Science. 2007. Vol. 4489. P. 191–198.
3. Koulouzis S., Meij E., Marshall M.S., Belloum A. Enabling data transport between web services through alternative protocols and streaming // Proceedings of the 2008 Fourth IEEE International Conference on eScience. – Washington, DC, USA: IEEE Computer Society, 2008. P. 400–401.
4. Lawenda M., Okon M., Oleksiak A., Ludwiczak B., Piontek T., Pukacki J., Meyer N., Nabrzyski J., Stroinski M. Running Interactive Jobs in the Grid Environment // Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science. 2006. Vol. 3911/2006. P. 758–765.
5. Tolosana-Calasanz R., Banares J.A., Rana O.F. Autonomic streaming pipeline for scientific workflows // Concurrency and Computation: Practice and Experience. 23, 16 (November 2011). P. 1868–1892.
6. Sims M., Kurose J., Lesser V. Streaming versus Batch Processing of Sensor Data in a Hazardous Weather Detection System // Proceedings of Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005). 2005. P. 185–196.
7. Бухановский А.В., Васильев В.Н., Виноградов В.Н., Смирнов Д.Ю., Сухоруков С.А., Яппаров Т.Г. CLAVIRE: перспективная технология облачных вычислений второго поколения // Известия вузов. Приборостроение. 2011. № 10. С. 7–13.
8. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Известия вузов. Приборостроение. 2011. № 10. С. 36–43.
9. ACLab – Демонстрационное композитное приложение моделирования флешмоб-акций – [http://acl.ifmo.ru/index2.php?ws=142].
10. Марьин С.В., Ковальчук С.В. Сервисно-ориентированная платформа исполнения композитных приложений в распределенной среде // Известия высших учебных заведений. Приборостроение. 2011. №10. С. 21–28.

РАСПРЕДЕЛЕННАЯ ПОДДЕРЖКА КОЛЛЕКТИВНОГО ПРИНЯТИЯ РЕШЕНИЙ В СИТУАЦИОННОМ ЦЕНТРЕ ДЛЯ ПРЕДОТВРАЩЕНИЯ НАВОДНЕНИЙ

А.В. Калюжная, С.В. Иванов

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

E-mail: kalyuzhnaya.ann@gmail.com

Обсуждаются вопросы создания ситуационного центра нового поколения для решения сложной мультидисциплинарной задачи предупреждения угрозы и предотвращения наводнений. Эффективность рабочего процесса и повышение качества управленческих решений в задачах подобного типа призваны обеспечить т.н. системы поддержки коллективного принятия решения (ПКПР). Рассматривается реализация ПКПР в системе предупреждения угрозы наводнений в Санкт-Петербурге с использованием технологии облачных вычислений.

Введение

В связи с развитием современных технологий и инструментария для поддержки принятия решений в чрезвычайных ситуациях (ЧС) все более востребованной становится концепция ситуационных центров (СЦ) нового поколения. Особенно это оправданно тогда, когда ЧС носит комплексный характер, относящийся к разным аспектам развития ситуации, которые не могут быть в полной мере охвачены опытом отдельных специалистов – лиц, принимающих решения (ЛПР), а являются сферами интересов группы экспертов и других заинтересованных лиц. Развитие технологий СЦ необходимо, в большей степени, применительно к быстропротекающим ЧС, в которых время принятия решений ограничено, а небольшое промедление может привести к гибели людей. Одним из характерных примеров таких ЧС являются наводненческие ситуации, частота которых на территории РФ в последние два десятилетия увеличилась вследствие изменений климата. Наибольшую опасность и ущерб создают наводнения двух типов: морские нагонные и паводковые. Нагонные наводнения характерны, прежде всего, для Санкт-Петербурга, для северной части Каспийского моря, устьев северных рек. Ливневые наводнения распространены в России практически повсеместно (Дальний Восток, Северный Кавказ, Южная Сибирь и др.).

Основные проблемы, решаемые при создании СЦ нового поколения, применительно к задаче предотвращения угрозы наводнений

Как и многие природные явления, наводнения вызываются совокупностью множества факторов, а их прогнозирование с помощью современных гидродинамических моделей является задачей высокопроизводительных вычислений. Несмотря на то, что на настоящий момент прогнозирование угрозы наводнений является достаточно отлаженным процессом, это не гарантирует успешное решение задачи предотвращения и устранения последствий наводненческой ситуации. Это обусловлено тем, что данный процесс связан с решением обратной мультидисциплинарной задачи – выбора оптимального сценария, определяющего меры по превентивному воздействию на развитие наводнения и мероприятия по минимизации ущерба в условиях неопределенности и неполноты входных данных. Как следствие, для решения такой задачи необходимо создание соответ-

ствующих инструментов ЛПР, позволяющих, используя все доступные источники информации, сократить время и обеспечить поддержку принятия решения, а также эффективно управлять действиями служб по предотвращению и ликвидации последствий ЧС.

Своевременное получение информации при управлении в ЧС позволяет принять более обоснованное, точное решение, что может, в свою очередь, качественно изменить ход событий и позволить предотвратить негативные последствия или сократить их. Однако в настоящее время разрыв между возможностью информационно-коммуникационных технологий предоставлять необходимую информацию и способностью ЛПР ее оперативно анализировать становится все более существенным. Особенно это характерно для уровня руководителя, на котором должны приниматься окончательные решения. Увеличить эффективность управленческих технологий и вывести их на новый уровень позволяет интерпретация СЦ как коллективной советующей системы. Разработка такого рода решения связана с преодолением следующих научно-технических проблем.

– Построение и организация «интеллектуального ядра» СЦ как комплекса средств сбора, агрегации и анализа данных, вычислительных моделей, а также средств интерпретации результатов и поддержки принятия решений. Это направление является приоритетным для развития технологий СЦ в мире [1]. Оно связано с обеспечением ЛПР не только необходимой информацией для самостоятельного принятия решений, но (в первую очередь) – соответствующими рекомендациями, позволяющими не только обосновать выбор того или иного действия, но и оценить связанные с ним риски.
– Разработка технологий, позволяющих эффективно сопрягать вычислительные модели, используемые в составе СЦ, с агрегированными данными из различных источников (например, ассимилируя и подстраивая работу моделей по мере развития ситуаций). Примером может служить концепция Dynamic Data Driven Application Systems, DDDAS, которая находит применение в различных областях научной деятельности: моделировании лесных пожаров, исследовании водных ресурсов, моделировании прибрежной среды и т.п.
– Создание динамической распределенной инфраструктуры для организации вычислительного процесса «интеллектуального ядра» в условиях ЧС. Основная проблема в данном случае состоит в том, что случаи возникновения ЧС достаточно редки, однако сам процесс поддержки принятия решений крайне ресурсоемок; при этом требования по вычислительной нагрузке изменяются нерегулярным образом. Как следствие, это делает неэкономичным использование для работы СЦ выделенных суперкомпьютеров и требует реализации т.н. концепции *экстренных вычислений* (UC, urgent computing) в распределенной среде [2].
– Основная проблема в развитии технологий создания базы вычислительных моделей и поддержки вычислительных и программных ресурсов для «интеллектуального ядра» состоит в том, что даже высококвалифицированные специалисты не владеют в полной мере всей «линейкой» вычислительных моделей (и реализующих их программных продуктов) для решения мультидисциплинарных задач, таких как развитие ЧС, связанных с угрозой наводнений. Это приводит к необходимости создания *коллаборативных подходов*, позволяющих эффективно объединять необходимых специалистов, вычислительные и программные ресурсы.

