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 XIII ВСЕРОССИЙСКОЙ КОНФЕРЕНЦИИ

Нижний Новгород, 14–16 ноября 2013 г.

Нижний Новгород
Издательство Нижегородского госуниверситета
2013

УДК 681.3.012:51
ББК 32.973.26−018.2:22
В93

В93 **Высокопроизводительные параллельные вычисления на кластерных системах.** Материалы XIII Всероссийской конференции (Н. Новгород, 14–16 ноября 2013 г.) / Под ред. проф. В.П. Гергеля. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2013. – 312 с.

Отв. за выпуск К.А. Баркалов

ISBN 978-5-91326-304-9

Сборник материалов Тринадцатой Всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах», состоявшейся 14−16 ноября 2013 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 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 978-5-91326-304-9 ББК 32.973.26−018.2:22

Поддержка конференции

Российский фонд фундаментальных исследований

Компания Intel

Компания NVIDIA

Информационно-аналитический центр Parallel.Ru

Газета научного сообщества «Поиск»

ПРЕДИСЛОВИЕ

14–16 ноября 2013 г. Суперкомпьютерным консорциумом университетов России на базе Нижегородского государственного университета им. Н.И. Лобачевского при поддержке Российского фонда фундаментальных исследований, компаний Intel и NVIDIA проведена XIII 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 темы нашли отражение в настоящем сборнике тезисов докладов, представленных на конференцию (публикуются частично в авторской редакции).

ПРОБЛЕМА ОЦЕНКИ НАДЕЖНОСТИ ПРОГРАММНЫХ КОМПЛЕКСОВ

Н.В. Аболмазова, А.Н. Коварцев

Самарский государственный аэрокосмический университет им. С.П. Королёва

Развернута проблематика оценивания надежности программных комплексов, а также выделены отличия от решения аналогичной задачи для технических устройств. Задано направление для дальнейшего решения поставленной задачи. Определено влияние вероятностных мер на изменение в оценке надежности модулей для одномерного случая.

Введение

Надежность является важным и естественным требованием, предъявляемым к качеству современных аппаратно-программных комплексов (АПК). В области обеспечения надежности технических устройств (ТУ) достигнут высокий уровень. Например, в работе [1] показано, что вероятность отказов вычислителей пилотажно-навигационного комплекса самолетов типа Ту-324 составляет 10^{-10}, что соответствует современным требованиям к аппаратуре космических аппаратов и авиации. В области оценки надежности программного обеспечения (ПО) для АПК положение дел обстоит не столь хорошо.

Известно, что качество программного продукта (ПП) характеризуется набором свойств, определяющих, насколько продукт «хорош» с точки зрения заинтересованных сторон. Каждый из участников его создания может иметь свое представление о свойствах, а также о том, как оценить продукт. Модель качества в рамках стандарта ISO 9126 состоит из 6 факторов, где одним из важнейших является надежность программного продукта.

1. Определение надежности ПП

Наиболее весомой причиной отсутствия методов оценки надежности ПП является неразработанность соответствующей теоретической базы, приемлемой для исследования именно надежности ПП. Попытаемся разобраться в этом вопросе.

Первоначально определим, что такое *надежность* ПП. В работе [2] *надежность* определяется как *вероятность работы ПО без отказов в течение определенного периода времени, рассчитанная с учетом стоимости (степени воздействия) для пользователя каждого отказа*.

Из определения следует, что в понятии надежности применительно к ПП мы должны упоминать, какие типы ошибок имеются в виду. Понятие «надежность элементов ТУ» обычно не связывают с типами отказов. В данном определении надежности ПП есть заимствование из терминологии надежности ТУ – фактор времени.

В ПП ошибки возникают при определенных сочетаниях исходных данных. Бесчисленные запуски программы при ограниченном количестве наборов исходных данных, где она работоспособна, в течение любого интервала времени не породят ни одного отказа. Надежность программных модулей (ПМ) со временем может даже увеличиваться за счет устранения выявляемых ошибок. Поэтому теоретически может возникнуть ситуация, когда при тестировании ошибки уже не будут обнаруживаться.

Такое положение вещей делает совершенно бессмысленным для программирования понятия наработки на отказ или вероятности отказа за определенное время, на которых построена вся теория надежности в технике.

Аксиоматически классическая теория надежности строится на предположении о том, что существует пусть маленькая, но не нулевая вероятность отказа p_0 любого ТУ. Программы не изнашиваются и не ломаются, следовательно, ненадежность ПП – следствие исключительно ошибок проектирования, внесенных в процессе разработки. В то же время можно разработать любое количество простых программ, для которых вероятность отказа равна нулю. Но в этом случае весь аппарат классической теории надежности становится непригодным для практического использования.

Наиболее общее определение *надежности* ПП предлагает Б. Мейер [3], он определяет надежность как *способность программы давать разумные результаты во всех возможных окружениях, и в частности в аномальных условиях*.

В этом случае ненадежность ПП можно трактовать как соотношение мощности множества ошибочных ситуаций $|\Omega_E|$ и мощности множества исходных данных $|\Omega_{In}|$ ($\Omega_E \subset \Omega_{In}$). Применительно к ЭВМ это отношение между количеством сочетаний исходных данных программы, когда возникает ошибочная ситуация, к общему числу сочетаний. В этом случае ненадежность можно вычислить по формуле $q = \left|\Omega_E\right| / \left|\Omega_{In}\right|$.

Однако количество сочетаний исходных данных ПМ (в общем случае) настолько велико, что перебрать все для современной ЭВМ практически невозможно. Т.о., проблему оценки надежности ПП можно сформулировать как комбинаторную проблему поиска алгоритмов частичного перебора пространства исходных данных модуля. Для сравнительно простых ПМ, имеющих несколько входных параметров, получены положительные результаты. Для некоторых модулей удается доказать корректность [3], т.е. $q = 0$. В других случаях q можно определить экспериментально [4, 5].

Вместо дискретной меры – мощности – введем непрерывную меру, оценивающую объем многомерного тела, тогда ненадежность модуля можно оценить по формуле

$$q = V(\Omega_E) / V(\Omega_{In}). \qquad (1)$$

Для современных сложных ПП число исходных данных измеряется тысячами переменных, поэтому прямые методы тестирования становятся практически нереализуемыми. Кажется естественным оценивать надежность сложных ПП, используя известные характеристики составляющих его компонент так же, как это делается для ТУ. Однако и здесь обнаруживаются неожиданности:

1) как бы ни была сложна структура ПП, она всегда имеет (в смысле надежности) последовательную схему соединения элементов (программных модулей ПП);

2) надежность ПП зависит не только от надежности составляющих его программных модулей, но и корректности организации функционирования ПП [6];

3) надежность ПП не может быть вычислена непосредственно, исходя из надежностей составляющих его элементов (программных модулей).

2. Постановка задачи

Предположим, что нам известны характеристики надежности всех ПМ, из которых составлен ПП. Пусть q_i – ненадежность i-го модуля $A_i : X_i^{In} \to Y_i^{Out}$, вычисленная с учетом всевозможных применений модуля, т.е. на достаточно широкой области значений исходных данных Ω_{In}^i: $X_i^{In} = (x_{1i}^{In}, x_{2i}^{In}, ..., x_{n_i i}^{In}) \in \Omega_{In}^i$. Как же с помощью известной надежности модуля получить значение надежности всей программы?

Допустим, что логические ошибки в организации логики функционирования ПП отсутствуют (данная проблема требует отдельного рассмотрения, см. [4]).

Пусть некоторый ПП имеет L маршрутов исполнения [6]. Каждый из маршрутов обозначим $M_j = i_{j_0} i_{j_1} ... i_{j_k}$, где i_{j_k} – номер ПМ для ПП. Пусть каждый i-й объект на j-м маршруте вызывается в среднем $\tilde{m}_{ij}$ раз. Ненадежность i-го модуля на j-м маршруте за $\tilde{m}_{ij}$ вызовов можно оценить величиной $Q(\tilde{m}_{ij}) = 1 - (1 - q_i)^{\tilde{m}_{ij}} = Q_{ij}$.

Очевидно, что если в любом из модулей маршрута возникнет ошибка, то маршрут считаем ошибочным. Тогда ненадежность маршрута: $Q_{M_j} = 1 - \prod_{i=1}^{j_k} (1 - Q_{ij})$.

Пусть каждый из маршрутов реализуется с вероятностью r_j, причем справедливо $\sum r_j = 1$. Тогда ненадежность ПП можно оценить величиной

$$Q_n = \sum_{j=1}^{L} r_j Q_{M_j} \ . \qquad (2)$$

Формула (2) справедлива, если оценки ненадежности ПМ неизменны при различных использованиях модулей, но это не так.

3. Пример программы

Рассмотрим программу термогазодинамического расчета двухвального ТРД.

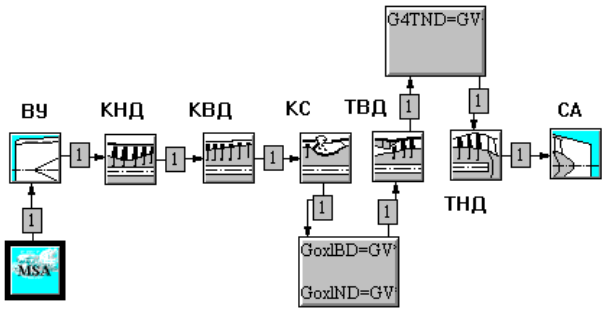


Рис. 1. Программа термогазодинамического расчета двухвального ТРД

Представленная программа достаточно проста и имеет единственный маршрут. Однако при ее выполнении изменяются области изменения исходных данных комплектующих ее модулей. На рис. 2-7 показаны области изменения исходных данных ПП расчета ТРД и его узлов (турбин высокого и низкого давления и др.).

Как видно из рисунков, изменилась топология областей исходных данных модулей, особенно для турбины НД и соплового аппарата, а для компрессора ВД область исходных данных выродилась в кривую. Первоначально при тестировании области исходных данных имели прямоугольный вид. Кроме того, уменьшились размеры областей исходных данных. Все это способно увеличить характеристику ненадежности ПМ за счет уменьшения знаменателя формулы $q_i = V(\Omega_E^i) / V(\Omega_{In}^i)$.

Но еще большую проблему представляет изменение законов распределения (ЗР) исходных данных модулей за счет соответствующих функциональных преобразований первоначально равномерного распределения исходных данных ПП. На рис. 8 показаны гистограммы распределения входных параметров P3ND, T3ND (входные для соплового аппарата) и P3BD (входного для модуля расчета турбины НД).

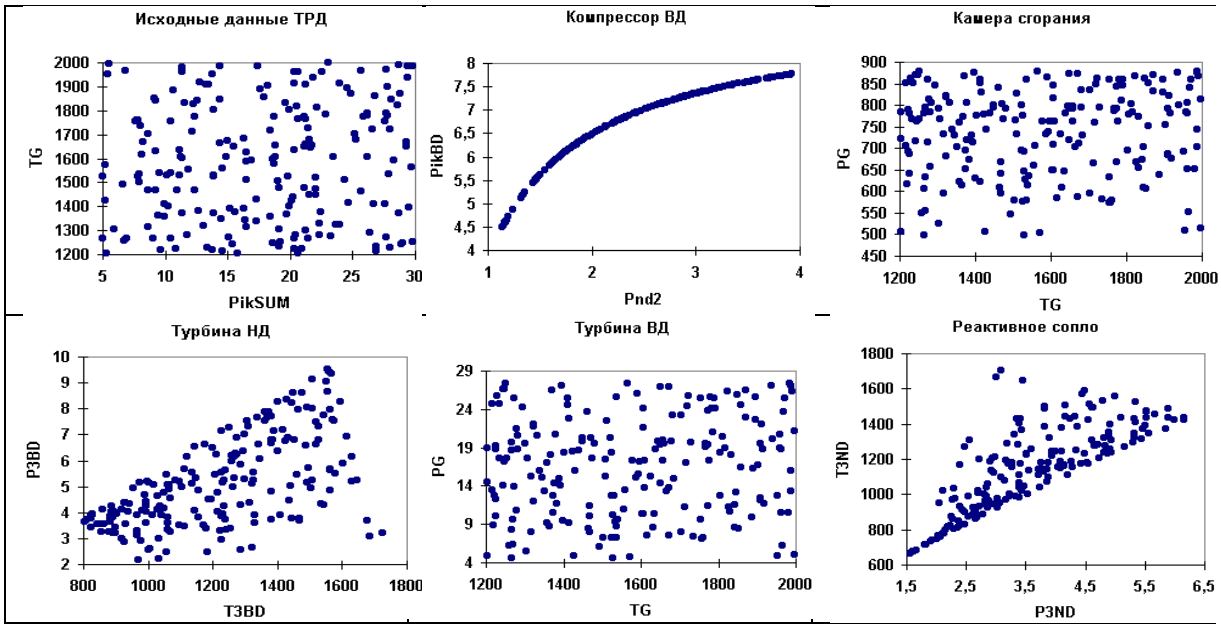


Рис. 2-7. Области изменения исходных данных ПП расчета ТРД и его узлов

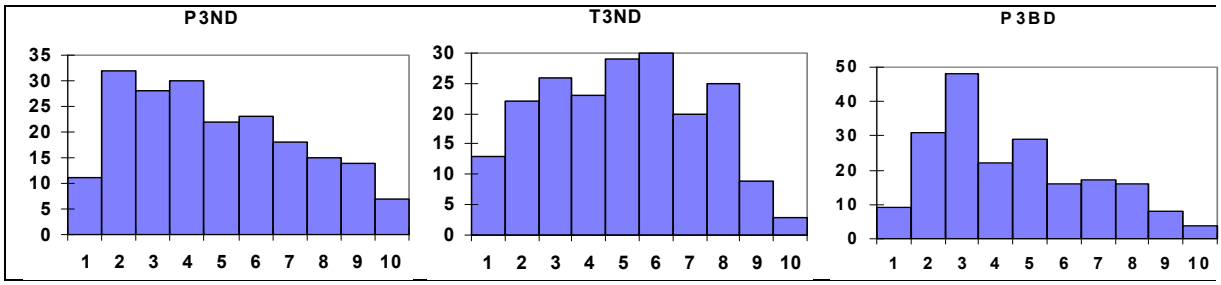


Рис. 8. Гистограммы распределения входных параметров P3ND, T3ND и P3BD

4. Влияние вероятностных мер на изменение в оценке надежности ПМ

Пусть некоторый модуль, имеющий единственных входной параметр x, «прошел» серию испытаний (из N испытаний), в которых не было обнаружено ни одной ошибки, причем $x \in [a,b]$. из чего можно сделать вывод, что $q \le 1/(N+1)$ и $V(\Omega_E) \approx (b-a)/(N+1)$. Пусть теперь при использовании модуля в ПП изменился диапазон значений входного параметра модуля $x \in [c,d] \subset [a,b]$ без изменения его ЗР. Итоговую ненадежность модуля, с учетом вероятности $P\{\Omega_E \subset [c,d]\}$ можно оценить следующим образом: $\tilde{q} = \frac{V(\Omega_E)}{d-c} P\{\Omega_E \in [c,d]\} = \frac{V(\Omega_E)}{d-c} \cdot \frac{d-c}{b-a} = \frac{V(\Omega_E)}{b-a} = q$, т.е. для равномерного ЗР параметра x характеристика ненадежности модуля не изменяет своего значения.

Для простоты положим a=c=0. Пусть теперь на отрезке [0, d] параметр x имеет квазиэкспоненциальное распределение с функцией плотности распределения $f(x) = K\lambda e^{-\lambda x}$, где $K = \frac{1}{1-e^{-\lambda d}}$. Положим $V(\Omega_E) = \Delta$. Ненадежность модуля можно определить из выражения $\tilde{q}(x) = P\{\Delta \in [0,d]\} \int_x^{x+\Delta} K\lambda e^{-\lambda x} dx = \frac{d}{b} \frac{e^{-\lambda x}}{1-e^{-\lambda d}} (1-e^{-\lambda \Delta})$.

Ненадежность модуля теперь зависит от того, куда попадет область ошибочных ситуаций, и не равна полученной в результате автономного тестирования величине q.

В предположении равновероятности попадания области ошибочных ситуаций в любое место отрезка [*0, d*] вычислим среднюю величину $\tilde{q}(x)$

$$q_{cp} = \int_0^d \tilde{q}(x)\frac{1}{d}dx = \frac{P\{\Delta \in [0,d]\}K(1-e^{-\lambda\Delta})}{\lambda d}\int_0^d \lambda e^{-\lambda x}dx = \frac{d}{b}\frac{(1-e^{-\lambda\Delta})K}{\lambda d}(1-e^{-\lambda d}) = \frac{(1-e^{-\lambda\Delta})}{\lambda b}.$$

Введенная характеристика не зависит от параметра x и определяется только размерами области ошибок (Δ) и параметром экспоненциального ЗР (λ). Как показывают расчеты (см. таблицу 1) существенные изменения значений λ (степени неравномерности распределения) практически не сказываются на величине q_{cp}. Более того, $q_{cp} \approx q$.

Таблица 1. Зависимость q_{cp} от λ

λ	q_{cp}	max $q(x)$	Δx
0,1	9,99999E-06	1,01E-05	0,1
1	9,99995E-06	1,1E-05	0,0975
10	9,9995E-06	2,31E-05	0,0825
100	9,995E-06	0,0002	0,0275
1000	9,95017E-06	0,00199	0,005

В данном примере, $q = 0.00001$, $d = 0.2$, $b = 1$. Для больших неравномерностей $\lambda > 10$ условие $q(x) > q_{cp}$ наблюдается на сравнительно небольших участках области определения входного параметра $x - \Delta x$, а в остальных случаях отмечается $q(x) \le q_{cp}$.