Поддержка коллективного принятия решений для предотвращения угрозы наводнения в Санкт-Петербурге

В соответствии с современными тенденциями в технологиях СЦ нового поколения, наиболее востребованным и актуальным является направление, связанное с развитием систем поддержки принятия решения (СППР). Особенно перспективными являются исследования в области систем распределенной поддержки коллективного принятия ре-

шений на основе проблемно-ориентированного моделирования. Основное преимущество данного подхода заключается в возможности решать сложные комплексные задачи в случаях, когда аналитические методы недоступны или неэффективны [3]. В ходе решения проблемы процесс принятия решений может включать информационный обмен, обсуждение и согласование мнений между заинтересованными сторонами [4]. В ходе процесса принятия решений группой экспертов, взаимодействующих между собой путем обмена информацией, могут возникать конфликтные ситуации, приводящие к неопределенности в выборе окончательного решения. Для разрешения таких конфликтов и минимизации неопределенности используется концепция поддержки коллективного принятия решения (ПКПР). Реализации концепции ПКПР могут иметь свои особенности в зависимости от предметной области [5].

В данной работе рассматривается реализация подхода поддержки коллективного принятия решений на основе проблемно-ориентированного моделирования для предотвращения угрозы наводнения в Санкт-Петербурге, которое может быть интегрировано с системой предупреждения угрозы наводнений. В качестве технологической платформы для реализации данного подхода выбрана среда для облачных вычислений CLAVIRE.

Схема реализации ПКПР в среде CLAVIRE показана на рис. 1.

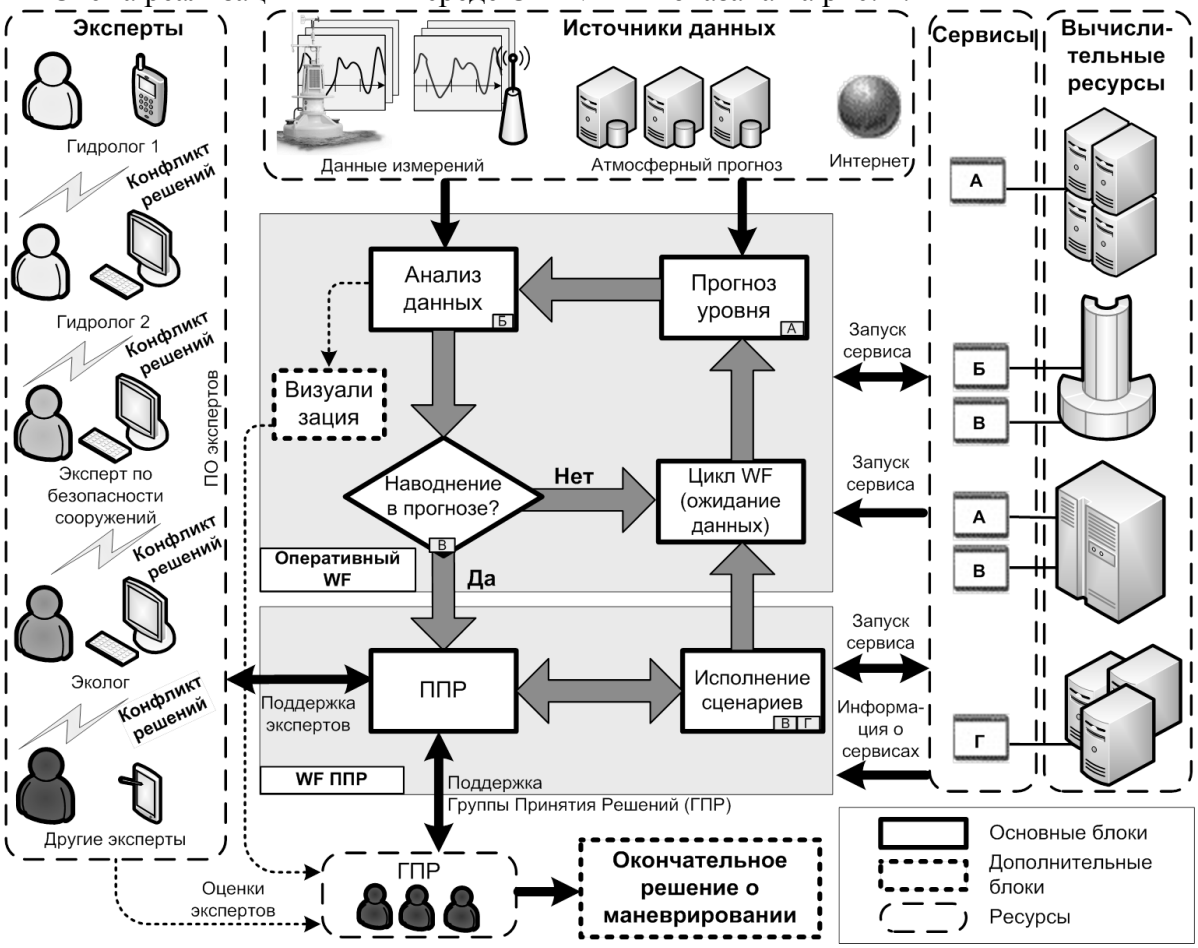


Рис. 1. Процедура коллективного принятия решений для предотвращения угрозы наводнения на базе платформы CLAVIRE

Процедура принятия решений запускается при поступлении внешних данных (атмосферных прогнозов и текущих измерений), которые позволяют оценить степень угрозы наводнения. Прогнозирование уровня воды на основе входных данных включает прогноз морского волнения, усвоение данных текущих измерений и собственно рас-

чет уровня и течений. В основе процедуры ПКПР лежат два потока задач (workflow, далее WF): оперативный WF и WF поддержки принятия решений (ППР).

Оперативный поток задач управляет цепочкой вызовов сервисов, отвечающих за предоставление метеорологического и гидрологического прогнозов, и инициирует вызов процедуры принятия решений. Поток задач поддержки принятия решений помогает экспертам и членам ГПР сделать правильную оценку качества различных планов маневрирования на основе выбранных критериев для принятия окончательного решения или подтвердить сценарий маневрирования, подготовленный в автоматическом режиме.

Заключение

Система распределенной поддержки коллективного принятия решений в ситуационном центре нового поколения для предотвращения угрозы наводнений обеспечивает ЛПР разнообразной информацией: от «сырых» данных до автоматически подготовленных сценариев маневрирования, а также позволяет ЛПР участвовать в процессе определения окончательного сценария маневрирования на всех этапах его выработки. Такая система реализуется на базе инфраструктуры платформы CLAVIRE в рамках концепции облачных вычислений

Литература

1. Новикова Е., Демидов Н. Средства интеллектуального анализа и моделирования сложных процессов как ключевой инструмент ситуационного управления // CONNECT! 2012. №3. – [http://www.ситцентр.рф/pressa/novikova.pdf].
2. Мостаманди С.В., Насонов Д.А., Калюжная А.В., Бухановский А.В. Ансамблевые прогнозы экстремальных гидрометеорологических явлений в распределенной среде CLAVIRE // Изв. вузов. Приборостроение. 2011. № 10. С. 102–104.
3. Karmani F. Simulation-based Optimization and Decision Making with Imperfect Information. Doctoral Thesis, Stockholm, Sweden, 2011. ISBN 978-91-7501-151-6.
4. Wang L., Cheng Q. Web-based Collaborative Decision Support Services: Concept, Challenges and Application // International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences (ISPRS) Technical Commission II Symposium, Vienna, 12-14 July 2006. – [http://www.isprs.org/commission2/proceedings06/pdf/wang.pdf].
5. Zha F.X., Sriram R.D., Fernandez M.G, Mistree F. Knowledge-intensive collaborative decision support for design processes: A hybrid decision support model and agent // Comput. Ind. 59, 9 (December 2008). P. 905–922.