При небольших значениях λ справедливо $\max q(x) \approx q_{cp}$. Если $q_{cp} \approx q$ справедливо для любых ЗР, то оценки q_i можно использовать для вычисления надежности ПП, интерпретируя Q_n как среднюю ожидаемую величину ненадежности ПП. Однако данный результат необходимо еще обобщить на многомерный случай.

Выводы

Структура ПП всегда имеет последовательную схему соединения программных модулей ПП в смысле надежности. Надежность ПП зависит от корректности организации функционирования системы и не может быть вычислена непосредственно, исходя из надежностей программных модулей.

Т.о., оценивание надежности ПП существенно отличается от решения аналогичной задачи для ТУ и требует разработки соответствующей теоретической базы.

Литература

1. Авакян А.А., Искандаров Р.Д., Новиков Н.Н. и др. Концепция построения высоконадежных вычислителей для авиационной и ракетной техники //Надежность и качество – 2001: Сб. докладов межд. симпоз. Пенза, 2001. С. 33-37.
2. Майерс Г. Надежность программного обеспечения. – М.: Мир, 1980. – 360 с.
3. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах. Т.2. – М.: Мир, 1982.
4. Коварцев А.Н. Автоматизация разработки и тестирования программных средств.– Самар. гос. аэрокосм. ун-т. Самара, 1999. – 150 с.
5. Коварцев А.Н. Автоматизация тестирования вычислительных модулей // Надежность и качество – 2001: Сб. докладов межд. симпоз. Пенза, 2001. С. 285-288.
6. Липаев В.В. Надежность программных средств. – М.: СИНТЕГ, 1998. – 232 с.

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОИСКА СТРУКТУР И РЕАКЦИЙ ДЛЯ МОДЕЛИРОВАНИЯ ПРОЦЕССА РОСТА МОНОКРИСТАЛЛИЧЕСКОЙ АЛМАЗНОЙ ПЛЁНКИ

Г.Ю. Аверчук[1], Е.С. Куркина[2], Э.М. Кольцова[1]

[1]РХТУ им. Менделеева, каф. ИКТ

[2]Московский госуниверситет им. М.В. Ломоносова

На основании анализа образуемых в процессе роста кристалла алмаза поверхностных структур, а также возможных их изменений посредством реакций построен предметно-ориентированный язык. С его помощью описывается кинетическая схема роста кристаллической плёнки с десятками элементарных стадий. Анализ кинетической схемы позволяет получить рецепты для эффективных параллельных алгоритмов поиска поверхностных структур и элементарных реакций, их изменяющих. Каркас приложения с методом Монте-Карло ориентирован на системы с общей памятью. На основе каркаса и полученных рецептов генерируется высокопроизводительный параллельный код, компиляция которого позволяет осуществлять моделирование изменения поверхности кристаллической плёнки, содержащей сотни тысяч атомов.

Введение

Алмазные кристаллические плёнки имеют большой потенциал возможного применения в бурильной, медицинской и наноэлектронной промышленности, а также могут служить в качестве химически устойчивых покрытий для различных материалов [1]. Одним из способов получения алмазных кристаллических плёнок является процесс осаждения углеводородных радикалов из высокотемпературной плазмы (CVD), содержащей большое количество водорода. Методами спектрального анализа, атомно-силовой микроскопии, молекулярной динамики и квантовой химии различными исследователями были определены атомарные структуры, образуемые на поверхности растущего кристалла алмаза [2]. Помимо структур, исследователями были рассчитаны значения параметров различных элементарных реакций между такими структурами [3].

Кинетическая схема базового случая осаждения плёнки из метил-радикалов содержит десятки элементарных стадий, в которых участвует более тридцати всевозможных поверхностных структур. А наиболее применяемые на практике способы осаждения из смеси метана, ацетилена и небольшого количества азота содержат ещё большее количество структур и элементарных стадий. Задача кодирования алгоритмов поиска структур и реакций является рутинной и требует автоматизации на основе базовых эффективных алгоритмов. Понимание процесса образования и роста кристалла на атомарном уровне может позволить определить оптимальные условия проведения процесса роста плёнки.

1. Модель процесса осаждения алмазной плёнки

Микроскопическая стохастическая модель процесса осаждения алмазной плёнки из газовой фазы строится на формальной теории описания взаимодействий поверхностных структур и молекул газовой фазы, которые состоят из атомов и отношений между ними. В качестве символов алфавита формальной теории выделяются следующие типы атомов: водород (H), аморфный углерод (C) и кристаллический углерод (C_d), ис-

пользуя которые обобщённую кинетическую схему процесса взаимодействия кристалла с газовой фазой можно записать следующими тремя уравнениями:

$H^* + C_dH \rightarrow H_2 + C_d^*$,	*активация поверхности*
$H^* + C_d^* \rightarrow C_dH$,	*дезактивация поверхности*
$C_d^* + {}^*C_xH_y \leftrightarrow C_dC_xH_y$,	*осаждение углеводородного радикала*

где активные атомы обозначены знаком *, а *x* и *y* являются произвольными числами, удовлетворяющими валентности атомов.

В свою очередь, кристаллический углерод может принадлежать различным атомарным структурам, образуемым на поверхности кристалла. Кристаллическая решётка накладывает определённые ограничения на расположение атомов друг относительно друга, а также на типы связей, образуемых между атомами. Для описания одной связи или отношения положения между атомами используется плоскость кристалла, обозначаемая индексами Миллера, а также направление в этой плоскости.

С помощью атомов и различных типов связей описываются газовые молекулы и поверхностные структуры, которые вместе являются словами формальной теории. Для базового случая осаждения алмазной плёнки из метил-радикала выделяются виды структур, представленные на рис. 1.

Рис. 1. Основные поверхностные структуры, выделяемые на плоскости (100) кристалла алмаза: 1 – мост, 2 – димер, 3 – димер с метил-радикалом, 4 – мост с метил-радикалом, 5 – высокий мост, 6 – мост на следующем слое, 7 – два моста

Рис. 2. Некоторые элементарные стадии кинетической схемы осаждения алмазной плёнки из газовой фазы

Формулами формальной теории являются элементарные реакции, в которых поверхностные структуры взаимодействуют либо с радикалами газовой фазы, либо друг с другом, изменяя конфигурацию взаимодействующих атомов с образованием новых структур (рис. 2). В модели осаждения плёнки поток всевозможных реакций является стохастическим процессом и может быть представлен марковской цепочкой событий, а изменение вероятности состояния поверхности кристаллической плёнки в марковском приближении описывается основным кинетическим уравнением, где вероятность каждой реакции определяется её скоростью. Единственной аксиомой формальной теории является необходимость соблюдения материального баланса в реакциях.

Поскольку адекватная кинетическая схема процесса роста кристалла из газовой фазы имеет большую размерность, а её анализ представляется относительно сложной задачей, которую необходимо выполнить один раз перед расчётом, было решено отделить этап составления и анализа кинетической схемы от этапа её расчёта посредством стохастического метода Монте-Карло.

2. Предметно-ориентированный язык

На основании вышеописанной формальной теории процесса осаждения алмазной плёнки построен интерпретатор, оперирующий множествами необходимых сущностей, каждая из которых выражена некоторым синтаксисом, понятным интерпретатору. Множество синтаксических структур образует предметно-ориентированный язык, с помощью которого вся необходимая информация о кинетической схеме записывается в конфигурационном файле, который после интерпретации и анализа полученных данных на выходе даёт программный код, основанный на базовом каркасе приложения расчёта процесса.

Язык позволяет описывать атомы, молекулы газовой фазы и их концентрацию, структуры на поверхности плёнки, взаимные расположения между атомами которых выводятся автоматически, согласно правилам вывода формальной теории, в соответствии с заданной кристаллической решёткой. С помощью атомарных структур описываются элементарные реакции кинетической схемы, и для каждой указываются необходимые константы, такие как предэкспоненциальный множитель и энергия активации. На этапе интерпретации уравнения реакции по валентностям задействованных атомов и количеству связей между ними проверяется материальный баланс. После проверки баланса реакции с целью определения атомов, подвергшихся изменению, производится сопоставление атомов в реагирующих структурах и определяется, каким образом атомы изначальных структур переходят в атомы продуктов. Сопоставление атомов осуществляется с помощью модернизированного алгоритма Хансера, учитывающего принадлежность атомов кристаллической решётке, а также возможность изменения типов отношений между атомами.

После интерпретации введённые данные анализируются и строятся зависимости структур друг от друга на основе максимальной общей подструктуры (MCS) и алгоритма Хансера [4]. Язык предоставляет инструмент для описания сначала общей реакции взаимодействия некоторых структур, а затем, если необходимо, более конкретных, зависимых от взаимного расположения реагирующих структур или от локального окружения обеих структур. На основании этого производится относительное упорядочивание зависимых реакций.

Атомы реагирующих структур классифицируются по типам (рис. 3) с учётом используемых связей и относительных характеристик, таких как «не связанность» и «не фиксированность». Между полученными атомами строятся отношения включения одного типа атома в другой (рис. 4), по которым, на этапе расчёта, выполняется алгоритм поиска структур. Для базового случая осаждения из метил-радикала выделяется более 30 различных типов атомов.

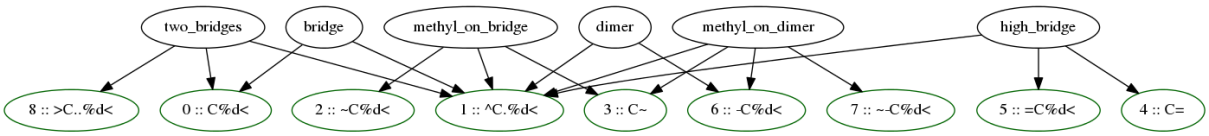


Рис. 3. Отношение атомов к структурам, из которых они состоят

После определения всех необходимых зависимостей данных производится генерация выходной информации, например изображений структур и реакций (рис. 1, 2), графов (рис. 3, 4) или программного кода, содержащего рецепты поиска и использующего методы каркаса расчётной программы [5].

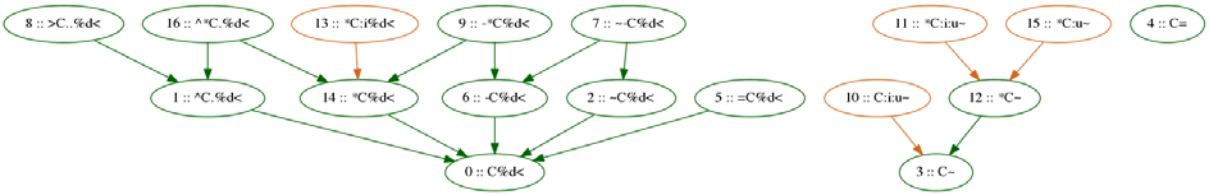


Рис. 4. Лес неявного преобразования типов атомов, необходимый для реализации эффективного алгоритма поиска структур, где *i* означает «не связанность», а *u* – «не фиксированность»

3. Параллельный алгоритм поиска структур и реакций

В начальный момент расчёта модели роста создаётся массив атомов, в котором хранятся и атомы, принадлежащие кристаллической решётке, и множество для аморфных атомов, связанных с поверхностью. Создаются два первых слоя атомов, в которых атомы инициируются определяемыми в конфигурационном файле типами, а затем обновляют своё относительное состояние в параллельном режиме. После инициализации для каждого атома параллельно запускается поиск структур, и, поскольку структуры могут включать друг друга, поиск начинается с наименьшей структуры. В случае успешного нахождения какой-либо структуры активируются задачи на поиск структур, включающих найденную. Поиск осуществляется благодаря рецептам, полученным после анализа интерпретируемых данных. Рецепт представляет собой последовательность типов атомов, с учётом распространения связей между атомами в структуре, их отношений друг к другу. Другими словами, рецепт является монадным выражением проверок типов атомов, с оператором *И* между ними, и, следовательно, может быть осуществлён параллельно. При проверке типов анализируемых атомов используется транзитивная матрица смежности, построенная для графа на рис. 4, позволяющая быстро определять, имеет ли рассматриваемый атом требуемый тип. Каждый атом запоминает все структуры, в описании которых он состоит, и какую он при этом играет роль, представляясь атомом необходимого типа на этапе поиска структуры.

После того как все возможные структуры найдены, запускается алгоритм поиска реакций, также основанный на рецептах, полученных после анализа языкового файла. С целью нахождения атомов определённых типов, принадлежащих структурам, необходимым для осуществления реакции, для каждой структуры в параллельном режиме производится анализ локального окружения. Найденные реакции запоминаются в векторах, соответствующих типу реакции, и используются в динамическом алгоритме Монте-Карло, благодаря которому среди всех найденных реакций выбирается наиболее вероятная, которая, осуществляясь, изменяет типы некоторых атомов реагирующих структур. Затем, если требуется, изменённые атомы обновляют своё относительное состояние, и в параллельном режиме проверяются все структуры, которым принадлежит атом, на соответствие тому типу, в роли которого он используется для данной структуры. Если соответствия нет, структура и использующие её реакции удаляются. Если соответствие найдено, состояние относительно этой структуры не обновляется. Таким образом, в каждый момент времени, атомы моделируемой поверхности имеют типы, соответствующие истокам на рис. 4. От подвергшихся изменениям атомов снова инициируется поиск новообразованных структур, и алгоритм повторяется.

Заключение

Для моделирования процесса осаждения алмазной плёнки из газовой фазы был реализован комплекс программ, позволяющих лаконично описывать кинетическую схему процесса фазового перехода. На основании кинетической схемы генерируется программный код, основанный на каркасе расчётного приложения, предоставляющего

эффективные базовые алгоритмы для организации параллельных вычислений. По сравнению с последовательным кодом параллельный алгоритм даёт увеличение производительности в 2.7 раза, на 4-ядерном процессоре.

Работа выполнена при финансовой поддержке РФФИ, проекты №11-01-00887 и №11-08-00979-а.

Литература

1. Sussmann R.S. CVD Diamond for Electronic Devices and Sensors. WILEY, London, 2009. – 582 p.
2. Cheesman A. Investigations into the fundamentals of gas-phase and gas-surface chemistry prevalent in growth of CVD diamond films. Dis. of doc. of phil., 2006. – 292 p.
3. Frenklach M., Skokov S. Surface Migration in Diamond Growth // J. Phys. Chem. B. – 1997. – Vol. 101. P. 3025-3036.
4. Савельев А.С.. Алгоритм поиска максимальной общей подструктуры набора молекул / SciTouch LLC., 2010. – С. 42.
5. https://github.com/newmen/versatile-diamond.

SCENE BOUNDARY DETECTION TECHNIQUE BASED ON BOTTOM-UP ATTENTION SYSTEM AND OPENCL PARALLEL IMPLEMENTATION

S. Axyonov, D. Kovalenko, I. Potapyev

National Research Tomsk Polytechnic University

E-mail: axoenowsw@tpu.ru, wf34@sibmail.com, ipotapev@gmail.com

This paper spotlights the maintaining of scene boundary detection system in video and process of porting it to the OpenCL. The scene boundary detection algorithm proposed by authors is based on bottom-up focus attention principle. The system builds Gaussian pyramids from input image, calculates map of saliency from the image and then detects the most salient regions. The scene cut is detected when correlation of previous and next image's histograms of salient areas is lower than threshold. To decrease the algorithm's running time a parallel architecture was implemented. Several tricks and techniques which served for reaching better performance are described.

Keywords: scene detection, pyramid of image, video analysis, OpenCL, parallel processing.

Introduction

In order to process video data efficiently, a video segmentation technique through scene cut detection must be required. This is a fundamental operation used in many digital video applications such as digital libraries, video on demand (VOD), etc. There is set of approaches used for solving this task. In [1][2] authors propose system for automatic detection of replay segment based on image templates. Paper [3] describes algorithm of processing a compressed digital video MPEG bitstream to detect scene changes. The main feature of chosen algorithm is ease of paralleling code development to increase the performance. In this paper, we proposed an approach based on using bottom-up attention[4].

The proposed algorithm contains huge amount of consecutive operations, where result of the previous one doesn't affect result of the next one. It means that such kind of applications contain a huge amount of the parallelism. Hence, execute it in a serial manner would be ineffective and slow. Our goal was to take advantage of the natural parallelism of the task and implement the parallel version of the system.

Hence, we had a goal to pick parallel programming technology which is suitable for servers and desktops. Most of them are highly specialized. For example, MPI oriented for execution on clusters and OpenMP devoted for creating parallel code for SMP systems. Finally, OpenCL was chosen. This decision is supported with following reasons:

- General purpose GPU is the approach acceptable for desktops and servers both
- OpenCL is oriented to solving mathematical tasks
- OpenCL technology supported with all vendors (NVIDIA, ATI, Intel , etc.)

Algorithm

The system takes video file as input by using OpenCV functions for consequent access to the frames. Each frame of video is represented as RGB image. Next, Gaussian pyramids of the frame are created for construction maps of characteristic features. Gaussian pyramid is a set of images that are duplicates of each other and where the next image is smaller than the previous one in 2 times[5]. The number of levels of the pyramid is determined by the mini-

mum allowable size of the image within the pyramid. The Gaussian blur is used to avoid the negative impact of pixelation when building pyramid.

The system builds five pyramids for each frame. The first pyramid characterizes the intensity of the image.The other four pyramids consist respectively of the four images, each of which represents a specific color channel, red, green, blue, or yellow, negative values are set to zero [6].

First set of feature maps is related to intensity contrast of image and consist of six maps I(c,s). Each of them is calculated by formula

$$I(c,s) = \left|I(c) - I(s)\right|$$

Where $c \in \{2,3,4\}$ and $s = c + \sigma$, $\sigma \in \{3,4\}$ are index levels of pyramid, «-» is center-surround difference[7].

Next sets of feature maps are connected with color input. Two sets of feature maps are combined by contrast principle from R, G, B, Y pyramids:

$$RG(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))|$$
$$BY(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))|$$

Feature maps are combined into three "conspicuity maps", I_{cons} for intensity, C_{cons} for color at the scale ($\sigma = 4$). They are calculated by across-scale addition, $\oplus$, which consists of reduction of each map to scale 4 and point-by-point addition:

$$I_{cons} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} I(c,s)$$
$$C_{cons} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} [RG(c,s) + BY(c,s)]$$

Next, the two conspicuity maps are normalized and summed into the final input S to the saliency map which is used by focus attention subsystem for the computation of the mask frame[8]. The motivation for creation of two separate channels, $\bar{I}$ and $\bar{C}$, is the hypothesis that similar features compete strongly for saliency, while different modalities contribute independently to the saliency map. Saliency map is calculated by naïve summation[9].

According to figure 1, the location of the areas with the greatest intensity on feature map significantly correlated with the location of the main objects on the frame. Thus, finding the local maximum on the map and use of adaptive threshold cutting off the area around the local maximum allow to apply these areas on the mask for obtaining frame information which is important for the efficient detection of scene boundaries.

Obtained salient areas define the regions on the image where histogram correlation would be computed for previous and next frames

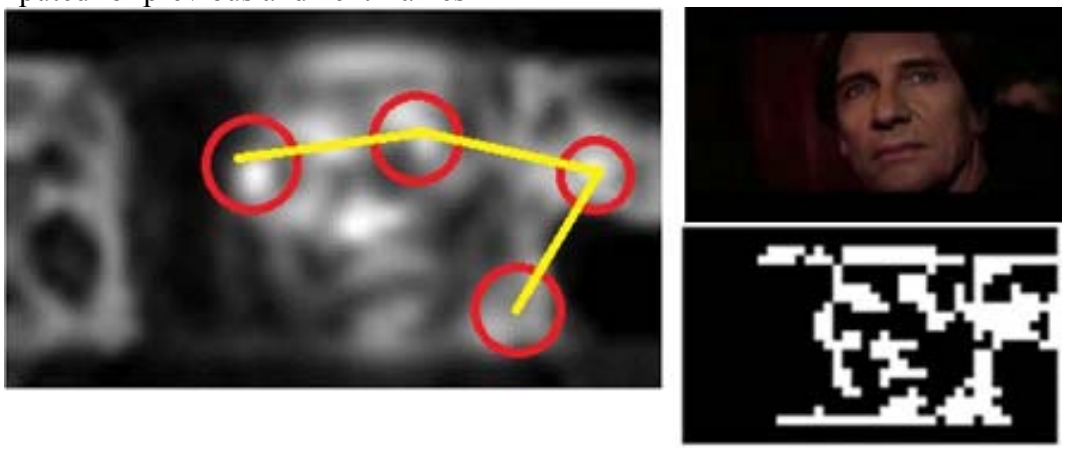


Figure 1. Salient regions on an image

The correlation function is described by the formula:

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}},$$

where H_1 – histogram of the previous frame, H_2 – histogram of the current frame $\bar{H}_k = \frac{1}{N}\sum_J H_k(J)$, N – number of histogram bins.

Thus, when passing of frames is done, the frame correlation vector for all frames is built. Frames where correlation is close to 1.0 are frames of one scene. The low values of the correlation with a high probability indicate a scene cut between these frames. Filtering of the vector on the threshold gives a number of pairs of frames located on the boundaries of scenes. Filtration is performed by using of adaptive threshold:

$$T = \sum_{i=0}^{2n} C_{ki}\varphi_i \ , \qquad \varphi(C) = \frac{1}{\sigma_c\sqrt{2\pi}} e^{-(c-M_c)^2/2\sigma_c^2} .$$

Where T – threshold value, φ_i – weighting factor, n – number of neighbors of the analyzed element, M_c – mean value, σ_c^2 – variance, $C_n = \frac{d(R_{N-1}, R_N) + d(G_{N-1}, G_N) + d(B_{N-1}, B_N)}{3}$ – elements of the analyzed vector, $d(x, y)$ – correlation function, R_i, G_i, B_i – vectors obtained from RGB histogram of the current frame. Obtained vector contains binary values, where 1 indicates a cut between scenes.

Architecture of parallel solution

Architecture of the system was slightly reconsidered and rewritten with the purpose to develop clear and supportable parallel code. Writing of parallel code with use of any technology has lots of pitfalls and dramatically slower than writing serial code. So, rewriting of the architecture allowed to deliver several important features.

First feature is that OpenCL and C++ versions are interchangeable. They represented as classes PyraProbe and OCLPyraProbe on figure 2. These classes encapsulate all business-logic related to pyramidal analysis of the frame (all computationally expensive work). Figure 3 shows that at the highest level of application call stack code operates with IBasicPyraProbe objects without even knowing which implementation it is.

Also, both versions are kept in the same project in purpose of easier building and execution in both ways. The library is being built with or without OpenCL support based on preprocessor definitions, and being executed serial or parallel way based on given settings.

Another architecture feature is namespaces Processors and OCLProcessors which shown on figure 3. Both of them contain image processing functions which are exploited by PyraProbe and OCLPyraProbe respectively. Some examples of these functions are listed for better insight of the idea:

- generation of the 2D Gaussian distribution,
- 2D convolution,
- normalizing color image,
- creating image pyramid.

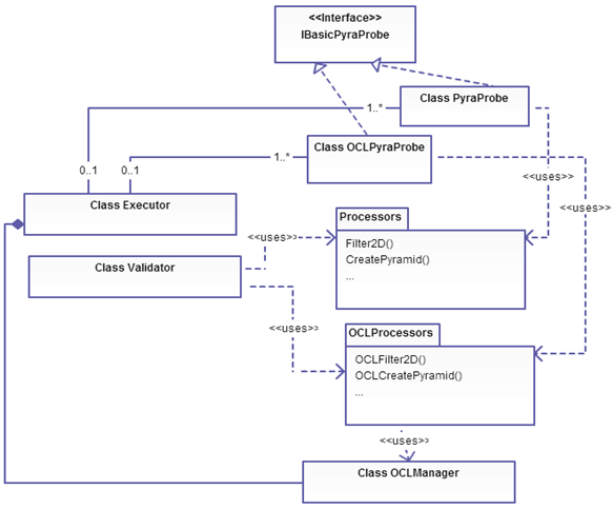


Figure 2. Architecture of the developed system

This design allowed to develop parallel version iteratively and enabled unit-testing of these functions. Class Validator serves for this functionality. So developers were able to prove that some function ported successfully by comparison of the results of C++ and OpenCL version of the function.

OCLManager serves for handling all SDK-related calls. It builds and stores kernels, keeps OpenCL context and other SDK objects required for execution.

Other aspects of the architecture:

- Minimized interaction between RAM and videomemory. Copying of image occurs only twice: delivering source image to OCLPyraProbe and then acquiring result back to RAM. All computation in between is being performed on GPU.
- Modularity of the system enables easy code adaptation for execution on the GPU farm.

When development of the system with architecture described above was finished, performance tests have been performed.

These tests revealed slight improvement at the high-resolution video, but parallel version didn't outperform serial one on the low-resolution videos.

Most of the application running time was taken by convolution calculation. Filtering is so such computationally expensive because model uses big kernels (25x25, 21x21). So, filter2D is the most promising target for optimizations. One of the properties of the efficient GPGPU code is low divergence. This means that all threads of kernel have approximately same execution time. If control flow of the kernel has a lot of conditional statements, most likely that code will be slow [10].

Removing of the conditionals gave significant speed gain for our case.

OpenCL global memory has huge capacity but the slowest access speed. So, the only way to implement something efficient is to put an effort on extracting advantage of the local and constant memory zones.

There is possibility to define memory buffers as constant if buffers are read only. Also size of the constant buffer is bounded by value which depends on computing device. So, first thing to do was to move all filter buffers from global memory to constant.

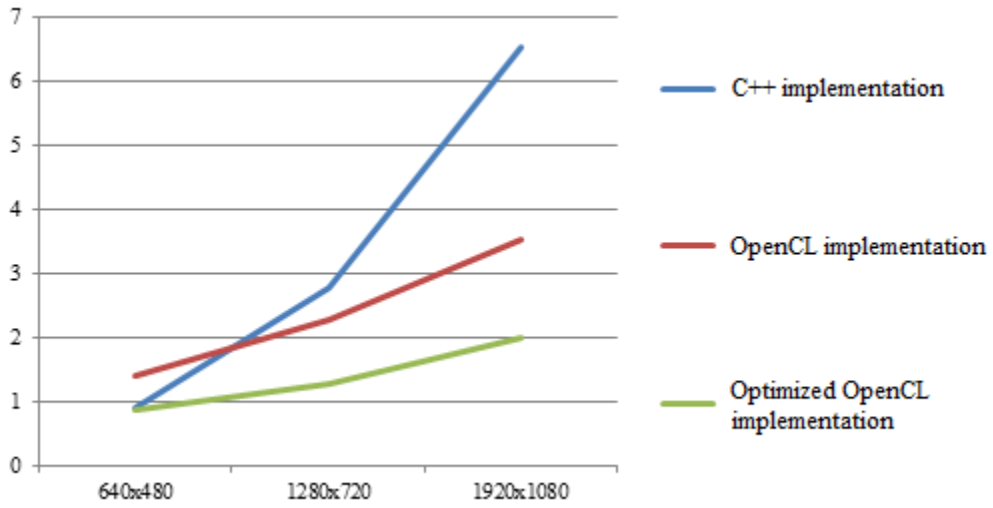


Figure 3. Time comparison of processing one frame

Another way to speed up convolution operation is to avoid reading neighbor pixel values from the global memory [11]. So, filter2D kernel execution model was changed to use workgroups of size 16x16. Items of the same workgroup load part of the image to the local memory and then use this data for convolution calculation.

It is good practices to keep amount of kernel arguments as small as possible. As it obvious from table 2, filter2D was separated to two kernels. This action enables us to not pass size of kernel (and some other variables like that) as argument, but define it as a constant. These values are used in the loop code; consequently, compiler will be able to implicitly unroll loops [12].

Our optimization work is not finished yet, but even now it is obvious that GPGPU solution achieved performance on this problem of image analysis which is way more effective than serial CPU solution. Tests which are represented in figure 3 ran at desktop with Intel i5-430m processor and AMD Radeon HD 5850M.

So, plans for future development are:

- Continuation of the OpenCL code optimization
- Work on the computational model for increasing precision and decreasing computational complexity.

This research was partially supported by the “Nauka” program of Ministry of Education of Russian Federation, contract number 8.8113.2013 and a grant from Russian Foundation for Basic Research RFFI 13-07-00397A.

References

1. Pan, Hao, Baoxin Li, Sezan, M.I. Automatic detection of replay segments in broadcast sports programs by detection of logos in scene transitions // Acoustics, Speech, and Signal Processing (ICASSP). 2002.

2. Papageorgiou C.P., Oren M., Poggio T. A General Framework for Object Detection // Proceedings of the Sixth International Conference on Computer Vision. 1998.

3. Jianhao Meng, Yujen Juan, Shih-Fu Chang. Scene change detection in an MPEG-compressed video sequence // Proc. SPIE 2419, Digital Video Compression: Algorithms and Technologies 1995. 14.

4. Kovalenko D., Potapyev I. Scene boundary localization based on contrast region analysis // Tomsk Polytechnic University, MSIT №10. 2012. – P. 60–62.

5. Burt P.J., Adelson E.H. The Laplacian Pyramid as a Compact Image Code // J. IEEE Transactions on Communications Communications. Volume 31, Issue 4. 1983. 532–540.

6. Ware C. Color sequences for univariate maps: theory, experiments and principles // Computer Graphics and Applications, IEEE. Volume 8. Issue 5. 1988. 41–49.

7. Sun-Gu Sun, Dong-Min Kwak, Won Bum Jang, Do-Jong Kim. Small target detection using center-surround difference with locally adaptive threshold // Image and Signal Processing and Analysis. Proceedings of the 4th International Symposium on. 2005. 402–407.

8. Zhaoping Li. A saliency map in primary visual cortex // Trends in Cognitive Science. Volume 6. Issue 1. 2002. 9–16.

9. Xiaodi Hou, Liqing Zhang. Saliency Detection: A Spectral Residual Approach // Computer Vision and Pattern Recognition, IEEE Conference on. 2007. 1–8.

10. In Kyu Park, Singhal N., Man Hee Lee, Sungdae Cho. Design and Performance Evaluation of Image Processing Algorithms on GPUs // Parallel and Distributed Systems, IEEE Transactions on. 2011. Volume 22. Issue 1. 91–104.

11. Goorts Patrik, Rogmans Sammy, Vanden Eynde Steven, Bekaert P. Practical examples of GPU computing optimization principles // Signal Processing and Multimedia Applications (SIGMAP), Proceedings of the 2010 International Conference on. 2010. 46–49.

12. Han Dong, Ghosh, D., Zafar, F., Shujia Zhou. Cross-Platform OpenCL Code and Performance Portability Investigated with a Climate and Weather Physics Model // Parallel Processing Workshops (ICPPW), 2012 41st International Conference on. 2012. 126 – 134.

СОЗДАНИЕ КОМПЛЕКСА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ АНАЛИЗА ЭЛЕКТРОФИЗИОЛОГИЧЕСКОЙ ИНФОРМАЦИИ ПРИ АПНОЭ

А.В. Андреева

Уральский государственный медицинский университет Минздрава России

Работа обобщает основные результаты анализа электрической активности головного мозга при апноэ методами нелинейной динамики. Представлены возможности применения разработанных программно-инструментальных средств оценки нелинейных живых систем, позволяющих определить основные хаотические характеристики и параметры системы.

Ночное апноэ – временная остановка дыхания во сне, которая может происходить за ночь многократно (более пяти событий в час) и длиться более десяти секунд. Нехватка кислорода в организме в результате многократных остановок дыхания во сне может привести к серьезным заболеваниям – инсульту, инфаркту миокарда, сердечной недостаточности, аритмии сердца, повышенному артериальному давлению и т.д. Различают три формы апноэ: центральную, обструктивную и смешанную. В данной работе была исследована обструктивная форма апноэ сна. При нормальном сне мышцы, которые контролируют язык и мягкое небо, удерживают верхние дыхательные пути в открытом состоянии и воздух поступает в легкие беспрепятственно. Но если эти мышцы расслабляются, то дыхательные пути оказываются суженными частично (возникает храп – из-за вибрации мягкой ткани гортани при вдохе) или полностью, и тогда возникает обструктивное апноэ сна, при этом дыхательные движения живота и грудной клетки сохраняются (рис. 1).

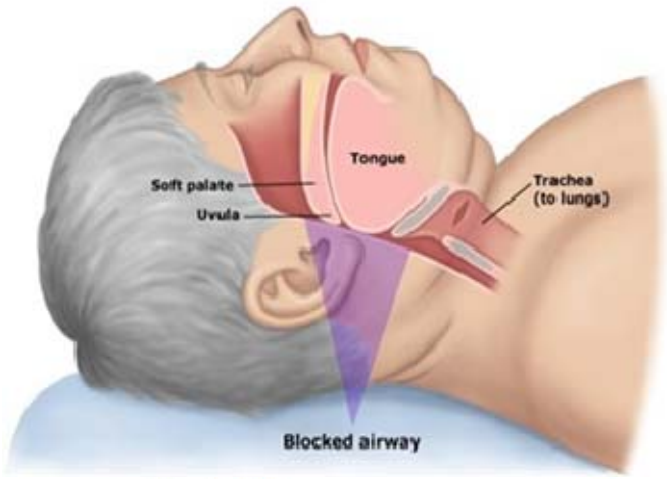


Рис. 1. Обструктивное апноэ сна

Диагностируют данное заболевание с помощью полисомнографического исследования – метода длительной регистрации различных функций организма во время сна (рис. 2). Регистрация параметров исследования осуществляется в течение всего ночного сна (6 – 8 часов).

Рис. 2. Пример записи сигналов различных функций организма полисомнографом

После удаления артефактов результаты обследования расшифровываются врачом. Одним из основных параметров регистрации является электроэнцефалограмма (ЭЭГ) – метод исследования электрической активности головного мозга, основанный на регистрации его потенциалов. На ЭЭГ можно выделить большое количество разнообразных ритмов и периодических колебаний. Частотный состав ЭЭГ, характеризующий электрическую активность головного мозга, может изменяться в зависимости от функциональной активности данной области и от ее взаимодействий с другими областями [1].

Отсутствие прямых клинических и электроэнцефалографических признаков, по которым можно было бы предсказать остановку дыхания, подтолкнуло к исследованию электрического потенциала головного мозга с помощью математического аппарата нелинейной динамики и теории динамического хаоса [2].

Цель исследования состояла в разработке программного продукта для диагностического комплекса оценки электрофизиологической информации с использованием методов нелинейной динамики и искусственного интеллекта.