ТЕХНОЛОГИЯ ВЫСОКОУРОВНЕВОЙ РАЗРАБОТКИ КОМПОЗИТНЫХ ПРИЛОЖЕНИЙ В РАМКАХ КОНЦЕПЦИИ ВИРТУАЛЬНЫХ МОДЕЛИРУЮЩИХ ОБЪЕКТОВ

П.А. Смирнов, С.В. Ковальчук

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

smirnp@niuitmo.ru

Рассмотрены существующие проблемы workflow-композиции, предложен метод их решения на основе виртуальных моделирующих объектов с семантическим описанием, а также описана реализация инструментальной среды, воплощающей предложенное решение.

Введение

Концепция потока задач (Workflow, WF [1]) позволяет разрабатывать композитные приложения с применением различных программно-аппаратных ресурсов. Использование распределенных систем (грид, кластер и т.д.) открывает возможности по решению комплексных научных задач, требующих знаний из различных предметных областей, обработки больших объемов данных, а также использования разнообразных методов и средств компьютерного моделирования. При этом платформа распределенной системы в общем случае должна обеспечивать управление процессом исполнения отдельных сервисов с целью обеспечения эффективного использования ресурсов и минимизации общего времени решения задач. Благодаря композиции информационных и вычислительных (как программных, так и аппаратных) ресурсов становится возможным взаимодействие географически удаленных друг от друга групп ученых и, как следствие, построение виртуальных лабораторий для организации междисциплинарных исследований – в соответствии с принципами электронной науки eScience [2]. Одной из очевидных проблем, затрудняющих проведение такого рода исследований, является высокая сложность инструментов композиции, зачастую требующих от ученых знаний и навыков в области информационных технологий, а также данных о корректности применения тех или иных доступных в системе ресурсов.

1. Обзор уровней композиции

На сегодняшний день предложены несколько подходов [3] по автоматизации построения потоков задач (WF), в которые можно условно разделить на три уровня:

1. Композиция из набора доступных сервисов. Первый и самый низкий уровень, на котором пользователю предлагаются программные и аппаратные ресурсы в виде «как есть», т.е. без каких-либо унифицированных и абстрактных описаний. Основная проблема данного уровня – сложность композиции из-за многообразия разнородных ресурсов: отладка взаимодействий между сервисами, порядок вызова, маршруты передачи данных – все эти задачи ложатся на пользователя и требуют от него соответствующих знаний. Для композиции на данном уровне применяются два подхода – последовательный (interleaved) и монолитный (monolithic) [3], различающиеся степенью взаимодействия с сервисами. Последовательный подход обращается к сервисам прямо во время композиции, предлагая к работе только доступные и рабочие варианты. Монолит-

ный подход использует только имеющиеся описания сервисов, полагая, что все сервисы развернуты правильно, находятся в рабочем состоянии и запуск WF априори завершится успешно, а получаемый на данном этапе WF-сценарий называется конкретным (concrete workflow, CWF) и может быть запущен соответствующим (зависит от WF-языка) менеджером задач, например Condor DAGMan [4]. На сегодняшний день существуют инструменты для автоматической композиции на уровне сервисов, использующие базы знаний и технологии искусственного интеллекта (см., например, [5]).

2. Сервисы с абстрактным описанием. Второй уровень композиции дополняет предыдущий наличием некоторого формата описания сервисов и ресурсов, имеющихся в системе. Композиция на данном уровне идет с использованием поэтапного (staged) подхода [3]. Поэтапный подход обычно характеризуется разделением на логический и физический уровни. Логический уровень представляет собой композицию типов, взятых из описаний сервисов. Получаемый на логическом этапе WF-сценарий носит название абстрактного потока задач (abstract WF, AWF). На физическом этапе (как правило, после запуска WF на выполнение) AWF обрабатывается исполняющим (execution) механизмом системы управления workflow, например Pegasus [6], который автоматически распределяет задачи на доступные в системе ресурсы и вызывает конкретные экземпляры сервисов, т.е. происходит трансформация абстрактного WF в конкретный. Дальнейшая работа по исполнению происходит по аналогии с первым уровнем, описанным выше.

3. WF-шаблоны. Третий уровень подразумевает использование WF-шаблонов – заранее составленных и классифицированных частей WF, предлагающих одно или множество решений текущей задачи. Сам по себе шаблонный подход [3] подразумевает использование некоторой заготовленной заранее базы шаблонов, однако на практике шаблоны создаются пользователями либо с нуля, либо путём модификации уже существующих. Недостаток у пользователя знаний о смежных предметных областях вызывает трудности при попытке проведения междисциплинарных исследований, поэтому заблаговременное формирование базы шаблонов на основании экспертных знаний – вполне логичное требование шаблонного подхода. На уровне WF-шаблонов работают такие системы, как CAT/KANAL [6] и WINGS [7], в которых благодаря иерархической организации базы шаблонов предоставляется возможность полуавтоматической композиции.

2. Виртуальные моделирующие объекты

Обзор условно выделенных уровней в подходах к композиции WF указывает на несколько существующих на сегодняшний день проблем. Во-первых, это недостаток у пользователя знаний об особенностях доступных ему программных и вычислительных ресурсов, соответственно непонимание, какой ресурс лучше решает ту или иную задачу. Вторая проблема заключается в трудностях проведения междисциплинарных исследований, осложняющихся в случае отсутствия у пользователя достаточных знаний о соседних предметных областях. Для решения этих проблем нами предлагается *четвертый уровень*, основанный на динамической генерации WF-шаблонов на основе композиции виртуальных объектов. Под виртуальным моделирующим объектом (virtual simulation object, VSO) понимается некоторая совокупность расчетных моделей и характеристик, представляющая виртуальный прототип объекта реального мира. Виртуальный объект имеет определенную внутреннюю структуру, задекларированную в виде семантического описания (метаописания), представляющего собой формализованные знания предметной области. Такое описание делает возможным автоматическую композицию соответствующих объектов и моделей внутри них, что впоследствии определяет порядок взаимодействия и передачу данных между ними. Разработкой образов объектов за-

нимаются эксперты предметной области, а благодаря некоторому унифицированному формату структуры появляется возможность объединить объекты разных разработчиков в единый каталог. Каталог представляет собой распределенную базу знаний, доступную во время работы различным специалистам, что решает проблему проведения мультидисциплинарных исследований. К тому же использование экспертных оценок тех или иных методов решения задач дает возможность оценивания конечного композитного приложения по какому-либо критерию [8].