В ходе работы были рассмотрены возможности интеллектуальной обработки и применения разработанных программно-инструментальных средств оценки нелинейных систем, позволяющих определить основные хаотические характеристики и параметры системы. На базе АНО «Клинический институт мозга» было проведено полисомнографическое исследование 42 пациентов (10 женщин и 32 мужчин) с данным заболеванием. Исходные медицинские данные представляют собой пары наборов действительных чисел. Каждый набор получен в результате дискретизации сигнала, шаг дискретизации равен 0,005 с. Для реализации программного продукта был выбран язык программирования C# (целевая платформа .NETFramework). Минимальные рекомендуемые системные требования совпадают с таковыми для клиентского профиля .NET 4. В программе используются возможности библиотеки Task Parallel Library, благодаря чему программа эффективно использует ресурсы многоядерных систем, получая значительный прирост производительности от увеличения числа исполнителей, автоматически подстраиваясь под их количество. Для построения графического интерфейса использовалась система Windows Presentation Foundation. Это позволило создать визуально приятные графики без существенных затрат ресурсов непосредственно на их отображение. Для реализации нейросетевой обработки хаотических процессов, которая по-

зволяет идентифицировать хаотическое поведение динамической системы и прогнозировать ее временные характеристики, было принято решение воспользоваться пакетом GANS (Genetic algorithms and neural systems [3,4]). Пакет содержит общую архитектуру и реализацию на ее основе нейронных сетей, генетических алгоритмов и алгоритмов коллективного разума. За счет особенностей архитектуры достигаются широкие возможности внесения в пакет дополнительного кода с реализациями нестандартных алгоритмов, а также реализации новых типов интеллектуальных систем. Пакет распространяется с открытым исходным кодом по открытому лицензионному соглашению (GPL) GNU, являясь, в упрощенном понимании, программным обеспечением. Вывод результатов разработанной программы осуществляется на экран ПК, а также в HTML-формат (изображения сохраняются в формате PNG).

При разработке программного продукта в качестве исходных данных использовались классические хаотические системы малой размерности: система Энона, система Лоренца, система Ресслера (в качестве тестовых систем), а также последовательности случайных чисел – так называемый «белый шум» и синусоида.

Программно были реализованы следующие способы и методы анализа нелинейных систем: метод псевдофазовых и фазовых портретов, определение временной задержки и размерности пространства вложения, расчет показателей Ляпунова и определение времени забывания начальных условий, эволюция фазового объема, анализ аттракторов фазовых траекторий, оценка энтропии Колмогорова, расчет показателя Херста, спектральный анализ. Исследователю предлагается загрузить файл с исходными данными временного ряда, произвести необходимые настройки и оценить полученные данные по вышеперечисленным методам. Система обрисовки изображений оформлена в виде отдельной структурированной объектной модели, благодаря чему было достигнуто увеличение быстродействия системы. Программа обеспечивает пакетный режим обработки исходных данных, что позволяет обрабатывать одновременно несколько файлов. Получено свидетельство о государственной регистрации программы для ЭВМ №2012617166.

Автоматическая обработка хаотических процессов была разделена на ряд этапов. Первый этап – анализ временного ряда. В результате его выполнения определяются размерность пространства вложения (метод ближайших ложных соседей) и временная задержка (метод взаимной информации, метод автокорреляционной функции). Определение параметров вложения обеспечивает максимальную предсказуемость временного ряда. Используя результаты предыдущего этапа, осуществляется идентификация хаотического процесса путем вычисления наибольшего показателя Ляпунова, определение времени забывания начальных условий, эволюции фазового объема, анализ аттракторов фазовых траекторий, вычисление энтропии Колмогорова, показателя Херста, метод псевдофазовых и фазовых портретов.

Результаты анализа ЭЭГ показывают на наличие хаотической детерминируемой составляющей. Рассчитанные характеристические показатели Ляпунова имеют диагностическое значение и подтверждают теорию присутствия хаоса в системе. Для условно здоровых участков сигнала значение характеристического показателя Ляпунова выше, это говорит о большей степени хаотичности сигнала для мозга в условно здоровом состоянии. Исследования различных типов сигналов показали, что чем сложнее аттрактор системы, тем в более спокойном и здоровом состоянии находится «система» (пациент), а при апноэ степень хаотичности снижается. Показатель Ляпунова при нормальном функционировании системы принимает значения внутри некоторого диапазона, характерного для данной системы. Отклонение этого показателя в любую сторону негативно характеризует функционирование системы. Применение аппарата нелинейной динамики может служить мощным средством анализа электрической активности головного

мозга и помочь выработать новые стратегии предсказания времени начала приступа. Отличительной особенностью данного исследования является снижение субъективности расшифровки сигнала, принципиальное улучшение точности и быстродействия.

Литература

1. Ситникова Е.Ю., Короновский А.А., Храмов А.Е. Анализ электрической активности головного мозга при абсанс-эпилепсии: прикладные аспекты нелинейной динамики // Известия вузов «ПНД». Нелинейная динамика и нейронаука. 2011. Т.19, №6. С.173–182.
2. Кузнецов С.П. Динамический хаос. Курс лекций. М.: Физматлит, 2001.
3. Окуловский Ю.С. A model and implementation of universal engine for neural systems. // 9-й международная конференция «Интеллектуальные системы и компьютерные науки», труды конференции, II том. М., 2006. С. 21–24.
4. Конончук Д.О., Окуловский Ю.С. Универсальный пакет поддержки интеллектуальных вычислений GANS // VI Всероссийская межвузовская конференция молодых ученых. Труды конференции. Выпуск 4. Математическое моделирование и программное обеспечение. СПб., 2009. С. 151–157.

ПОДАВЛЕНИЕ ВОЗБУЖДЕНИЯ В АКТИВНОЙ СРЕДЕ С ПОМОЩЬЮ ВНЕШНЕГО ВОЗДЕЙСТВИЯ

И.И. Бастраков, К.А. Гаврилова, С.А. Григорьева, Г.В. Осипов

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлены два новых метода подавления импульса в одномерной и двумерной возбудимых средах с помощью внешнего воздействия. Исследования проводились на модели Зыкова. Определены условия на амплитуду и длительности внешних воздействий, необходимые для подавления возбуждений.

Введение

Модель возбудимой среды – одна из базовых моделей активных сред. В таких средах возможно существование устойчивых самоподдерживающихся волн. Также возбудимые среды интересны тем, что в них могут распространяться волны возбуждения, которые можно рассматривать как один из механизмов связи между различными частями сред. В некоторых случаях существование распространяющихся волн является нежелательным эффектом. Например, самопроизвольный разрыв спиральной волны на несколько волн и их последовательное дробление из-за неоднородности могут приводить к хаотическому пространственно-временному поведению. Такая динамика рассматривается как возможный механизм для начала желудочковой фибрилляции в сердечной мышце. Таким образом, существует необходимость в разработке эффективных методов для инициации, управления и уничтожения волн возбуждения.

В работе предлагается метод, который использует кратковременное импульсное воздействие на среду для уничтожения нестатических состояний в возбудимых средах, таких как распространяющиеся импульсы, одиночная спиральная волна и хаос спиральных волн.

1. Модель

Рассмотрим модель Зыкова:

$$\begin{cases} \dfrac{\partial u}{\partial t} = F(u,v) + D_u \Delta u \\ \dfrac{\partial v}{\partial t} = \varepsilon G(u,v) + e(x,t) \end{cases}, \quad x \in [0,L], \quad e(x,t) = \begin{cases} E_0, t \in \Delta t \\ 0, t \notin \Delta t \end{cases}, \tag{1}$$

где L – параметр определяющий размер среды, Δt – продолжительность внешнего импульса, E_0 – амплитуда импульса, F и G – следующие кусочно-линейные функции:

$$F(u,v) = \begin{cases} -k_1 u - v, u < \delta \\ k_f (u-a), \delta \le u \le b - \delta \\ k_2 (b-u), u > b - \delta \end{cases}, \tag{2}$$

$$G(u,v) = \begin{cases} g + k_g u - v, k_g u - v \ge 0 \\ g + k_\varepsilon (k_g u - v), k_g u - v < 0 \end{cases}. \tag{3}$$

Рассмотрим систему (1) с функциями (2) и (3) при следующих значениях параметров: $a = 5$, $b = 20$, $k_f = 0.4$, $k_g = 1.7$, $\delta = 0.2$, $k_\varepsilon = 6$, $\varepsilon << 1$. Выберем $D_u = 1$ и $L = 35$. k_1, k_2 – определим из непрерывности функции $f(u) = F(u,v) + v$, при $u = \delta$ и $u = b - \delta$:

$$k_1 = -\frac{k_f}{\delta}(\delta - a); k_2 = \frac{k_f}{\delta}(b - \delta - a).$$

2. Подавление возбуждения с помощью слабого внешнего воздействия

Волны переключения, распространяющиеся со скоростями $c_{f,b}$, являются решениями уравнения:

$$-c_{f,b}\frac{du}{d\xi} = F(u, v_{f,b}^{imp}) + D_u\frac{d^2u}{d\xi^2}, \quad (4)$$

где $\xi = x - c_{f,b}t$, c_f – скорость переднего фронта, c_b – скорость заднего фронта.

Граничные условия:

$$u(0) = u_1 \text{ и } u(T) = u_3, \left.\frac{du}{d\xi}\right|_{\xi=0} = 0 \text{ и } \left.\frac{du}{d\xi}\right|_{\xi=T} = 0 \quad (5)$$

для переднего фронта волны.

$$u(0) = u_3 \text{ и } u(T) = u_1, \left.\frac{du}{d\xi}\right|_{\xi=0} = 0 \text{ и } \left.\frac{du}{d\xi}\right|_{\xi=T} = 0 \quad (6)$$

для заднего фронта волны.

Решение для переднего и заднего фронтов волны может быть найдено при малых значениях δ:

$$F(u, v_{f,b}^{imp}) = k_f(u - a) - v_{f,b}^{imp}. \quad (7)$$

Система (4) с одним из граничных условий (5) или (6) является задачей на собственные значения для единственных значений скорости распространения $c_{f,b}$. При этом значения $v_{f,b}^{imp}$ определяют скорости и направления движения переднего и заднего фронтов распространяющегося импульса. Поэтому кратковременное импульсное воздействие может вызывать изменение значения медленной переменной на переднем и заднем фронтах бегущего импульса, что приводит к изменению скоростей распространения фронтов. Распространение переднего и заднего фронтов с различными скоростями может привести к дестабилизации распространяющегося импульса и переходу в состояние равновесия.

Решая краевую задачу (4) – (6), получим выражение для скорости невозмущенного распространяющегося импульса (см. также рис. 1):

$$c_{f,b}^2 = \frac{4k_f \ln^2\left(\frac{b}{a_{f,b}} - 1\right)}{\pi^2 + \ln^2\left(\frac{b}{a_{f,b}} - 1\right)}, \; a_f = b - a - \frac{v_f^{imp}}{k_2}, \; a_b = a - \frac{v_b^{imp}}{k_2}.$$

Продолжительность импульса равна

$$T_{f,b} = \frac{1}{\sqrt{k_f}}\sqrt{\pi^2 + \ln^2\left(\frac{b}{a_{f,b}} - 1\right)},$$

где $a_{f,b}$ имеет смысл порогового значения медленной переменной.

В зависимости от знака значения E_0 можно выделить два случая подавления распространяющегося импульса:

- $E_0 < 0$ – в результате дестабилизации импульса происходит увеличение возбужденной части импульса и уменьшение невозбужденной части импульса – «довозбуждение импульса»;

- $E_0 > 0$ – в результате дестабилизации импульса происходит уменьшение возбужденной части импульса и увеличение невозбужденной части импульса – «гашение импульса».

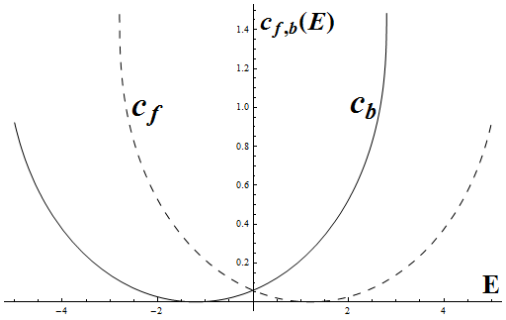


Рис. 1. Скорости c_f переднего и c_b заднего фронтов импульса в зависимости от E

3. Численные эксперименты

Проведены численные эксперименты для различных значений параметров E_0 и Δt. На рис. 2 представлены численные результаты для различных значений амплитуды E_0 при фиксированной длительности приложенного внешнего импульса $\Delta t = 10$. Внешний импульс приложен в момент времени $t = 0$. На рисунках приведены пространственно-временные диаграммы эволюции начального импульса $u(x, 0)$. Темным цветом обозначены возбужденные области, светлым – невозбужденные области. Довозбуждение импульса представлено на рисунке 2(*a–c*). Гашение импульса представлено на рисунке 2(*g–i*). Результаты неудачного подавления представлены на рисунке 2(*d–f*). В этом случае значение импульсного воздействия $E = E_0\Delta t$ не достаточно для дестабилизации начального импульса. Расширение возбужденной части исходного импульса, продолжающееся и после прекращения подачи внешнего воздействия, через некоторое время заканчивается, область возбуждения начинает сужаться, и в итоге начальный импульс восстанавливается.

Таким образом, подавление распространяющегося импульса возможно, если значение E больше некоторой критической величины Ecr^T.

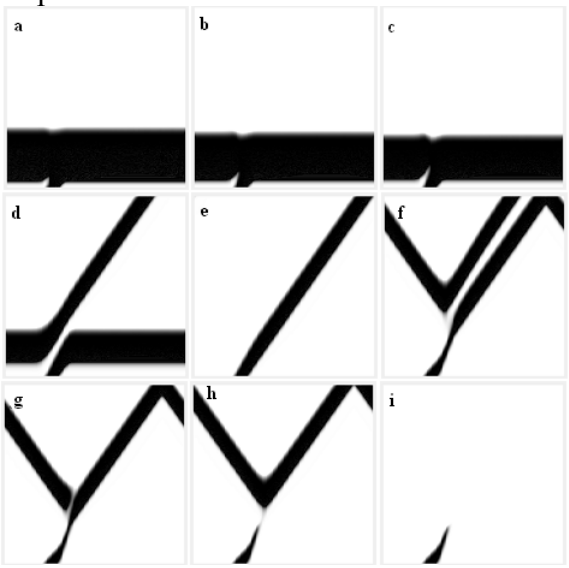


Рис. 2. (a-c) Успешное довозбуждение при $E_0 = -5.5$; $E_0 = -3.5$; $E_0 = -2.5$.
(d-f) Неуспешное довозбуждение при $E_0 = -0.5$; невозмущенная модель Зыкова при $E_0 = 0$; неуспешное гашение при $E_0 = -3$.
(g-i) Успешное гашение при $E_0 = 3.5$; $E_0 = 3.59$; $E_0 = 3.6$

4. Подавление возбуждения в двумерной среде

4.1. Подавление одиночной спиральной волны

Рассмотрим двумерную модель с импульсным внешним воздействием:

$$\begin{cases} c\dfrac{\partial u}{\partial t} = F(u,v) + D_u\left(\dfrac{d^2u}{dx^2} + \dfrac{d^2u}{dy^2}\right), \\ \dfrac{\partial v}{\partial t} = \varepsilon G(u,v) + e(x,t) \end{cases}$$

со свободными граничными условиями:

$$\left.\frac{du}{dx}\right|_{x=0,x=L} = 0 \text{ и } \left.\frac{du}{dy}\right|_{y=0,y=L} = 0.$$

Спиральная волна имеет ширину возбужденной части, которая является относительно постоянной вдали от ядра спиральной волны и от границ среды. Будем предполагать, что ширина возбужденной части волны совпадает с шириной импульса, распространяющегося в одномерной среде с тем же набором параметров. Можно заметить, что эволюция спиральной волны может рассматриваться как движение свободного конца спирали и движение переднего и заднего фронтов спиральной волны. Поэтому внешний импульс, применяемый к одиночной спиральной волне или пространственно-временному беспорядку, приведет к результатам, аналогичным полученным ранее для одномерной среды.

Внешнее воздействие может привести к изменению скорости распространения переднего и заднего фронтов спиральной волны. Таким образом, изменится ширина возбужденной части, что приведет к дестабилизации и уничтожению спиральной волны.

Проведены численные эксперименты для различных значений параметров E_0 и Δt. На рисунке 3 представлены численные результаты для различных значений амплитуды E_0 при фиксированной длительности приложенного внешнего импульса Δt =10. Внешний импульс приложен в момент времени t = 0. На рисунках приведены мгновенные распределения эволюции начального импульса $u(x, y, 0)$. Темным цветом обозначены возбужденные области, светлым – невозбужденные области.