3. Программная реализация

Практической частью данной работы является разработка решения, воплощающего концепцию виртуальных объектов [9]. Функционально решение состоит из двух компонентов:

Конструктор (VSO construct) – инструментарий для работы конечного пользователя – специалиста, проводящего моделирующие расчетные исследования путём композиции виртуальных объектов из одной или нескольких предметных областей. Графический интерфейс максимально упрощает работу, делая процесс композиции наглядным и не требуя от пользователя никакой технической информации касательно вычислительных ресурсов. Для построения корректных вычислительных схем достаточно иметь лишь необходимые наборы входных данных и базовое понимание предмета. Система автоматически объединяет соответствующие параметры, оставляя пользователю возможность конфигурировать их значения вручную. В результате наполнения среды объектами, указания им свойств, организации взаимодействия друг с другом пользователь запускает построение сценария расчетных задач, а затем производит запуск построенного WF в распределенной исполнительной платформе CLAVIRE [10]. Генерация потока задач производится платформой с использованием специального проблемно-ориентированного языка EasyFlow [11] и представляет собой последовательность шагов и параметров для запуска расчетных пакетов. Процесс выполнения WF и результаты расчетов отражаются в интерфейсе CLAVIRE.

Редактор виртуальных объектов (VSO editor) – ориентирован на использование специалистом-экспертом предметной области, в задачи которого входит разработка, редактирование образов виртуальных объектов, декларирование их свойств, моделей, сценариев исполнения, привязки пакетов, параметров и т.д.,– результатом работы эксперта в редакторе является метаописание, автоматически сгенерированное в формате OWL и пригодное для чтения в любых известных OWL-редакторах (напр. Protege). Графический интерфейс редактора позволяет в полной мере создавать, редактировать, сохранять, скачивать, загружать файлы в формате OWL, а также работать с внутренней структурой объектов, а именно: задавать сценарии, варианты реализации, добавлять пакеты, привязывать переменные пакетов к значениям метапараметров и т.д. В конечном итоге, описания объектов попадают в каталог, который используется в работе конструктора. Взаимодействие компонентов инструментария VSO с системами платформы CLAVIRE представлено на рис. 1.

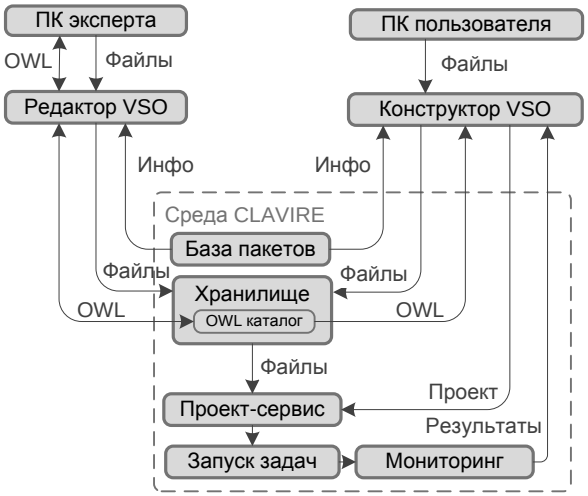


Рис. 1. Взаимодействие компонентов VSO с системами платформы CLAVIRE

4. Сценарий использования

Работа пользователя по композиции объектов происходит на самом верхнем – четвертом уровне. В конструкторе пользователю предлагается каталог объектов для выбора и добавления их в рабочую область. В рабочей области объекты представлены в виде графических элементов с некоторым набором свойств и внутренней структурой, содержащей наборы расчетных моделей, а также варианты их реализации (сценарии). Для каждой используемой модели пользователь должен выбрать сценарий и реализацию, после чего он получит доступ к параметрам реализующих её пакетов – необходимыми переменными, представленными отдельными графическими элементами. Эквивалентные параметры автоматически объединятся с параметрами других моделей, наглядно демонстрируя маршруты пересылки выходных данных одних моделей/объектов на вход другим. После построения требуемой конфигурации объектов пользователь запускает генерацию WF-сценария. Модель каждого объекта вычисляется рядом моделирующих операций (шагов), содержащих информацию о запускаемом пакете и требуемых входных параметрах. Работа автоматической системы (а не пользователя) по динамической генерации WF-шаблонов происходит по аналогии с третьим уровнем композиции, представленным выше. Следующий этап – запуск сгенерированного WF в распределенной среде (кластер, грид и т.д.) – происходит в интерфейсе CLAVIRE.

Заключение

Обозначенная концепция виртуальных объектов предлагает новый уровень разработки композитных приложений в распределенных средах. Переход на данный уровень позволит решить проблемы проведения междисциплинарных исследований, а также обеспечит пользователей дополнительной информацией касательно выбора оптимальных решений для поставленных задач. Разработанный базовый инструментарий позволяет оценить удобство композиции виртуальных объектов благодаря наглядному графическому интерфейсу и некоторым автоматическим функциям. В перспективе – внедрение критериев экспертных оценок эффективности тех или иных решений и реализация механизма автоматического выбора соответствующих требованиям вариантов.

Литература

1. Gil Y. From Data to Knowledge to Discoveries: Scientific Workflows and Artificial Intelligence // Scientific Programming. 2008. Vol. 17, Issue 3. P. 1–25.
2. Hey T., Tansley S., Tolle K. The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmond Washington, 2009.
3. Agarwal V. et al. Understanding Approaches for Web Service Composition and Execution // IBM Research Report. 2007. – [http://domino.watson.ibm.com/library/cyberdig.nsf/papers/9E61D97ED1DDA2A28525733D00444D0D/$File/RI07005.pdf].
4. Directed Acyclic Graph Manager – [http://research.cs.wisc.edu/condor/dagman/].
5. Ozorhan E.K., Kuban E.K., Cicekli N.K. Automated composition of web services with the abductive event calculus // Information Sciences. 2010. №180. P. 3589-3613.
6. Kim J., Gil Y., Spraragen M. Principles For Interactive Acquisition And Validation Of Workflows // Journal of Experimental & Theoretical Artificial Intelligence. 2010. Vol. 22. P. 103-134.
7. Gil Y., Ratnakar V., Deelman E. Wings: Intelligent Workflow-Based Design of Computational Experiments // Intelligent Systems, IEEE. January 2010.
8. Ковальчук С.В., Маслов В.Г. Интеллектуальная поддержка процесса конструирования композитных приложений в распределенных проблемно-ориентированных средах // Известия вузов. Приборостроение. 2011. №10. С. 29–35.

9. Kovalchuk S.V., Smirnov P.A., Kosukhin S.S., Boukhanovsky A.V. Virtual Simulation Objects Concept as a Framework for System-Level Simulation // Proceedings of IEEE e-Sceince Conference. 2012. CD-ROM. ISBN 978-1-4673-4465-4.
10. Бухановский А.В., Васильев В.Н., Виноградов В.Ю., Смирнов Д.А., Сухоруков С. С., Яппаров Т.Г. CLAVIRE: перспективная технология облачных вычислений второго поколения // Изв. вузов. Приборостроение. 2011. Т. 54, № 10. С. 7–14.
11. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Изв. вузов. Приборостроение. 2011. Т. 54, № 10. С. 40–49.

ОЦЕНКА ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРИКЛАДНЫХ ПАКЕТОВ В ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ СРЕДАХ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ ВТОРОГО ПОКОЛЕНИЯ

Т.Н. Чуров

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

E-mail: tchurovtim@gmail.com

Введение

Важной характеристикой высокопроизводительных вычислений в любом направлении научной деятельности является оценка времени выполнения этих вычислений. Более того, в таких случаях, как вычисления в ситуационных центрах, именно оценка времени становится определяющим фактором эффективности и целесообразности самих вычислений. Таким образом, подсистемы расчета оценки времени вычислений являются необходимыми атрибутами предметно-ориентированных сред облачных вычислений. Кроме обязательной функциональности таким системам предъявляются требования автоматической корректировки используемых методов на основе собираемых статистических данных.

Задачи системы

Основными задачами системы оценки времени выполнения прикладных пакетов являются следующие оценки:

- Оценка прогнозного времени выполнения отдельного пакета на конкретном ресурсе на основе параметрических моделей.
- Оценка на основе параметрических моделей объемов требуемых для исполнения пакета входных файлов и файлов, генерируемых пакетом во время выполнения.
- Оценка накладных расходов при выполнении пакетов на конкретных ресурсах, с учетом времени передачи файлов к ресурсу, на основе статистических данных.
- Оценка пользовательского прогнозного времени выполнения с учетом загруженности системы и времени запуска пакета.

Автоматизация процесса корректировки методов оценки, в свою очередь, предъявляет к системе следующие требования:

- Сбор статистики по времени выполнения отдельного пакета и по оценкам его выполнения.
- Сбор статистики по накладным расходам как общесистемным, так и для конкретных ресурсов.
- Автоматическая или полуавтоматическая корректировка коэффициентов моделей по набранным статистическим данным.

Система оценки времени выполнения в платформе CLAVIRE

Платформа CLAVIRE (CLoud Applications VIRtual Environment, [1]) предназначена для эффективного управления вычислительными и программными ресурсами неоднородных распределенных вычислительных инфраструктур в рамках модели облачных вычислений. Она обеспечивает предоставление доступа к прикладным программным

пакетам, возможность конструирования на основе этих пакетов сложных композитных приложений и запуска их на доступных вычислительных ресурсах. В связи со спецификой, равно как и для удобства использования, платформе потребовалась разработка развитой системы оценки времени выполнения прикладных пакетов.

Параметрическая модель оценки времени выполнения пакета интегрирована с его общим описанием и реализована в компоненте «База пакетов» (PackageBase, [2]). Значения коэффициентов, которые применяются к модели для вычисления, содержатся в описаниях ресурсов и, соответственно, в компоненте «База ресурсов» (ResourceBase, [3]). При запуске вычисления компонент исполнения применяет коэффициенты конкретных ресурсов к модели запускаемого пакета и передает их дальше, соответственно, компонентам планирования и взаимодействия с пользователем. Ядром системы оценки времени выполнения является сервис, разработанный на основе методов компонента платформенной статистики (Provenance). На основе общей статистики этот сервис вычисляет отклонения ожидаемых времен выполнения пакета от реальных и корректирует коэффициенты моделей для соответствующих ресурсов с использованием открытых методов компонента «База ресурсов».

Заключение

Таким образом, предложенный подход позволяет решить проблему предварительного биллинга облачных сервисов и композитных приложений в рамках моделей SaaS и AaaS для платформы CLAVIRE. Пользователю на этапе подготовки к запуску композитного приложения представляются предварительные оценки времени выполнения и стоимости расчетов с учетом неопределенности загрузки ресурсов среды и рисков превышения заданного лимита. При этом пользователь сам может выбрать наиболее приемлемый для него сценарий исполнения (что определяет способ планирования и используемый тариф).

Литература

1. Бухановский А.В., Васильев В.Н., Виноградов В.Н., Смирнов Д.Ю., Сухоруков С.А., Яппаров Т.Г. CLAVIRE: перспективная технология облачных вычислений второго поколения // Известия высших учебных заведений. Приборостроение. 2011. №10. С. 7–13.
2. Князьков К.В., Ларченко А.В. Предметно-ориентированные технологии разработки приложений в распределенных средах // Известия высших учебных заведений. Приборостроение. 2011. №10. С. 36–43.
3. Марьин С.В., Ковальчук С.В. Сервисно-ориентированная платформа исполнения композитных приложений в распределенной среде // Известия вузов. Приборостроение. 2011. №10. С. 21–28.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ АВТОРОВ

Агапова Т.А. 342
Аксёнов С.В. 4
Андреев А.М. 236
Андрюшин Д.В. 388
Артамонов Ю.С. 82
Афраймович Л.Г. 8
Ахматнуров Д.В. 353
Ашарина И.В. 12
Балахонцева М.А. 464
Баранов Д.А. 18
Баркалов К.А. 24, 29
Бартенев Ю.Г. 34
Бастраков С.И. 37
Безгодов А.А. 464
Белозерова Т.С. 122
Беляков В.А. 43
Биллиг В.А. 48, 52, 247
Билятдинова А.З. 464
Богачева А.В. 460
Болгова Е.В. 464
Болдырев Ю.Я. 455
Болодурина И.П. 54
Борисейко П.П. 61
Бородулин К.В. 63
Бутымова Л.Н. 287
Быков А.Н. 424
Быстриков Д.В. 66
Валиуллин А.Ф. 71
Вильдеманов А.В. 75
Волкович А.Н. 77
Волченская Т.В. 194
Востокин С.В. 82
Гавриленко В.В. 83
Гаврилов Н.И. 87
Галкин А.А. 83
Гергель В.П. 416
Гибадуллин Р.Ф. 91
Голубев А.С. 96
Голубев В.И. 100
Гоносков А.А. 37
Гордлеева С.Ю. 377
Городецкий С.Ю. 101
Горшков А.В. 106
Гришагин В.А. 111
Гурбатов С.Н. 116
Данилов И.Г. 117
Деменев А.Г. 122
Демин И.Ю. 116
Демичев А.П. 125, 228
Дерябин М.А. 129
Дикарев К.И. 8
Диков Д.А. 134
Дикусар В.В. 139
Дмитриев Н.А. 342
Добряев Д.Н. 111
Донченко Р.В. 37
Дружков П.Н. 143
Духанов А.В. 460, 464
Ефименко Е.С. 37
Жирнова Т.А. 148
Жунин А.Ю. 149
Зайцев А.А. 129
Залялов Д.Г. 156
Замотин К.Ю. 455
Заручевская Г.В. 153
Захаров Р.К. 160
Зиновьев И.И. 96
Золотых Н.Ю. 29
Зорин Д.Ю. 164
Зубко И.Ю. 225
Иванов С.В. 477
Иванченко М.В. 75, 164
Игрунов В.И. 342
Идрисов Р.И. 168
Ильин В.А. 125
Исмагилова А.С. 393
Исупов К.С. 190
Казанцев В.Б. 377
Калачев А.В. 170
Калюжная А.В. 477
Карбовский В.А. 470
Карпов А.П. 34
Карсаков А.С. 170
Катеров А.С. 176