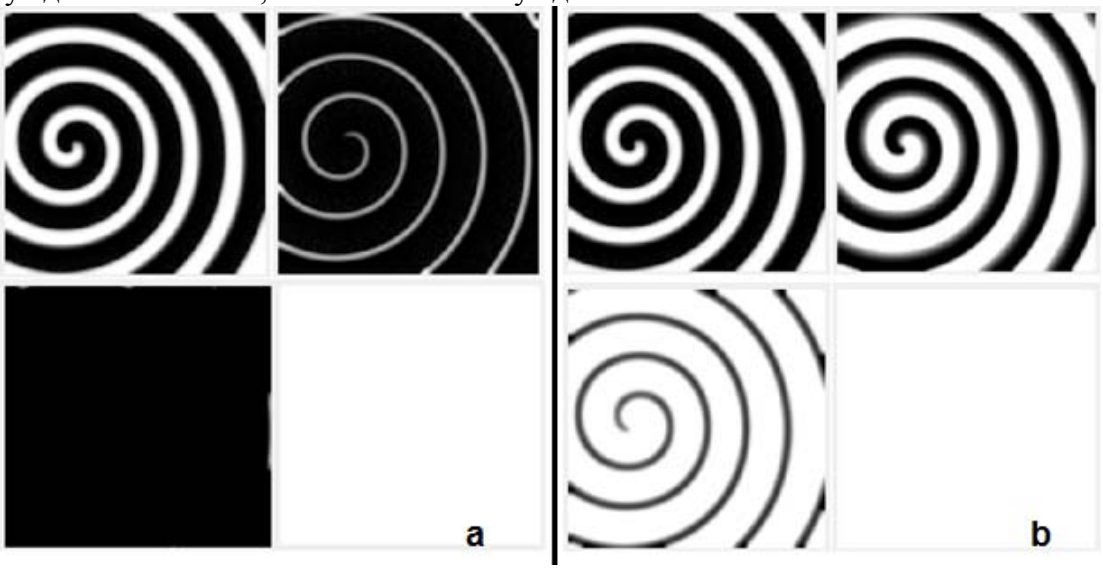


Рис. 3. (a) Успешное довозбуждение при E_0 = –2.5 при временах t = 0, t = 4, t = 10, t = 17. (b) Успешное гашение при E_0 = 5, при временах t = 0, t = 2, t = 4, t = 7.

4.2. Подавление спирально-волнового беспорядка

Проведены численные эксперименты для различных значений параметров E_0 и Δt. На рисунке 4 представлены численные результаты для различных значений амплитуды E_0 при фиксированной длительности приложенного внешнего импульса $\Delta t = 10$. Внешний импульс приложен в момент времени $t = 0$. На рисунках приведены мгновенные распределения эволюции начального импульса $u(x, y, 0)$. Темным цветом обозначены возбужденные области, светлым – невозбужденные области.

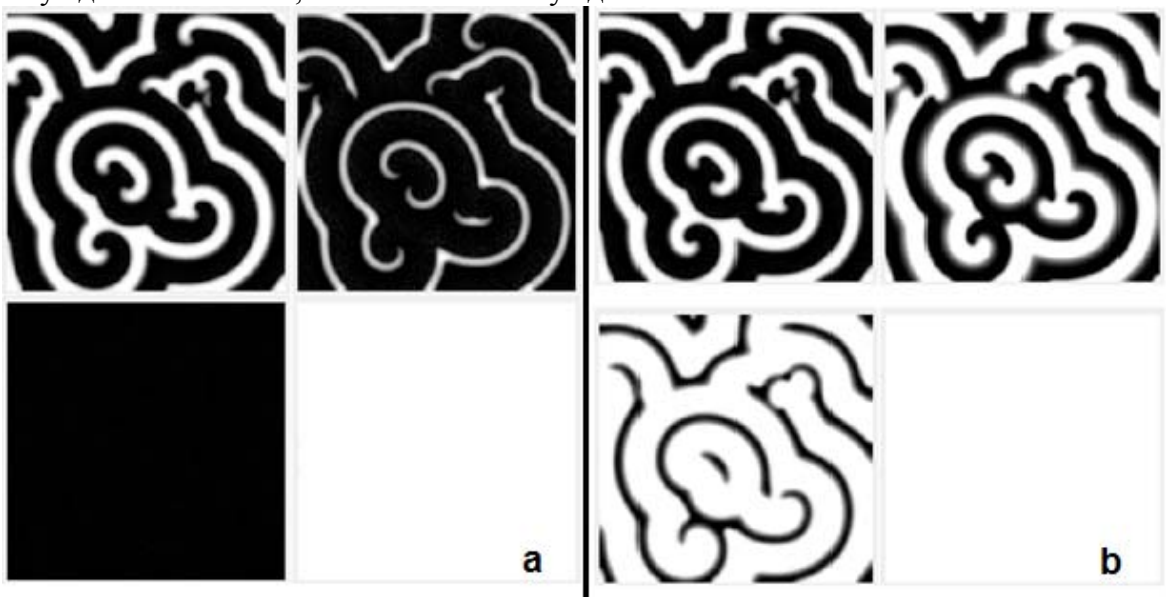


Рис. 4. (a) Успешное довозбуждение при $E_0 = -3$ при временах $t = 0$, $t = 3$, $t = 11$, $t = 16$. (b) Успешное гашение при $E_0 = 4.5$ при временах $t = 0$, $t = 3$, $t = 5$, $t = 10$

Заключение

В данной работе были предложены способы подавления нестатических состояний в возбудимых средах, таких как распространяющиеся импульсы, одиночная спиральная волна, спирально-волновой беспорядок. Было показано, что короткое импульсное воздействие может вызывать изменение значений медленной переменной переднего и заднего фронтов волн и соответствующее изменение скоростей фронтов. Распространение фронтов с различными скоростями ведет к дестабилизации распространяющегося импульса и переходу среды в невозбужденное состояние. Приведенные методы не требуют постоянного во времени применения воздействия. И в случае «довозбуждения», и в случае «гашения» внешнее воздействие не приводит явно к возбуждению или невозбуждению всей среды, а влияет только на передний и задний фронты. Поэтому внешнее воздействие может быть приложено только к небольшому участку среды, содержащему область возбуждения.

Литература

1. Востриков В.А., Горбунов Б.Б. Сравнение биполярных импульсов, генерируемых внешними дефибрилляторами // Клиническая анестезиология и реаниматология. 2006. Т.3, № 6.
2. Zykov V.S., Mikhailov A.S. // Phys. Rev. Lett. 78. 1997. 3398.
3. Андронов А.А., Витт А.А., Хайкин С.Э. Теория колебаний. – М.: Наука. Главная редакция физико-математической литературы, 1981. – 568 с.
4. Лоскутов А.Ю., Михайлов А.С. Введение в синергетику. – М.: Наука, 1990. – 272 с.

ПОДАВЛЕНИЕ ПРОСТРАНСТВЕННО-ВРЕМЕННОГО БЕСПОРЯДКА В ВОЗБУДИМОЙ СРЕДЕ С ПОМОЩЬЮ ЛОКАЛИЗОВАННОГО СЛАБОГО ВНЕШНЕГО ВОЗДЕЙСТВИЯ В МОДЕЛИ ФИТЦХЬЮ–НАГУМО

И.И. Бастраков, Г.В. Осипов

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлены результаты исследования способов подавления спирального беспорядка в возбудимых средах с помощью локального внешнего воздействия. Исследования проводились на модели Фитцхью-Нагумо, которая при некотором наборе параметров может быть интерпретирована как модель сердечной клетки. Было предложено воздействие, изменяющее параметр среды *a*, имеющее ряд недостатков. Была предложена периодическая кусочно-линейная функция, обеспечивающая бифазное локальное слабое внешнее воздействие, вытесняющее пространственно-временной беспорядок в решетке слабо неидентичных связанных элементов.

Введение

Согласно одной из гипотез, сердечные аритмии ассоциируются с существованием в фазовом пространстве соответствующей динамической системы пространственно-временного спирального беспорядка. Традиционный способ подавления – мощный электрический импульс, который может нарушить целостность сердечной ткани и ее функциональные способности. Поэтому важно найти решение задачи борьбы с аритмией с помощью слабого внешнего воздействия. В работе предлагается такой метод, реализованный на двумерной среде возбудимых элементов системы Фитцхью–Нагумо.

Исследование

Рассмотрим систему Фитцхью–Нагумо в виде:

$$\begin{cases} \dot{x} = -y + x - x^3/3 + I(t), \\ \dot{y} = \varepsilon(x + a). \end{cases}$$

При $I(t) = t\,(b\text{-}a)\,/\,\varepsilon$ система примет вид:

$$\begin{cases} \dot{x} = -z + x - x^3/3 \\ \dot{z} = \varepsilon(x + b) \end{cases}, \quad \text{где } z = y - \varepsilon \dot{I}(t).$$

Таким образом, подача линейного воздействия в первое уравнение эквивалентна изменению значения параметра *a*. Поэтому можно сделать элемент автоколебательным с нужной частотой.

У данного подхода есть два существенных недостатка. Первый из них заключается в том, что функция является линейной по времени, следовательно, ее амплитуда увеличивается, что не позволяет считать такое воздействие слабым. Второй недостаток заключается в том, что таким воздействием мы можем достичь только некой конечной частоты, то есть существует такая область значений параметра *a*, что воздействием такого типа мы не сможем добиться возрастания частоты окружающей среды, достаточного для вытеснения спиральных волн.

Поэтому рассмотрим новую функцию воздействия, имитирующую связь с элементом, на который подается линейное воздействие. Если такую функцию вычислить зара-

нее, то подавать линейное воздействие в систему уже не потребуется. Вид такой функции для различных значений параметров совпадает и представлен на рис. 1. Для её аппроксимации рассмотрим класс периодических кусочно-линейных функций с параметрами, зависящими только от параметров среды:

$$F(t)=\begin{cases} B(1-|t/t_1-1|), \text{если } (t \bmod p)\in[0;2t_1] \\ B(|(t-2t_1-t_2)/t_2-1|-1), \text{если } (t \bmod p)\in[2t_1+t_2;2t_1+t_2+2t_3]. \\ 0, \text{иначе} \end{cases}$$

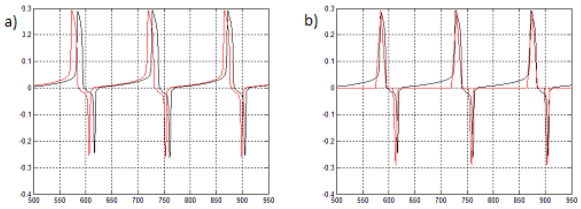


Рис. 1. Зависимость функции воздействия от времени

Численно были получены оптимальные значения коэффициентов при различных значениях параметров среды, и аппроксимированы следующими кусочно-непрерывными функциями:

$$p=\begin{cases} 1.2(2.55+|a|)/(0.01+\varepsilon), \text{если } |a|<1, \\ 1.2(2.55+2|a|)/(0.01+\varepsilon), \text{если } |a|\geq 1. \end{cases}$$

$$B=1/30d;\ t_1=5(2.55+a);\ t_2=0.3/(1+\varepsilon);\ t_3=5(2.55-a).$$

При этом обе обозначенные проблемы решены: амплитуда малая и не нарастает с течением времени и частота не ограничена максимальной частотой элемента системы Фитцхью–Нагумо.

Эффективность такого воздействия представлена на рис. 2. Размер решетки 150x150, воздействие подается в область 10x10 при параметрах системы: a = 1.1, ε = 0.02, d = 0.1.

Таким образом, было подобрано локальное слабое внешнее воздействие, вытесняющее пространственно-временной беспорядок в решетке слабо неидентичных связанных элементов, описываемых системой Фитцхью–Нагумо.

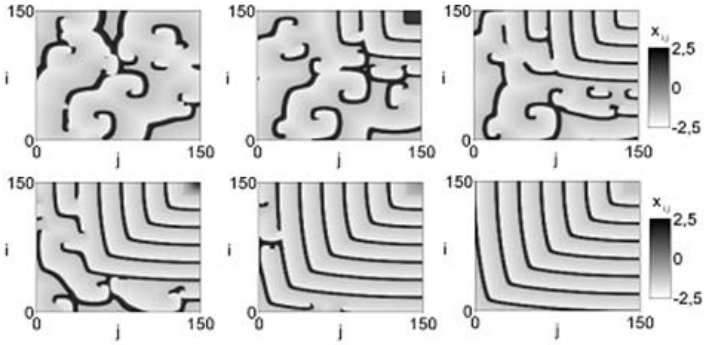


Рис. 2. Изменение мгновенного распределения $x_{i,j}$ во времени

Литература

1. Востриков В.А., Горбунов Б.Б. Сравнение биполярных импульсов, генерируемых внешними дефибрилляторами // Клиническая анестезиология и реаниматология. 2006. Т.3, № 6.

2. FitzHugh R. Impulses and physiological states in theoretical models of nerve membrane. // Biophysical J. 1961. 1:445-466.
3. Кузнецов А.П., Емельянова Ю.П., Сатаев И.Р., Тюрюкина Л.В. Синхронизация в задачах. – Саратов: ООО «Издательский центр «Наука», 2010. – 256 с.

МОДЕЛИРОВАНИЕ ПЛАЗМЫ МЕТОДОМ ЧАСТИЦ В ЯЧЕЙКАХ НА УСКОРИТЕЛЯХ INTEL XEON PHI В КОДЕ PICADOR

С.И. Бастраков[1], А.А. Гоносков[2], Е.С. Ефименко[2], А.С. Малышев[1], И.Б. Мееров[1], И.А. Сурмин[1], М.А. Ширяев[1]

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

Рассмотрена эффективная реализация метода частиц в ячейках для моделирования плазмы на CPU и ускорителях Xeon Phi. Предлагается схема хранения и обработки частиц, основанная на принципе локальности. Рассматриваются вопросы векторизации операций с частицами, предлагается схема частичной векторизации. Суммарное ускорение от предлагаемых оптимизаций по сравнению с базовой версией составляет от 4 до 6 раз. Оптимизированная версия достигает до 33% от пиковой производительности на CPU и 6% от пиковой производительности на Xeon Phi.

Введение

Моделирование плазмы методом частиц в ячейках (Particle-in-Cell, PIC) является одной из востребованных областей вычислительной физики. Решение актуальных задач часто требует моделирования плазмы с использованием тысяч процессоров. Это обуславливает интерес к высокопроизводительной реализации метода частиц в ячейках на традиционных и гетерогенных кластерных системах.

1. Моделирование плазмы на гетерогенных системах

В течение последних лет наблюдается растущий интерес к эффективной реализации метода частиц в ячейках на GPU [1]. Главной трудностью является обеспечение высокой степени параллелизма и эффективного паттерна использования глобальной и «разделяемой» (shared) памяти. При этом значительную сложность вызывает использование специализированных технологий и средств программирования.

Появившиеся в 2012 году ускорители Intel Xeon Phi обладают близкой к GPU производительностью и пропускной способностью памяти в сочетании с традиционными технологиями параллельного программирования: MPI, OpenMP, Cilk Plus, TBB и др. Это делает Xeon Phi перспективной платформой для реализации метода частиц в ячейках. Производительность приложений на Xeon Phi существенно зависит от двух факторов: эффективное использование 512-битных векторных инструкций, хорошая масштабируемость до 60 ядер на общей памяти.

2. Оптимизация реализации метода частиц в ячейках в коде PICADOR

С 2010 года коллективом ННГУ и ИПФ РАН разрабатывается программный комплекс PICADOR [2, 3], ориентированный на использование гетерогенных кластерных систем с использованием технологий MPI, OpenMP, CUDA. Данный код был портирован для ускорителей Xeon Phi с использованием нативного режима.

Представляются техники оптимизации реализации метода частиц в ячейках и их влияние на производительность на CPU и Xeon Phi. Предлагается схема хранения и обработки данных на основе принципа локальности, использование данной схемы дает ускорение от 2.5 до 4 раз (на различных устройствах) по сравнению с базовой реализацией. Рассматриваются вопросы векторизации операций обработки частиц. Предлага-

ется подход, позволяющий выполнить частичную векторизацию операции с частицами, что приводит к ускорению от 1.5 до 2 раз.

3. Результаты

Суммарное ускорение от оптимизации составляет от 4 до 6 раз. Для оптимизированной версии на различных устройствах получены следующие показатели производительности в процентах от пиковой для данного устройства: 33% на 4-ядерном процессоре Intel Xeon 5150, 21% на 8-ядерном Intel Xeon E2690, 6% от пика на 61-ядерном Intel Xeon Phi 7110X. Производительность на Xeon Phi в 2 раза превосходит производительность на 8-ядерном Intel Xeon E2690. Исследуется зависимость производительности на Xeon Phi от количества потоков на ядро и конфигурации запуска с использованием MPI и OpenMP.

Работа выполнена в лаборатории ННГУ–Intel «Информационные технологии» при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», соглашение № 14.B37.21.0393, при поддержке Совета по грантам Президента Российской Федерации (грант № НШ-1960.2012.9).

Литература

1. Burau H., et al. PIConGPU: A Fully Relativistic Particle-in-Cell Code for a GPU Cluster // IEEE Transactions on Plasma Science. 38(10), 2010. 2831-2839.
2. 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. 3. 2012. 474-479.
3. Bastrakov S., Meyerov I., Gergel V., Gonoskov A., Gorshkov A., Efimenko E., Ivanchenko M., Kirillin M., Malova A., Osipov G., Petrov V., Surmin I., Vildemanov A. High performance computing in biomedical applications // Procedia Computer Science. 18. 2013.

ФИЛЬТРАЦИЯ ДВУХФАЗНОЙ ЖИДКОСТИ В НЕОДНОРОДНОЙ СРЕДЕ НА КОМПЬЮТЕРАХ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ

Е.В. Бервено, А.А. Калинкин, Ю.М. Лаевский

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассматривается математическая модель фильтрации двухфазной жидкости и её программная реализация на кластерах, состоящих из сотен узлов, с использованием MPI-технологии. Переход к сеточной задаче осуществлен с помощью смешанного метода конечных элементов. Приведенный алгоритм для программной реализации задачи обладает высокой масштабируемостью и эффективностью с точки зрения операций и обмена данными на многопроцессорных системах. При проведении ряда тестов были получены численные результаты, демонстрирующие значительное варьирование времени прорыва воды в добывающие скважины, в зависимости от местоположения неоднородностей.