Квасов Д.Е. 365
Кириллин М.Ю. 106
Кириллов А.С. 180
Кислицын Д.И. 185, 427
Князьков В.С. 190, 194
Князьков К.В. 472
Ковальчук С.В. 472, 481
Козинов Е.А. 205
Копысов С.П. 212, 217
Корзаков Ю.Н. 342
Коротченко А.Г. 210
Коршунова А.Л. 106
Костенецкий П.С. 223
Кочуров В.И. 225
Крылов И.Б. 43
Крюков А.П. 125, 228
Крючков И.А. 34, 342
Кузнецов К.В. 231
Кузьмин И.М. 212, 217
Кузьминский М.Б. 236
Кулакович У.С. 335
Курсиков А.О. 238
Кустикова В.Д. 241
Лайком Д.Н. 4
Лапин А.В. 156
Лаптева Т.В. 75
Лашманов В.Н. 342
Лепшова Е.С. 247
Линева Т.Н. 252
Лобанов А.В. 12
Логвин Ю.В. 342
Логинов А.В. 190
Ломтев А.Г. 342
Макаров А.А. 8
Макарова Т.Д. 252
Малова А.Ю. 257
Малышев А.С. 37
Мееров И.Б. 29, 37, 170
Милюкин Ю.А. 265, 270
Минязев Р.Ш. 274
Миронов В.И. 377
Михелев В.М. 231, 277
Модорский В.Я. 282, 287
Мухина И.В. 377
Назаров Ю.П. 82
Наместников Г.И. 8
Напыльникова Я.А. 170
Недожогин Н.С. 212
Никитина Н.Н. 289
Новиков А.К. 212
Новиков А.М. 292
Новокрещенов А.А. 295
Обшта А.Ф. 83
Овсюхно А.Ю. 170
Одякова Д.С. 300
Оленёв Н.Н. 139, 305, 308
Осинин И.П. 194
Осокин Д.В. 450
Павлов Д.А. 313
Панков С.В. 315
Панкратова М.А. 397
Парфёнов Д.В. 8
Парфёнов Д.И. 54
Парфенова Л.В. 421
Пекунов В.В. 320
Перекатова В.В. 399
Перетрухин С.Ф. 8
Петрик А.Н. 342
Петров Д.В. 231, 277
Петров И.Б. 100
Петухов Е.П. 455
Пищулин И.А. 34
Подчукаев В.А. 270
Поздняк Е.И. 370
Полежаев П.Н. 325
Половинкин А.Н. 29, 330
Поляков С.П. 125
Прилуцкий М.Х. 8, 335
Приходько Н.В. 228
Прокин И.С. 377
Прокошев В.В. 440
Прончатов-Рубцов Н.В. 116
Пыстогов С.В. 91
Радченко Г.И. 134, 430
Решетников А.Н. 337
Румянцев А.С. 289
Рыбаков М.А. 342
Рыбкин А.С. 148, 342
Рычков В.Н. 217
Рябов В.В. 308
Савельев А.Г. 343
Самохвалов С.В. 345
Сапрыкин В.А. 347
Сатанин А.М. 353
Свиридов А.П. 356
Свистунов С.С. 360
Селякин А.Г. 342
Семенов А.И. 223
Семенов Г.П. 342
Сергеев Я.Д. 365

Сидоренко К.В. 43
Сидоров И.А. 370
Сидоров С.В. 29, 372
Сизов Е.А. 424
Симонов А.Ю. 377
Смирнов П.А. 481
Смирнова А.И. 379
Сморякова В.М. 210
Соболев С.И. 383
Соколов А.А. 388
Спивак С.И. 393
Старостин Н.В. 8, 397
Стрюков В.Н. 342
Субочев П.В. 399
Суворов С.В. 270
Супрун А.Н. 401
Сурмин И.А. 37
Тарасов В.Л. 406
Тарасов Г.В. 300
Титов В.В. 411
Тонков Л.Е. 217
Торопчин Д.А. 231
Турлапов В.Е. 87, 416
Турчин И.В. 399
Устюгов И.И. 421
Ушаков А.Ю. 342
Ушаков Ю.А. 325
Федоров А.А. 424
Федосеев Д.В. 353
Филимонов А.В. 8
Халилов Л.М. 421
Харебов П.В. 122
Харитонов Д.И. 300
Хеннер В.К. 122
Хеннер Е.К. 122
Хохлов Н.И. 100
Хромых В.Е. 427
Худякова Е.С. 430
Чернецов А.М. 236
Черноскутов М.А. 435
Чуров Т.Н. 486
Шамардин Л.В. 228
Шамин П.Ю. 440
Шамов Е.А. 445
Шатохин А.В. 342
Швецов А.В. 450
Шлыков С.Н. 342
Шувар Б.А. 83
Шухман А.Е. 325
Ярулин Д.Р. 342

СОДЕРЖАНИЕ

Аксёнов С.В., Лайком Д.Н. Кластеризация показателей энергетической активности звезд на базе алгоритмов Густафсона-Кесселя и DBSCAN с помощью суперкомпьютерного кластера 4

АфраймовичЛ.Г., ДикаревК.И., МакаровА.А., НаместниковГ.И., ПарфёновД.В., ПеретрухинС.Ф., ПрилуцкийМ.Х., СтаростинН.В., ФилимоновА.В. Программные средства для определения оптимальных режимов газотранспортной системы на стадии ее проектирования или модернизации с применением технологий многопоточного программирования 8

АшаринаИ.В., ЛобановА.В. Метод разбиения многомашинной системы на комплексы при организации сбое- и отказоустойчивых параллельных вычислений 12

Баранов Д.А. Анализ применимости OpenCL-реализаций методов Гаусса, Холецкого и Зейделя для решения линейной задачи наименьших квадратов 18

Баркалов К.А. Использование значений производных в параллельных алгоритмах условной глобальной оптимизации 24

Баркалов К.А., Мееров И.Б., Половинкин А.Н., Сидоров С.В., Золотых Н.Ю. Оптимизация параметров SVM-регрессии с использованием алгоритма глобального поиска 29

Бартенев Ю.Г., Карпов А.П., Крючков И.А., Пищулин И.А. Адаптация некоторых решателей СЛАУ для арифметических ускорителей NVIDIA 34

БастраковС.И., ГоносковА.А., ДонченкоР.В., ЕфименкоЕ.С., МалышевА.С., МееровИ.Б., СурминИ.А. Анализ производительности и масштабируемости кода Picador для моделирования плазмы 37

БеляковВ.А., СидоренкоК.В., КрыловИ.Б. Использование графических ускорителей при моделировании релаксации массива кремниевых нанокристаллов по методу Монте-Карло 43

Биллиг В.А. Оценки времени в модели параллельного выполнения программ 48

Биллиг В.А. «Параллельные вычисления и многопоточное программирование» – учебный курс и учебное пособие 52

Болодурина И.П., Парфёнов Д.И. Моделирование распределения ресурсов и динамической балансировки нагрузки в системе дистанционного обучения 54

Борисейко П.П. Параллельные расчеты сложных задач механики сплошных сред на базе программно-вычислительного комплекса Star-CCM+ 61

Бородулин К.В. Разработка технологии хранения данных для параллельных СХД 63

Быстриков Д.В. Применение многосеточных методов для решения эллиптических задач с помощью пакета OpenFOAM 66

Валиуллин А.Ф. Использование генетических алгоритмов с критерием Джефриса-Мацуситы для выбора оптимального набора входных каналов в задаче дешифрации наземных объектов по данным ДЗЗ 71

ВильдемановА.В., ИванченкоМ.В., ЛаптеваТ.В. Моделирование динамики взаимодействующих квантовых частиц в пространственно-неоднородных решетках 75

Волкович А.Н. Параллельная реализация многокритериального метода построения карт диспаратности 77

Востокин С.В., Артамонов Ю.С., Назаров Ю.П. Templet-сервис непрерывной интеграции для разработки высокопроизводительных приложений 82

ГавриленкоВ.В., ОбштаА.Ф., ШуварБ.А., ГалкинА.А. Приближенное решение нелинейных операторных уравнений с помощью итеративного агрегирования 83
Гаврилов Н.И., Турлапов В.Е. Качественная и высокопроизводительная медицинская визуализация методом volume ray casting..... 87
Гибадуллин Р.Ф., Пыстогов С.В. Параллельная система управления полнообъектными защищенными базами данных картографических сцен..... 91
Голубев А.С., Зиновьев И.И. Детектирование лиц людей на изображении с использованием графических ускорителей..... 96
Голубев В.И., Петров И.Б., Хохлов Н.И. Разработка параллельного программного комплекса для оценки сейсмостойкости наземных сооружений путём прямого численного моделирования 100
Городецкий С.Ю. Обобщения метода Direct на задачи с функциональными ограничениями 101
ГоршковА.В., КириллинМ.Ю., КоршуноваА.Л. Компьютерное моделирование оптической диффузионной спектроскопии для функциональной диагностики мозга человека..... 106
Гришагин В.А., Добряев Д.Н. Использование параллельных алгоритмов глобального поиска с обобщённой квадратичной характеристикой для решения прикладных задач гидродинамики..... 111
Гурбатов С.Н., Демин И.Ю., Прончатов-Рубцов Н.В. Численное моделирование эволюции акустических волн в медико-биологических средах..... 116
Данилов И.Г. Об актуальности исследования и развития современных моделей распределенного и параллельного программирования..... 117
Деменев А.Г., Хеннер Е.К., Белозерова Т.С., Хеннер В.К., Харебов П.В. О решении задач магнитодинамики и когерентных процессов в наномагнитных структурах на суперкомпьютере 122
ДемичевА.П., ИльинВ.А., КрюковА.П., ПоляковС.П. Сравнительный анализ алгоритмов построения больших коммуникационных сетей со свойствами “малого мира” 125
Дерябин М.А., Зайцев А.А. Использование модулярной арифметики для ускорения выполнения операций над числами большой разрядности 129
Диков Д.А., Радченко Г.И. Интеграция сервиса интерактивной визуализации для распределенных виртуальных испытательных стендов с системой CAEBeans 134
Дикусар В.В., Оленёв Н.Н. Применение параллельных алгоритмов в решении краевой задачи на примере входа аппарата в атмосферу 139
Дружков П.Н. Программная реализация параллельного алгоритма градиентного бустинга деревьев решений 143
Жирнова Т.А., Рыбкин А.С. Способы векторизации вычислений для систем с арифметическими ускорителями..... 148
Жунин А.Ю. Вычислительная реализация обобщенного метода Direct для задачи с функциональными ограничениями 149
Заручевская Г.В. Реализация решения разностной схемы расщепления двумерного уравнения теплопроводности в клеточной машине 153
Залялов Д.Г., Лапин А.В. Численное решение задачи оптимального управления системой, описываемой линейным эллиптическим уравнением, при наличии нелокальных ограничений на состояние системы..... 156
Захаров Р.К. Распознавание изображений с применением модифицированного алгоритма, основанного на дескрипторе контекста формы 160
Зорин Д.Ю., Иванченко М.В. Моделирование динамики иммунного ответа в ансамблях Т-клеток с применением графических процессоров 164

Идрисов Р.И. Облачная среда функционального программирования для научных вычислений и образования 168
Калачев А.В., Карсаков А.С., Мееров И.Б., Напыльникова Я.А., Овсюхно А.Ю. Разработка программного комплекса для анализа энергоэффективности программного обеспечения 170
Катеров А.С. Высокопроизводительные вычисления при исследовании сводимости трех- и четырехиндексных задач 176
Кириллов А.С. Анализ масштабируемости параллельных алгоритмов решения кооперативной игры 180
Кислицын Д.И. Автоматизированная система управления расчетом сложных строительных объектов на базе метода разделения объекта на проектные единицы 185
Князьков В.С., Исупов К.С., Логинов А.В. Организация позиционно-модулярных вычислений для высокоточных численных расчетов 190
Князьков В.С., Осинин И.П., Волченская Т.В. Организация арифметического разрядно-параллельного СБИС-процессора для массовых вычислений 194
Коваль Е.А., Коваль О.А. Алгоритм поиска совпадающих подстрок двух геномов и его распараллеливание 199
Козинов Е.А. Об одном подходе к представлению данных в системе «Визограф» 205
Коротченко А.Г., Сморякова В.М. Алгоритм поиска наибольшего значения в классе функций, определяемом кусочно-линейной мажорантой 210
Копысов С.П., Кузьмин И.М., Недожогин Н.С., Новиков А.К. Параллельный алгоритм формирования и решения дополнения Шура 212
Копысов С.П., Кузьмин И.М., Рычков В.Н., Тонков Л.Е. Параллельные алгоритмы и технологии решения сопряженных задач FSI 217
Костенецкий П.С., Семенов А.И. Исследование производительности вычислительных систем с многоядерными ускорителями в рамках приложений баз данных 223
Кочуров В.И., Зубко И.Ю. Применение графических ускорителей для определения зависимости потенциальной энергии и равновесного межатомного расстояния от температуры в ГЦК-металлах 225
КрюковА.П., ШамардинЛ.В., ПриходькоН.В., ДемичевА.П. Унифицированный удаленный доступ к суперкомпьютерным ресурсам 228
Кузнецов К.В., Михелев В.М., Торопчин Д.А., Петров Д.В. Суперкомпьютерные технологии в задаче поиска изображений по визуальному подобию 231
КузьминскийМ.Б., АндреевА.М., ЧернецовА.М. Применение GPU для задач квантовой химии: прямой расчет матриц плотности сверхбольших молекул методом PDM 236
Курсиков А.О. Проект практикума по курсу высокопроизводительных вычислений, основанный на CUDA 238
Кустикова В.Д. Об одном подходе к решению задачи видеодетектирования транспортных средств 241
Лепшова Е.С., Биллиг В.А. Реализация и распараллеливание алгоритма интеллектуального анализа данных, основанного на деревьях решений 247
Линева Т.Н., Макарова Т.Д. Разработка сайта и модуля проверки тестов системы WCTEST дистанционного тестирования задач по программированию 252
Малова А.Ю. Обзор методов переупорядочения симметричных разреженных матриц 257
Милюкин Ю.А. Автоматизация построения вычислительного ядра для спецвычислителей на базе FPGA 265