Введение

На сегодняшний день методы математического моделирования широко используются в практике проектирования и оптимизации разработки месторождений и решения задач фильтрации. Создание моделей, адекватно описывающих строение пластов, а также происходящие в них фильтрационные процессы, является актуальной задачей.

Ранее в рамках данной тематики авторами статей [1, 2] исследовались модели фильтрации двухфазной несжимаемой жидкости в однородной среде с различным расположением скважин. Данное исследование является продолжением этого цикла работ, и теперь акцент ставится на изучение того, как неоднородности в почве могут влиять на процесс фильтрации.

1. Модель двухфазной фильтрации

Математическая постановка модели включает в себя закон сохранения массы компонент двухфазной несжимаемой жидкости и закон Дарси:

$$\frac{1}{k(s)} v + \nabla \psi = G(s),$$
$$\nabla v = 0,$$
$$\frac{1}{k(s)} w = \nabla \sigma(s),$$
$$v_2 - \frac{k_2(s)}{k(s)}(v - w) = -k_2(s) G(s),$$
$$m \frac{\partial s}{\partial t} + \nabla v_2 = 0,$$

где v_i – векторы скоростей фильтрации фаз, $v = v_1 + v_2$, ψ – обобщенное давление, k_i – проницаемость фаз, $k = k_1 + k_2$, s – насыщенность второй фазы, $G(s)$ – вектор гравитации, m – пористость. Здесь и далее подстрочный индекс i означает номер фазы, где $i = 1$ соответствует вытесняемой фазе (нефть), $i = 2$ – вытесняющей фазе (вода).

Приведённые выше соотношения характеризуют процесс фильтрации в однородных средах (рис. 1). Система замыкается путем задания условий непротекания на гра-

нице области и задания удельного потока на границах нагнетательной и добывающей скважин.

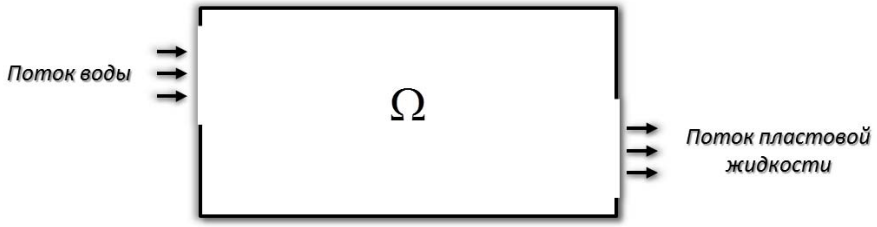


Рис. 1. Процесс фильтрации

2. Программная реализация задачи

Задача была реализована с применением высокопроизводительных кластерных вычислений, необходимость чего обусловлена сложностью и большим объемом задачи. Вычисления производились в Сибирском суперкомпьютерном центре (ССКЦ) на кластере НКС-30Т, содержащем 60 вычислительных узлов, каждый из которых содержит 2 процессора Intel®Xeon®E5540 с 8Gb RAM. В итоге алгоритм продемонстрировал эффективную работу на компьютерах с распределенной памятью и хорошую масштабируемость до 256 процессов.

Для построения дискретной модели используется смешанный метод конечных элементов. Скалярные функции (давление и насыщенность) ищутся в пространстве кусочно-постоянных функций, а векторные поля скоростей аппроксимируются элементами Равьяра-Тома минимальной степени. В качестве разностной схемы по времени используется явная схема типа предиктор–корректор, что позволяет свести распараллеливание к параллельной реализации вычислений правых частей. Для эффективного решения седловой задачи согласно [2] используется метод сопряженных градиентов для дополнения Шура с переобуславливателем, допускающим разделение переменных.

Основным преимуществом этого алгоритма является хорошая масштабируемость на компьютерах с распределенной памятью: каждый процесс делает работу независимо от других, и все процессы обмениваются данными с помощью процедуры MPI_Alltoall. Распределение данных между процессами происходит, как указано на рисунке 2.

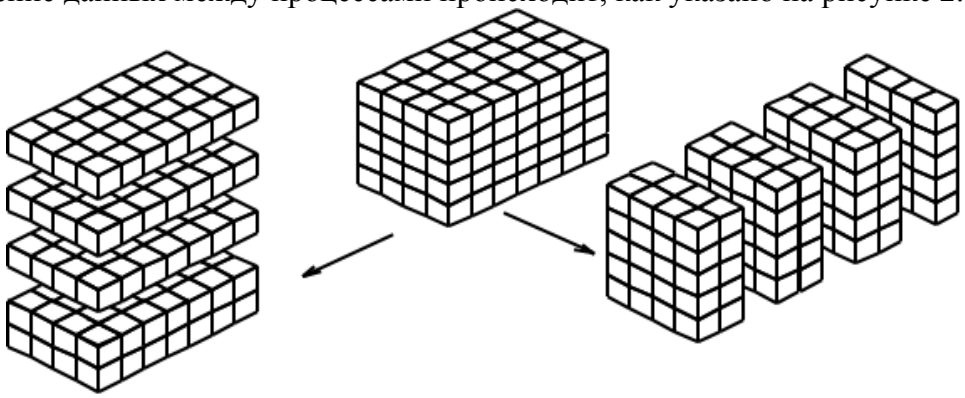


Рис. 2. Распределение данных между процессами

Для решения задачи на каждом шаге необходимо сделать лишь несколько итераций переобусловленного метода сопряженных градиентов. В процессе реализации переобуславливателя требуется выполнить огромное количество дискретных разложений Фурье малой размерности. Для ее реализации мы используем FFT функциональность HPC библиотеки Intel® MKL [7]. В итоге, благодаря тому, что каждый рабочий массив делится поровну между процессами, мы смогли решать задачи достаточно большого раз-

мера, которые не могут быть рассчитаны на последовательных машинах ввиду естественных ограничений доступной памяти.

3. Результаты численных экспериментов

С целью изучения движения водяного фронта был проведен ряд вычислительных экспериментов для модели нефтяного пласта, где неоднородные блоки задавались геометрически при помощи параметров пористости и проницаемости.

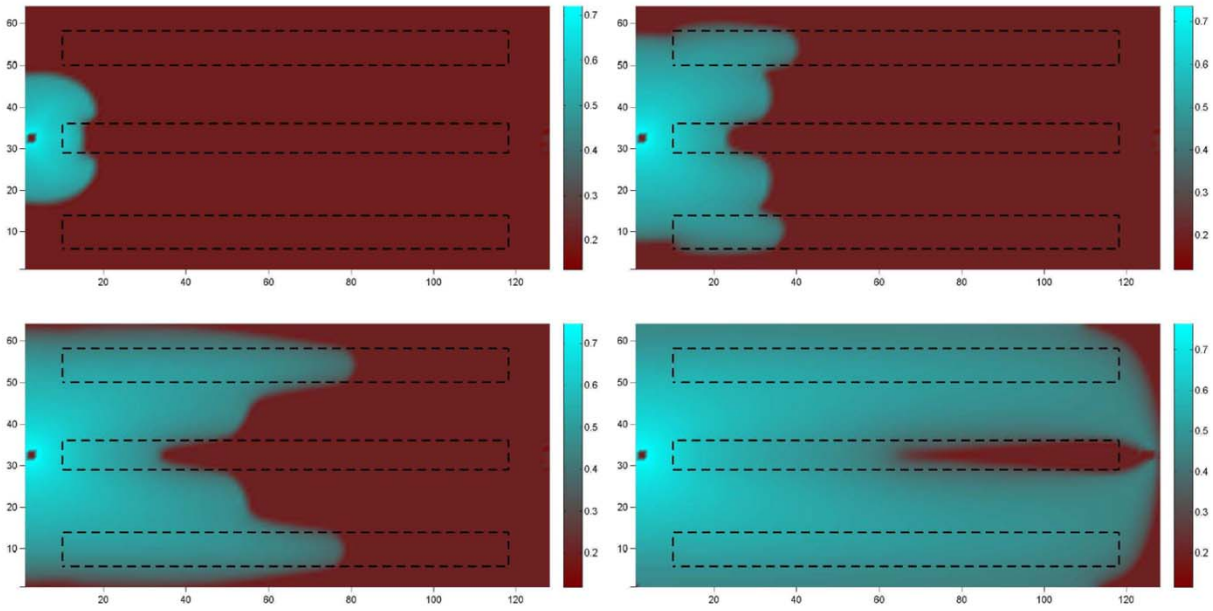


Рис. 3. Фильтрация с неоднородной пористостью m

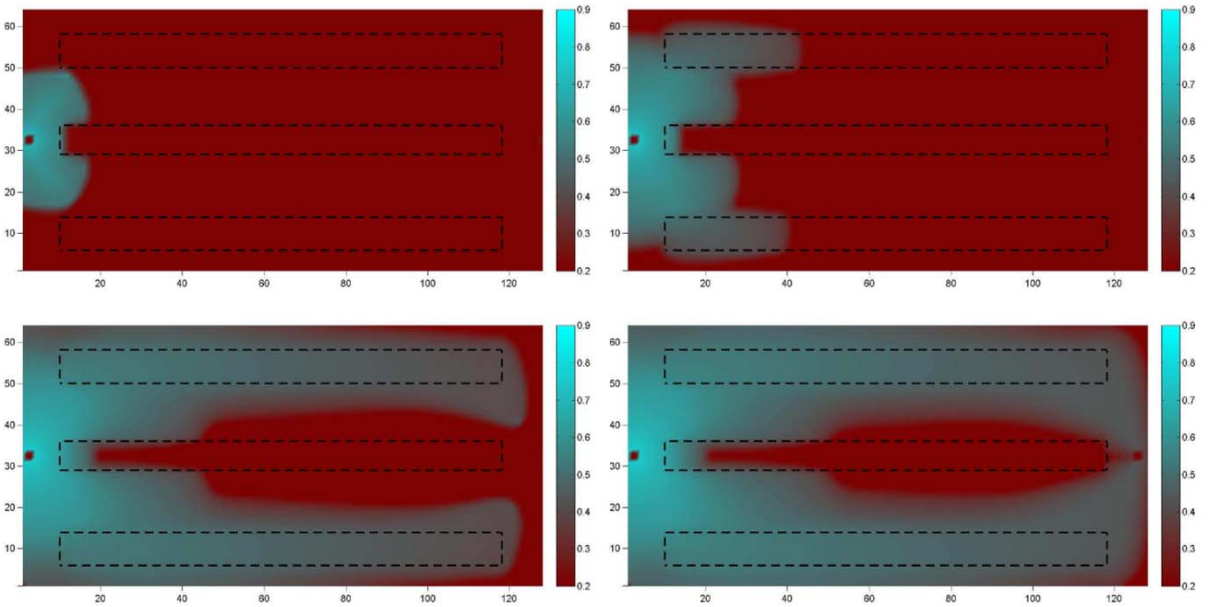


Рис. 4. Фильтрация с неоднородной проницаемостью k_0

На рисунке 3 приведена визуализация процесса фильтрации нефти водой в случае включения в нефтеносный слой блоков с различными параметрами пористости m. В верхнем и нижнем пунктирных блоках значение пористости m =0,1, в среднем значение m =0,9, а в остальном объеме m =0,375 (соответствует пористости нефти).

На рисунке 4 изображены результаты эксперимента, где в качестве варьируемого параметра неоднородности среды задается абсолютная проницаемость k_0. В верхнем и нижнем пунктирных блоках значение проницаемости $k_0 = 3{,}06 \times 10^{-11}$, в среднем значение $k_0 = 3{,}06 \times 10^{-13}$, а в остальном объеме $k_0 = 3{,}06 \times 10^{-12}$.

Интересно, что если задавать неоднородности разными способами –используя либо параметр пористости, либо параметр проницаемости, то при одном и том же положении неоднородностей в пластах скорость вытеснения нефти водой во всём объеме будет значительно отличаться. Это объясняется тем, что проницаемость входит в уравнение множителем при членах первого порядка малости, а изменения пористости – с множителем порядка единицы. Зависимость проницаемости от давления может быть существенной для процессов, происходящих в призабойной зоне, где велики перепады давления, или для весьма длительных процессов. Таким образом проиллюстрирована важность точных моделей и исходных данных о структуре нефтяного коллектора.

Полученный алгоритм эффективно работает на компьютерах с распределенной памятью. Результаты работы программы на различных последовательностях сеток представлены в таблице 1. Алгоритм не может работать на большем количестве процессов, чем размерность сетки, поэтому некоторые ячейки оставлены пустыми. Строки соответствуют различным размерам сетки, столбцы – числу процессов. Коэффициент ускорения представлен в каждой ячейке как отношение времени работы на 1 процессе к времени работы на p процессах.

Таблица 1. Результаты масштабируемости MPI-версии

p / $N \times M$	2	4	8	16	32	64	128	256
32x4	1,47	1,94	-	-	-	-	-	-
64x8	1,79	3,40	4,39	-	-	-	-	-
128x16	1,61	3,06	5,99	8,63	-	-	-	-
256x32	1,98	3,36	6,83	10,38	15,08	-	-	-
512x64	2,07	4,12	4,72	8,65	16,36	20,71	-	-
1024x128	1,78	3,54	7,09	13,84	24,01	34,04	43,31	-
2048x256	2,05	3,44	6,06	12,19	24,53	47,85	67,94	82,98

Как видно из результатов, «одноядерный» MPI более эффективен на небольшом числе процессов, когда вычислительная нагрузка достаточно велика и обмен данными не оказывает существенного влияния. Распараллеливание с использованием ядер с разделенной памятью более эффективно на большом количестве процессов. С ростом числа процессов коэффициент ускорения будет расти настолько быстро, насколько быстро пересылаются данные.

Заключение

Одной из главных целей данного исследования являлось изучение влияния введения неоднородностей на процесс двухфазной фильтрации. Благодаря построенной модели и проведенным в этом направлении экспериментам выявлена необходимость работы над адекватным заданием среды реального нефтяного пласта, поскольку алгоритм демонстрирует зависимость движения фронта воды от изменения параметров среды. Следующим этапом нашего исследования планируется построение модели фильтрации в трещиновато-пористой среде, что сделает доступным более корректное воспроизведение процесса.

Проведен цикл исследований по разработке и реализации программного комплекса, созданию вычислительной среды и разработке методики распараллеливания трехмерной задачи для компьютерных систем, удовлетворяющих кластерной структуре, с использованием MPI-технологии.

Полученный в итоге комплекс программ для трехмерного гидродинамического моделирования резервуаров нефти позволяет проводить инженерный анализ нефтяных месторождений с помощью современных вычислительных технологий и визуализацию результатов высокопроизводительных кластерных вычислений.

Работа поддержана проектами РФФИ №13-01-00019 и №12-01-31046.

Литература

1. Laevsky Yu.M., Popov P.E., Kalinkin A.A. Simulation of two-phase fluid filtration by mixed finite element method // Matem. Mod. 2010. Vol. 22. N. 3. P. 74-90.
2. Popov P.E., Kalinkin A.A. The method of separation of variables in a problem with a saddle point // Russian J. Numer. Anal. Math. Model. 2008. Vol. 23. N. 1. P. 97-106.
3. Баренблатт Г.И., Ентов В.М., Рыжик В.М. Движение жидкостей и газов в природных пластах. М.: Недра, 1984. С. 104-112.
4. Berveno E.V. Simulation of two-phase fluid filtration with nonuniform media on clusters (в печати).
5. Brezi F. and Fortin M. Mixed and Hybrid Finite Element Methods / New York: Springer-Verlag, 1991.
6. Демидов Г.В., Новиков Е.А. Экономичный алгоритм интегрирования нежестких систем обыкновенных дифференциальных уравнений. // Численные методы в математической физике. Новосибирск: ВЦ СО СССР. 1979. С. 69-83.
7. http://software.intel.com/en-en/articles/intel-mkl/.

МУЛЬТИАГЕНТНОСТЬ В КЛАСТЕРНЫХ СИСТЕМАХ

А.В. Березин

Нижегородский государственный технический университет им. Р.Е. Алексеева

Снижения нагрузки на ресурсы компьютера можно достигнуть различными способами, например, распараллеливанием программ и использованием многопоточности, но они оба приводят к тому, что одна задача разделяется на ряд более мелких. Таким образом, можно использовать и мультиагентный подход.

Мультиагентность позволяет использовать меньше ресурсов компьютера при возрастании эффективности и уменьшении количества операций на один компьютер (для запуска одного агента используется один компьютер). Таким образом, для решения мультиагентной задачи можно будет использовать кластерные системы.

Схема взаимодействия агентов следующая: система управления агентами порождает агента, настраивает его необходимым образом, затем следит за его жизненным циклом, по окончании которого уничтожает агента, высвобождая таким образом ресурсы системы. Описанная схема взаимодействия применена в решении задачи мониторинга сетевого оборудования, поэтому уточним ее следующим образом: система управления агентами порождает агента–сканера сети, который следит за наличием в ней сетевого оборудования: маршрутизаторов, сетевых принтеров, различных серверов (рабочие станции игнорируются). В случае если появилось оборудование, не найденное в базе данных устройств, порождается агент для этого оборудования. При возникновении ситуации, когда не найдено оборудование, присутствующее в базе данных устройств, система управления агентами посылает административное оповещение о непредвиденной ситуации сетевому инженеру с описанием возможных причин (используется база знаний) и удаляет агента.