Милюкин Ю.А., Суворов С.В., Подчукаев В.А. Динамический хаос как бесконечный разрыв второго рода в координатах центра смещения 270
Минязев Р.Ш. Архитектура параллельной СУБД на платформе GPU-кластера 274
Михелев В.М., Петров Д.В. Применение высокопроизводительных вычислительных систем для поиска граничных контуров карьеров рудных месторождений 277
Модорский В.Я. Численное решение и экспериментальное исследование междисциплинарных задач 282
Модорский В.Я., Бутымова Л.Н. Разработка расчетно-экспериментального комплекса по исследованию аэроупругих колебаний в энергетических установках 287
Никитина Н.Н., Румянцев А.С. Задача оптимизации времени выполнения приложения в Desktop Grid 289
Новиков А.М. Облако данных для активного хранилища многомерных массивов 292
Новокрещенов А.А. Алгоритм поиска параллельности с синхронизациями во вложенностях циклов 295
Одякова Д.С., Тарасов Г.В., Харитонов Д.И. Система WBS как расширенный инструмент управления вычислительной средой 300
Оленёв Н.Н. Идентификация моделей экономики со структурными изменениями 305
ОленёвН.Н., РябовВ.В. Глобальный поиск в задачах сверхвысокой размерности на примере идентификации математической модели экономики Нижегородской области 308
Павлов Д.А. Модуль исполнения запросов для базы данных, размещенной в памяти GPU 313
Панков С.В. Логико-математическое моделирование рекуррентных нейронных сетей 315
Пекунов В.В. Параллельное решение задачи классификации лиц методом голосования с нейросетевым арбитром 320
Полежаев П.Н., Ушаков Ю.А., Шухман А.Е. Планирование задач для облачных вычислительных центров обработки данных 325
Половинкин А.Н. Методы решения задачи классификации изображений с большим числом категорий 330
Прилуцкий М.Х., Кулакович У.С. Распределение ресурсов в сетевых структурах с активными элементами с использованием параллельной вычислительной среды 335
Решетников А.Н. Об одном из алгоритмов поиска объектов по биомедицинским данным 337
Рыбкин А.С., Агапова Т.А., Дмитриев Н.А., Игрунов В.И., Корзаков Ю.Н., Крючков И.А., Лашманов В.Н., Логвин Ю.В., Ломтев А.Г., Петрик А.Н., Рыбаков М.А., Селякин А.Г., Семенов Г.П., Стрюков В.Н., Ушаков А.Ю., Шатохин А.В., Шлыков С.Н., Ярулин Д.Р. Компактные супер-ЭВМ производства ФГУП «РФЯЦ-ВНИИЭФ». Итоги, инновации, перспективы 342
Савельев А.Г. Ассоциативное сокрытие данных с применением графических процессоров 343
Самохвалов С.В. Реализация технологий параллельного программирования в учебном процессе 345
Сапрыкин В.А. Архитектура телемедицинской системы с использованием гетерогенного вычислительного кластера в качестве серверной части 347
Сатанин А.М., Федосеев Д.В., Ахматнуров Д.В. Моделирование роста наноструктур методом молекулярной динамики с применением технологии CUDA 353
Свиридов А.П. Использование расширяемых языков для программирования многопоточных приложений 356

Свистунов С.С. Интерактивный рендеринг на GPU: вычисление функции видимости 360
СергеевЯ.Д., КвасовД.Е. Вычисления высокой точности на компьютере нового типа 365
Сидоров И.А., Поздняк Е.И. Разработка методов и языковых средств сборочного программирования для распределенных вычислительных сред 370
Сидоров С.В. Исследование эффективности параллельного двухуровневого индексного алгоритма глобальной оптимизации 372
СимоновА.Ю., МироновВ.И., ПрокинИ.С., ГордлееваС.Ю.,. И.В Му3хина, КазанцевВ.Б. Сигнальные процессы в нейронных сетях мозга. Виртуальное моделирование для задач диагностики патологий и тестирования лекарств 377
Смирнова А.И. Применение параллельных вычислений в выделении целевых регионов на мультиспектральных космических снимках 379
Соболев С.И. Исследование эффективности распределенных расчетов в суперкомпьютерных средах 383
Соколов А.А., Андрюшин Д.В. Сравнительное тестирование вычислительных узлов двух суперкомпьютеров 388
СпивакС.И., ИсмагиловаА.С. Распараллеливание схем химических реакций при решении обратных задач 393
Старостин Н.В., Панкратова М.А. Отображение графа параллельной программы на граф вычислительной сети 397
Субочев П.В., Перекатова В.В., Турчин И.В. Вычислительные задачи и перспективы 3D-реконструкции в оптоакустике 399
Супрун А.Н. Метод массового распараллеливания вычислений при решении краевых задач математической физики 401
Тарасов В.Л. Обмен данными в параллельной программе расчета собственных частот оболочек вращения 406
Титов В.В. Объемная визуализация результатов газодинамического моделирования методом проецируемых тетраэдров с использованием технологии CUDA 411
Турлапов В.Е., Гергель В.П. Макет телемедицинской системы нового поколения 416
Устюгов И.И., Парфенова Л.В., Халилов Л.М. Квантовохимические вычисления с использованием программно-аппаратной архитектуры CUDA 421
Федоров А.А., Быков А.Н., Сизов Е.А. Метод распараллеливания прогонки на гибридных ЭВМ 424
Хромых В.Е., Кислицын Д.И. Укрупненная схема построения распределенных вычислений при расчетах сложных строительных конструкций методом разделения на проектные единицы 427
Худякова Е.С., Радченко Г.И. Веб-система разработки и внедрения распределенных виртуальных испытательных стендов 430
Черноскутов М.А. Анализ эффективности использования GPU в задаче поиска в ширину на графе 435
Шамин П.Ю., Прокошев В.В. Модернизация параллельного сетевого симулятора ПСС 440
Шамов Е.А. Подсистемы управления и контроля работоспособности решаемой задачи и аппаратуры гетерогенного вычислительного кластера 445
Швецов А.В., Осокин Д.В. Использование графических ускорителей при расчете оптических свойств полупроводниковых нанокристаллов в рамках теории функционала плотности 450
Болдырев Ю.Я., Замотин К.Ю., Петухов Е.П. Суперкомпьютеры и вопросы фундаментальных основ инженерного образования 455

Богачева А.В., Духанов А.В. Создание виртуальных лабораторных практикумов с применением облачных технологий второго поколения 460
Балахонцева М.А., Безгодов А.А., Билятдинова А.З., Болгова Е.В., Духанов А.В. Инфраструктура стажировки и повышения квалификации молодых специалистов и учёных в области научной компьютерной графики и виртуальной реальности на базе многопрофильной инструментально-технологической платформы CLAVIRE 464
Карбовский В.А. Система активного мониторинга пакетов и ресурсов платформы облачных вычислений 470
Князьков К.В., Ковальчук С.В. Особенности разработки интерактивных композитных приложений 472
Калюжная А.В., Иванов С.В. Распределенная поддержка коллективного принятия решений в ситуационном центре для предотвращения наводнений 477
Смирнов П.А., Ковальчук С.В. Технология высокоуровневой разработки композитных приложений в рамках концепции виртуальных моделирующих объектов 481
Чуров Т.Н. Оценка времени выполнения прикладных пакетов в предметно-ориентированных средах облачных вычислений второго поколения 486

Алфавитный указатель авторов 488

Ф О Р У М
«СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ
В ОБРАЗОВАНИИ, НАУКЕ И ПРОМЫШЛЕННОСТИ»

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ

Материалы XII Всероссийской конференции
(Нижний Новгород, 26–28 ноября 2012 г.)

Под ред. проф. В.П. Гергеля

Отв. за выпуск К.А. Баркалов

Формат 60×84 1/8. Уч.-изд. л. 47,6. Тир. 150 экз.
Издательство Нижегородского госуниверситета им. Н.И. Лобачевского.
603950, Н. Новгород, пр. Гагарина, 23.