В программе мониторинга другим компонентом, требующим больших вычислительных мощностей, является база данных мониторинга. В зависимости от выбранных параметров мониторинга (SNMP OID – object identifier), для сбора которых используется свободно распространяемый пакет утилит SNMP-Net, таких как загрузка процессора, количество переданных по сети пакетов и данных, объем занимаемой памяти и др., а также информация, снимаемая с температурных датчиков, размер базы данных мониторируемого оборудования может колебаться от 1 до 40 Гб в месяц, что как раз и требует распределенности ее хранения и обработки. Все эти данные необходимо уметь собирать, обрабатывать, анализировать и строить на их основе графики и отчеты, которые помогли бы сетевому инженеру понять, в чем проблема, а в идеале – решить ее с помощью полученных рекомендаций. В качестве СУБД используется Microsoft SQL Server, который также имеет встроенные функции бизнес-аналитики.

В итоге получается, что системы мониторинга крупных сетей требуют также и больших вычислительных мощностей, однако для малых и средних сетей достаточно использовать несколько современных компьютеров.

ОПЕРАТИВНАЯ СИСТЕМА ПРОГНОЗА ПОГОДЫ COSMO-RU И ИССЛЕДОВАНИЯ ЕЁ МАСШТАБИРУЕМОСТИ

Д.В. Блинов, М.А. Никитин, Г.С. Ривин

Гидрометеорологический научно-исследовательский центр Российской Федерации

Рассматривается оперативная система прогноза погоды COSMO-Ru для ограниченной территории и её реализация на вычислительном кластере с архитектурой «РСК Торнадо». Представлены результаты численных экспериментов с различным количеством используемых ядер для оценки свойств масштабируемости кода модели.

Одна из областей, в которых широко применяются суперкомпьютерные технологии, – численный гидродинамический прогноз погоды. Современные модели прогноза погоды представляют собой сложный программный продукт, насчитывающий десятки тысяч строк исходного кода.

Прогноз погоды необходимо передать потребителю вскоре после срока наблюдения, к четко заданному моменту времени. Это требует эффективной оптимизации технологической линии численного прогноза погоды, которая включает следующие этапы: передача информации (с метеорологических и аэрологических станций, судов, буёв, самолётов и т.п.), подготовка начальных и граничных данных в узлах сетки, решение численного прогноза на суперкомпьютере, подготовка информации в нужном для потребителя виде (включая визуализацию) и ее распространение.

В данной работе рассматривается система прогноза погоды COSMO-Ru для ограниченной территории, работающая в оперативном режиме на суперкомпьютерах Главного вычислительного центра Росгидромета. Аббревиатура COSMO расшифровывается как COnsortium for Small-scale MOdelling – Консорциум по мезомасштабному моделированию. В настоящий момент в консорциум входят метеорологические организации семи стран. Федеральная служба по гидрометеорологии и мониторингу окружающей среды (Росгидромет) входит в число этих организаций.

Одним из компонентов системы прогноза погоды является негидростатическая совместная модель атмосферы и деятельного слоя почвы COSMO. Первая версия этой модели была создана Немецкой службой погоды (Deutscher Wetterdienst, DWD). Модель COSMO основана на решении системы уравнений гидро- и термодинамики, описывающей сжимаемый поток во влажной атмосфере. Расчёт производится на географической сетке со сдвинутыми полюсами, что дает возможность провести «экватор» через центр области интегрирования. Благодаря этому расстояние между соседними узлами по меридиану и параллели равномерное для широт, меньших 20°. В модели используются параметризации различных физических процессов. Кроме самой модели, система прогноза погоды COSMO-Ru включает в себя дополнительные компоненты: усвоение данных и инструменты пре- и постпроцессинга.

Система прогноза погоды COSMO-Ru введена в оперативную эксплуатацию в 2011 г. в ФГБУ «Гидрометцентр России». Регулярно, четыре раза в сутки (в 0, 6, 12 и 18 UTC) считаются модели для большей части Европы и всей европейской территории России (шаг сетки 7 км), Южного и Центрального федерального округов (шаг сетки 2,2 км) и Западной Сибири (шаг сетки 14 км). В настоящее время внедряется в оперативную практику модель COSMO-Ru ENA (Europe–North Asia) с областью решения, включающей территорию Европы и Северной Азии (шаг сетки 13 км).

В Главном вычислительном центре Росгидромета существуют два основных вычислительных кластера – «Альтикс» и «Торнадо». Модель ENA может считаться на обоих кластерах, но в настоящее время она считается на «Торнадо». Этот кластер обладает пиковой производительностью 36 Тфлопс и основан на архитектуре «РСК Торнадо», с применением энергоэффективного жидкостного охлаждения, процессоров Intel® Xeon® E5–2600 и серверных плат Intel® S2600JF.

На кластере «Торнадо» расположено 96 узлов, каждый узел содержит два восьмиядерных процессора. Таким образом, общее количество ядер составляет 1536. Модель ENA считается на сетке с шагом 13 км, количество узлов сетки – 1500 по параллели и 1000 по меридиану. Для оперативных расчетов используется 288 ядер – 18 по широте и 16 по долготе. Такое количество ядер вызвано необходимостью параллельного расчета прогнозов погоды одновременно для 4 вышеперечисленных территорий, для каждой территории выделяется 288 ядер. Распараллеливание вычислений в модели проводится с помощью пакета MPI.

В докладе будет дано описание системы COSMO-Ru, трехлетнего опыта применения суперкомпьютеров для нахождения численных прогнозов погоды с помощью этой системы, а также представлены результаты численных экспериментов с различным количеством используемых ядер для нахождения оптимального разбиения области решения на подобласти и оценки свойств масштабируемости кода модели.

Литература

1. Вильфанд Р.М., Ривин Г.С., Розинкина И.А. Мезомасштабный краткосрочный региональный прогноз погоды в Гидрометцентре России на примере COSMO-RU // Метеорология и гидрология. 2010. №1. С. 5-17.
2. Вильфанд Р.М., Ривин Г.С., Розинкина И.А. Система COSMO-RU негидростатического мезомасштабного краткосрочного прогноза погоды Гидрометцентра России: первый этап реализации и развития // Метеорология и гидрология. 2010. №8. С. 6-20.
3. Ривин Г.С., Розинкина И.А. (ред.). Гидрометеорологические прогнозы // Труды Гидрометцентра России. 2011. Вып. 346. 188 с.
4. Ривин Г.С. Современные системы мезомасштабного прогноза погоды: состояние и перспективы // В кн.: «80 лет Гидрометцентру России». М.: Триада ЛТД, 2010. С. 82-93.

ОБ ОДНОЙ РЕАЛИЗАЦИИ ТРАССИРОВКИ ПУТЕЙ ДЛЯ ГРАФИЧЕСКОГО ПРОЦЕССОРА

Д.К. Боголепов[1], Д.Я. Ульянов[2], В.Е. Турлапов[1]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]Нижегородский государственный технический университет им. Р.Е. Алексеева
E-mail: denisbogol@gmail.com, danila-ulyanov@ya.ru, vadim.turlapov@gmail.com

Предложен оптимизированный метод трассировки путей, дополненный концепцией регуляризации. Для комбинирования вкладов различных путей в одну несмещенную оценку предложена эффективная реализация многократной выборки по значимости.

Введение

Методы глобального освещения широко применяются в компьютерной графике, в том числе в научной и медицинской визуализации для отображения данных, полученных в ходе инженерных расчетов, оптических экспериментов, компьютерной или магнитно-резонансной томографии. В данных областях часто возникают ситуации, в которых расчет освещенности традиционными несмещенными методами затруднен. Типичными примерами служат задачи моделирования сложных материалов, оптических систем и конфигураций источников света, включающих в себя лазеры и светодиоды. Указанные типы источников характеризуются малой площадью излучающей поверхности (которая может сочетаться с узконаправленным потоком излучения), что позволяет их моделировать в виде точечных (направленных) источников света. Другой типичный пример затрудненного расчета освещенности – источники света, закрытые прозрачной оболочкой (абажур лампы). Наконец, определенные трудности вызывают сцены с зеркальными или прозрачными объектами, особенно в сочетании с источниками света малой площади. Примером может служить стеклянный лист на столе, который освещается точечным источником. Если при этом используется модель камеры с точечной диафрагмой (стеноп), то традиционные несмещенные методы (включая двунаправленную трассировку путей и Metropolis Light Transport) в принципе не способны сформировать изображение поверхности стола под слоем стекла.

Указанные выше проблемы расчета освещенности связаны с особенностями (сингулярностями) подынтегрального выражения в уравнении визуализации. Хорошо известно, что алгоритм BPT недостаточно эффективно обрабатывает пути переноса излучения, которые содержат SDS-подпоследовательности вершин (через S и D обозначены вершины пути, в которых поверхность обладает соответственно зеркальными и диффузными рассеивающими свойствами) [4]. Причина этой проблемы кроется в техниках построения SDS-путей, которые реализуются в методе BPT. Пути данного класса характеризуются малой плотностью вероятности, которая может даже обращаться в ноль (если используются точечные источники света и камера с точечной диафрагмой). Указанные особенности алгоритма принято именовать проблемой недостаточного числа техник построения путей [2].

В настоящей работе концепция *регуляризации*, предложенная в статье [1], применяется для обработки «сложных» путей с SDS-фрагментами, вклад которых проблематично оценить традиционными несмещенными методами. В отличие от оригинальной работы регуляризация применяется для ввода дополнительных «смещенных» техник

генерации путей, которые наряду с обычными «несмещенными» техниками позволяют получить Монте-Карло-оценку интеграла измерения. Для комбинации вкладов «смещенных» и «несмещенных» техник вновь используется многократная выборка по значимости, что позволяет значительно поднять скорость сходимости в «сложных» фрагментах изображения. В ходе расчета дополнительные «смещенные» техники вводятся только в том случае, если обычных «несмещенных» техник недостаточно (или таковые отсутствуют вовсе), что ведет к минимальному отклонению оценки (которая в любом случае остается состоятельной).

1. Трассировка путей (Path Tracing)

Поскольку рассматриваемая концепция регуляризации путей не зависит от конкретного алгоритма визуализации, в настоящей работе она иллюстрируется на примере трассировки путей с многократной выборкой по значимости. Алгоритм трассировки путей включает в себя построение двух типов путей от диафрагмы виртуальной камеры:

1. Каждая вершина y_i пути соединяется со случайной точкой z на источнике света, за счет чего формируется *явная видовая* траектория переноса излучения $y_0 \ldots y_i z$.
2. Если в процессе продолжения пути очередная вершина y_i оказалась на источнике света, то промежуточная последовательность $y_0 \ldots y_i$ образует *неявную видовую* траекторию переноса излучения.

Вклады различных путей комбинируются в одну оценку с помощью многократной выборки по значимости [3].

2. Проблема дефицита техник генерации путей

Многократная выборка по значимости (MIS) позволяет понизить дисперсию Монте-Карло-оценки за счет использования различных техник генерации путей. Однако, если в подобласти G исходной области интегрирования $\mathcal{P}$ доступна только одна техника, многократная выборка по значимости вырождается в обычную выборку по значимости. При этом для путей из области G повышается дисперсия оценки, что наглядно проявляется на изображении в виде ярких точек. В работе [2] данная ситуация получила название проблемы дефицита техник генерации путей. Важно подчеркнуть, что некоторые пути в принципе не могут быть обработаны традиционными несмещенными алгоритмами. Для таких путей не существует ни одной техники генерации, а их вклад в результирующее изображение будет не учтен.

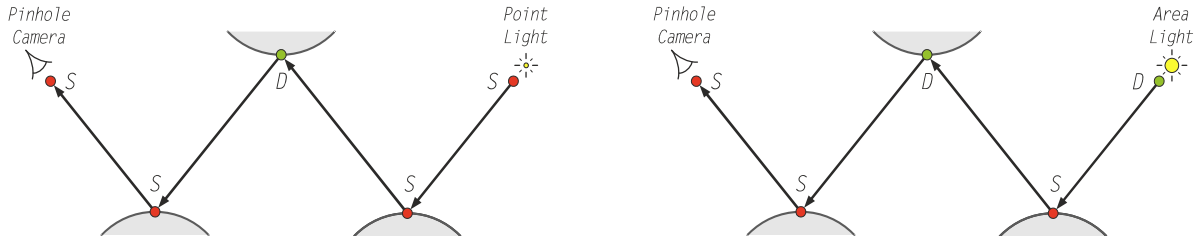


Рис. 1. Примеры путей, которые не могут быть обработаны (*слева*) или допускают только одну технику генерации (*справа*)

Как правило, дефицит техник генерации вызван особенностями подынтегральной функции измерения f_j. Особые точки возникают в областях с «зеркальными» свойствами, рассеяние света на которых описывается δ-функциями Дирака. В частности, BRDF идеального отражения может быть описана δ-функцией, которая обращается в ноль всюду, кроме направления зеркального отражения R (где она принимает бесконечно большое значение):

$$f_r(\mathbf{x}, \Psi \to \Theta) = F_r(\Psi)\frac{\delta(R-\Psi)}{|N_\mathbf{x} \cdot \Psi|}. \quad (1)$$

При этом доля отраженной световой энергии определяется коэффициентом Френеля F_r. Для генерации направления вторичного луча в вершине с «зеркальными» свойствами используется плотность вероятности, которая также выражается δ-функцией:

$$p_{\omega^\perp}(\Psi) = \frac{\delta(R-\Psi)}{|N_\mathbf{x} \cdot \Psi|}. \quad (2)$$

Аналогичным образом определяется BTDF идеального преломления.

3. Выборочная регуляризация путей переноса излучения

Таким образом, крайнее проявление проблемы дефицита техник построения выражается в наличии «нерегулярных» путей, которые в принципе не могут быть построены несмещенными методами синтеза изображений (включая двунаправленную трассировку путей и Metropolis Light Transport). В других случаях дефицит техник построения приводит к снижению эффективности многократной выборки по значимости, которая при наличии всего одной техники вырождается в обычную выборку по значимости.

В работе [1] авторы исследовали первый аспект данной проблемы, предложив простое и элегантное решение на основе выборочной регуляризации путей – замены δ-функции в одной или нескольких особых точках на приближенные распределения с протяженным носителем. В частности, метод классической трассировки путей не позволяет учесть вклад прямого освещения в точке с «зеркальными» свойствами (одно из последствий выражается в отсутствии каустик от точечных источников света). Однако, за счет замены δ-функции в BSDF «зеркальной» вершины на близкое распределение, для данной точки можно сгенерировать теневой луч и приближенно оценить вклад пути в яркость соответствующего пикселя. В работе [1] для аппроксимации δ-распределения используется кусочно-постоянная функция, которая за пределами малого угла σ обращается в ноль:

$$\phi_\sigma(D-\Psi) = \frac{1}{2\pi(1-\cos\sigma)} I_{\angle \Psi D < \sigma}. \quad (3)$$

В данном выражении через D обозначено направление идеального отражения или преломления (в зависимости от типа BSDF в «зеркальной» вершине пути), а через I – индикаторная функция, которая указывает на принадлежность направления Ψ прямому круговому конусу Ω с осью D и половинным углом раствора σ.

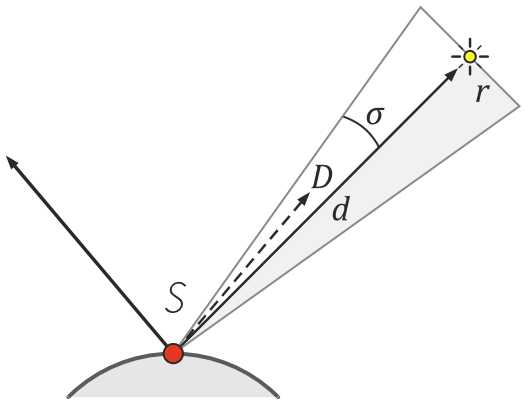


Рис. 2. Регуляризация S-вершины путем замены δ-функции на функцию с протяженным носителем

На практике угол σ удобнее вычислять на основе расстояния d между соединяемыми вершинами и заданного радиуса «аппроксимации» r. В этом случае

$$\cos\sigma = \frac{d}{\sqrt{d^2 + r^2}}. \quad (4)$$

Следует отметить, что после регуляризации Монте-Карло-оценка интеграла измерения уже не будет *несмещенной*, однако необходимо обеспечить ее *состоятельность*. В работе [1] данное требование удовлетворяется за счет уменьшения радиуса r после каждого шага интегрирования согласно формуле $r_N = r_0 N^{-\lambda}$ (здесь r_0 – заданный начальный радиус, а параметр $\lambda = 1/6$ для случая регуляризации только одной из двух соединяемых вершин). В результате, с ростом числа шагов интегрирования (объем выборки $N \to \infty$) систематическая (неслучайная) ошибка оценки будет стремиться к нулю.

3.1. Регуляризация как механизм повышения числа техник генерации

В данной работе исследуется другой аспект проблемы, вызванной особыми точками путей переноса излучения. Принцип регуляризации предлагается использовать для повышения числа стратегий генерации «сложных» путей, для которых не доступен полный набор «несмещенных» техник построения (что повышает дисперсию Монте-Карло-оценки). Регуляризация отдельных вершин позволяет расширить набор традиционных «несмещенных» техник дополнительными «смещенными» техниками генерации путей. В результате растет эффективность многократной выборки по значимости и понижается дисперсия оценки в «сложных» областях изображения. Необходимо подчеркнуть, что новые техники построения путей дают конкурирующие оценки по отношению к традиционным, поэтому возникает проблема оптимального комбинирования обычных несмещенных оценок с дополнительными смещенными в единую оценку интеграла измерения. При решении данной проблемы необходимо обеспечить баланс между скоростью сходимости и величиной систематической ошибки.

На этапе трассировки путей от диафрагмы виртуальной камеры регуляризация выполняется для вершин, соединяемых теневыми лучами с источником света. В результате к единственной *неявной* траектории добавляется «смещенная» *явная* траектория переноса света. Регуляризация вершины выполняется заменой δ-функции в «зеркальной» BSDF на приближенную функцию ϕ_{σ_N}, при этом косинус угла σ_N вычисляется на основе текущего радиуса «аппроксимации» r_N по формуле (4).

Результаты

Для сравнения техник была выбрана сцена со сложными SDS-путями. Сравнивалась трассировка путей (PT) без регуляризации и с регуляризацией. Скорость сходимости была измерена используя разницу (L2 norm) с эталонным изображением (PT 4 часа).

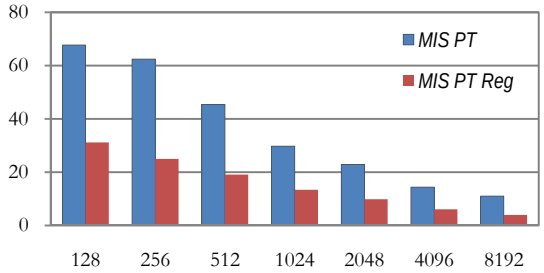


Рис. 3. Абсолютная ошибка в зависимости от числа итераций на пиксел

Рис. 4. PT (слева) и PT с регуляризацией (справа), 64 итерации на пиксел

Заключение

В настоящей работе представлен метод оптимизации, позволяющий ускорить сходимость даже базовых техник визуализации, таких как MIS PT, более чем в два раза без значительной потери качества. Алгоритм регуляризации также хорошо масштабируется на более сложные техники визуализации, такие как двунаправленная трассировка путей и MLT.

Литература

1. Kaplanyan A.S., Dachsbacher C. Path Space Regularization for Holistic and Robust Light Transport // Proc. of Eurographics – 2013.
2. Kollig T., Keller A. Efficient bidirectional path tracing by randomized quasi-monte carlo integration // In MCQMC Methods. 2000. 290–305.
3. Veach E. Robust Monte Carlo methods for light transport simulation. PhD thesis. Stanford University, 1998.
4. Heckbert P.S. Adaptive radiosity textures for bidirectional ray tracing // Computer Graphics (Proc. of SIGGRAPH). 24. 4. 1990. 145–154.

К РАЗРАБОТКЕ УЧЕБНЫХ КУРСОВ ПО ТЕМАТИКЕ СУПЕРКОМПЬЮТЕРНОГО ИНЖИНИРИНГА

Ю.Я. Болдырев, К.Ю. Замотин, Е.П. Петухов, М.А. Чурилова

Санкт-Петербургский государственный политехнический университет

Рассматриваются вопросы разработки учебных курсов, предназначенных для подготовки выпускников высшей технической школы к широкому применению суперкомпьютерных технологий в области инженерного анализа и проектирования.

Прежде чем перейти к рассмотрению процесса разработки учебных курсов, необходимо описать ключевые понятия – «суперкомпьютерные технологии» и «суперкомпьютерный инжиниринг». В работе [1] суперкомпьютерные технологии определяются как триада процессов – разработка суперЭВМ, разработка системного и прикладного программного обеспечения (ПО) и совокупность предметных областей их использования. Под суперкомпьютерным инжинирингом понимаются знания и технологии применения суперкомпьютеров, на основе которых создаются конкурентоспособные машины, агрегаты, системы и прочая продукция. Такое определение суперкомпьютерного инжиниринга полностью попадает в сегмент суперкомпьютерных технологий, как один из его наиболее наукоемких прикладных сегментов. Наукоемкость здесь упоминается с целью подчеркнуть, что суперкомпьютерные технологии являются объединяющим началом, связывающим естественные и прикладные (инженерные) науки.

Рассмотрим примеры, тактику и стратегию использования компьютерных (суперкомпьютерных) технологий в сфере промышленного производства в наиболее развитых странах мира, основываясь на публикациях последних лет. В работе [2] приведен ретроспективный анализ применения компьютерных технологий за последние 30-40 лет, который показывает, что такие технологии стали не только неотъемлемой частью практически любого производства, но и его интегрирующим ядром на основе PLM (Product Lifecycle Management) технологий. Это ядро определяет все стороны производства, начиная от задач, связанных непосредственно с созданием новых изделий, и до его обслуживающих составляющих (планирование, поставки комплектующих, экономика предприятия и т.д.). В рамках данной статьи нас в первую очередь интересует первая группа задач, связанная с созданием новых изделий на основе новых подходов инженерного анализа.

Как же эти новые подходы отразились на самом процессе производства и как их последствия должны повлиять на процесс подготовки кадров в высшей школе? Эти вопросы обсуждаются в работе «Современное инженерное образование» [3], где авторы пособия утверждают, что во многих отраслях производства произошла полная смена технологических цепочек разработки. Речь идет не только о широком внедрении компьютеров в производство, но и о смене парадигмы всей разработки. В первую очередь имеется в виду процесс параллельного проектирования, сформировавшийся в последние десятилетия. Суть его состоит в том, что проектирование изделия ведется одновременно разными группами разработчиков. При этом каждая группа отвечает либо за разработку своего сегмента изделия, либо за анализ эффективности уже разработанных другими группами сегментов. Весь процесс разработки изделия реализуется на общей

базе данных предприятия, поэтому очевидно, что продуктивность работы напрямую зависит от имеющихся вычислительных мощностей.

Второй немаловажный аспект – это переход в естественно-научном и инженерном анализе к решению междисциплинарных задач. На сегодняшний день ставятся и решаются такие классы задач, решение которых не представлялось возможным 15-20 лет тому назад. Эти задачи во многих сферах практики приближают нас к описанию реального физического мира. Поэтому именно научно- и учебно-методические работы, посвященные постановке и решению междисциплинарных задач, определяют тенденции и направления разработки учебных курсов по тематике суперкомпьютерного инжиниринга [5, 6].

Следует отметить, что переход от компьютерного инжиниринга к суперкомпьютерному связан с использованием параллельных вычислительных технологий [4, 7-9], то есть с переходом к совершенно другому классу вычислительных задач. Умение поставить задачу, в том числе и междисциплинарную, для решения на суперЭВМ составляет фундаментальную основу современного естественно-научного и инженерного знания.

Чтобы выделить основные необходимые курсы по тематике суперкомпьютерного инжиниринга, рассмотрим отрасли инженерного знания, в которых они будут применяться. В каждой из крупных областей (механика жидкости и газа, механика твердого и деформируемого тела и др.) имеется свой, присущий только данной области, блок математических моделей и технологий, составляющих ее ядро. Например, для задач горения газов и их смесей таковыми являются модели газовой динамики и модели турбулентности. Для задач, связанных с расчетами прочности машин и механизмов, такое ядро составляют уравнения теории упругости и модели механики конструкций. Таким образом, фундаментальные основы каждой из отраслей составляют теоретические и экспериментальные методы построения математических моделей, описывающих те или иные явления в данной области.

Каждой из перечисленных отраслей присущи свои особенности применения суперкомпьютерных технологий как в инженерных, так и в естественно-научных расчетах. Например, в прикладных задачах аэрогидродинамики одной из важнейших проблем является проблема описания турбулентности. Именно она была одной из тех проблем, что стимулировали развитие высокопроизводительных вычислительных технологий. Приведем таблицу из работы [10], иллюстрирующую соотношение вычислительных ресурсов и перспективы практического применения различных подходов к моделированию турбулентных течений. В табл. 1 приведены наиболее распространенные модели турбулентности, именуемые в англоязычной литературе Direct Numerical Simulation (DNS) – прямое численное моделирование, Large Eddy Simulation (LES) – метод моделирования крупных вихрей, Detached-Eddy Simulation (DES) – метод моделирования отсоединенных вихрей и, наконец, Reynolds Averaged Navier-Stokes (RANS) – уравнения Навье–Стокса осредненные по Рейнольдсу [11].

Помимо фундаментальных основ инженерного знания в виде математических моделей и методов вычислительной математики в тех или иных отраслях, для применения суперкомпьютеров инженерам необходимо знать методы постановки задачи на суперкомпьютере. Например, к ним относится декомпозиция задачи или разбиение ее на части, каждая из которых обрабатывается отдельным вычислительным узлом или одним из его ядер. От решения этой проблемы весьма существенно зависит сходимость вычислительного процесса и его продолжительность.

Таблица 1. Перспективы практического применения различных подходов к моделированию турбулентных течений

Метод	Необходимое число узлов сетки	Необходимое число шагов по времени	Год готовности[1]
3D Steady RANS	10^7	10^3	1985
3D Unsteady RANS	10^7	$10^{3.5}$	1995
DES	10^8	10^4	2000
LES	$10^{11.5}$	$10^{6.7}$	2045
DNS	10^{16}	$10^{7.7}$	2080[2]

Является ли поиск решения, удовлетворяющего определенным начальным и граничным условиям (для задач, связанных с уравнениями математической физики), единственной задачей суперкомпьютерного инжиниринга? Предположим, что мы получили решение нашей задачи для какой-то совокупности определяющих параметров. Таких параметров может быть множество, и они могут иметь самый разнообразный характер. Например, это могут быть характерные размеры или их совокупность, различные формы поверхности и так далее. Тогда перед разработчиком встает вопрос: насколько удачно выбраны параметры для того, чтобы характеристики системы были наилучшими? То есть оптимально ли построена рассматриваемая система? Таким образом, мы приходим к необходимости включения оптимизационных задач в область применения суперкомпьютерного инжиниринга. Примером такой проблемы является задача оптимизации формы профиля крыла самолета для получения максимальной подъемной силы.

В качестве характерного примера междисциплинарной задачи, требующей для своего решения применения суперкомпьютерных технологий, рассмотрим разработку гидротурбин в энергомашиностроительной отрасли. Мощные гидротурбины современных гидроэлектростанций – это высокотехнологичные изделия. При наличии жесткой конкуренции на мировом рынке ведется борьба за доли процентов КПД гидромашин. КПД определяется геометрией проточной части гидротурбины (рабочее колесо, направляющий аппарат и т.д.), поэтому для его повышения необходима оптимизация геометрии, которую невозможно провести только с помощью физического эксперимента. Требуемое оптимальное профилирование всех элементов проточной части, с точки зрения математики, есть связанная пространственная задача Лагранжа вариационного исчисления [8], где в качестве связей выступают дифференциальные уравнения механики вязкой жидкости. Также необходимо учесть турбулентный характер течения. Более глубокий анализ показывает, что при испытываемых нагрузках рабочее колесо турбины изнашивается под воздействием кавитации. Таким образом, проблема становится междисциплинарной – при расчете кроме уравнений турбулентного движения жидкости необходимо учесть процессы кавитации на поверхности металла, включая проблемы, связанные с физикой металлов. Также нельзя забывать про процессы упругогидродинамического взаимодействия рабочего колеса с потоком жидкости, которые при имеющемся уровне давлений (многие десятки атмосфер) могут оказаться существенными.

Таким образом, полномасштабная корректная физико-математическая постановка задачи об описании течения в проточном тракте гидротурбины представляет собой совокупность связанных начально-краевых задач математической физики. При этом зада-

[1] Под готовностью подразумевается возможность расчета одного варианта в течение суток на самых мощных из доступных компьютеров.

[2] На компьютере с производительностью 1 Тфлоп/с время расчета составляет 5000 лет.

ча решается в весьма сложной по своей геометрии области и для ее описания необходимо разбиение области решения на многие миллионы ячеек (конечных объемов). В работах с ЛМЗ в 2006-2007 годах [6] вычисления во всей проточной части проводились на 5,6 миллионах конечных объемов, причем в каждом конечном объеме вычисляются три компоненты вектора скорости и давление, плюс две составляющие энергии турбулентных пульсаций. Также при решении междисциплинарных задач дополнительно необходимо знать температуру воды. Таким образом, число неизвестных доходит уже до десятков миллионов.

Таким образом, мы видим, что в результате необходимо разрабатывать систему курсов, в рамках которых студентам даются представления не только о современных суперкомпьютерах и их возможностях, но и множество материала для постановок и решения вычислительно ресурсоемких междисциплинарных задач математического моделирования. В таких дисциплинах должны рассматриваться и современная концепция математического моделирования, роль и место в ней высокопроизводительных вычислительных систем. И, конечно, курсы должны быть направлены на освоение учащимися подходов к решению широкого круга междисциплинарных прикладных инженерных задач, а также на приобретение ими начальных знаний по методам и подходам применения передовых программных комплексов для инженерного и естественно-научного анализа в их версиях для суперкомпьютеров.

Таким образом, главная цель создаваемых курсов – привитие учащимся практических навыков работы (в естественно-научном и инженерном анализе) на мощных вычислительных системах.

Авторы благодарят компанию Intel за поддержку работ в данной области.

Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, соглашение 14.B37.21.0594.

Литература

1. Болдырев Ю.Я., Петухов Е.П. Суперкомпьютерные технологии и их приложения. СПбГПУ: Изд-во Политехн. ун-та. 2011. 88 с.
2. Боровков А.И., Бурдаков С.Ф. и др. Компьютерный инжиниринг. Учебное пособие. СПб.: Изд. СПбГПУ, 2012. 93 с.
3. Боровков А.И., Бурдаков С.Ф. и др. Современное инженерное образование. Учебное пособие. СПб.: Изд. СПбГПУ, 2012. 81 с.
4. High Performance Computing in Science and Engineering ’11. Transactions of the High Performance Computing Center (Editors Wolfgang E. Nagel, Michael M. Resch, Dietmar B. Kröner). Stuttgart (HLRS), 2011. P. 649.
5. Болдырев Ю.Я. Суперкомпьютерные технологии как современное воплощение междисциплинарного подхода в научно-образовательной деятельности // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. № 4. 2010. С. 99-106.
6. Болдырев Ю.Я. Роль суперкомпьютерных технологий в инженерном образовании // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. № 162. 2012. С. 9-15.
7. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: BHV, 2002. 599 с.
8. Гергель В.П. Теория и практика параллельных вычислений: Учеб. Пособие. М.: Интернет-Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2007. 423 с.

9. Гергель В.П., Фурсов В.А. Лекции по параллельным вычислениям. Самара: Издательство СГАУ, 2009. 163 с.
10. Spalart P.R. Strategies for turbulence modeling and simulations // Int. J. Heat Fluid Flow. 2000. V. 21. P. 252–263.
11. Гарбарук А.В., Стрелец М.Х., Шур М.Л. Возможности и ограничения современных подходов к моделированию турбулентности в аэродинамических расчетах. Учебное пособие. СПбГПУ, 2012. 88 с.

МОДЕЛИРОВАНИЕ ТУРБУЛЕНТНОСТИ В ИНЖЕНЕРНЫХ РАСЧЕТАХ НА ОСНОВЕ СУПЕРКОМПЬЮТЕРОВ

Ю.Я. Болдырев, А.О. Рубцов

Санкт-Петербургский государственный политехнический университет

Поиск приемлемых для практики технологий математического описания турбулентных течений или, как принято говорить, моделей турбулентности уже на протяжении более 100 лет (начиная с классических работ О. Рейнольдса) занимает умы многих выдающихся математиков и механиков. Это объясняется как исключительной сложностью самого феномена турбулентности как физического явления, так и тем, что именно турбулентная форма движения газов и жидкостей наиболее часто реализуется в природе и, соответственно, должна учитываться при расчетах в различных технических приложениях. Согласно современным представлениям, турбулентные течения подчиняются классическим уравнениям Навье–Стокса. Однако, несмотря на значительный прогресс вычислительной техники, наблюдаемый в последние десятилетия, ее возможности все еще недостаточны для решения этих уравнений при представляющих практический интерес высоких числах Рейнольдса, то есть при развитой турбулентности. И даже по самым оптимистичным прогнозам, ситуация останется таковой, по крайней мере вплоть до второй половины – конца XXI века.

При этом общий прогресс вычислительной аэродинамики не мог не сказаться на состоянии проблемы моделирования турбулентности. В частности, в последние годы все большее применение находят подходы к моделированию турбулентности, базирующиеся на «первых принципах» аэродинамики, таких как метод прямого численного моделирования (DNS – Direct Numerical Simulation) и метод моделирования крупных вихрей (LES – Large Eddy Simulation). Отметим, что такие подходы крайне вычислительно трудоемки. Например, моделирование турбулентности по методу DNS с необходимым для практики числом узлов сетки порядка 10^{16} и числом шагов по времени порядка $10^{7.7}$ на компьютере производительностью в 1 терафлоп должно занять около 5000 лет. Также отметим, что для достижения необходимой точности расчета аэродинамических характеристик, независимо от выбранного метода моделирования турбулентности, требуется учитывать большое число переменных, что приводит к необходимости производить расчеты на суперкомпьютере. Для сравнения можно рассмотреть таблицу, приведенную в учебном пособии А.В. Гарбарука, М.Х. Стрельца и М.Л. Шура (2012 г.) [1], иллюстрирующую объемы необходимых вычислительных ресурсов и перспективы практического применения различных методов моделирования турбулентных течений.

Таблица 1

Метод	Необходимое число узлов сетки	Необходимо число шагов по времени	Год готовности[1]
3D Steady RANS	10^7	10^3	1985
3D Unsteady RANS	10^7	$10^{3.5}$	1995

[1] Подразумевается, что расчет производится на самых мощных компьютерах из доступных.

DES	10^8	10^4	2000
LES	$10^{11.5}$	$10^{6.7}$	2045
DNS	10^{16}	$10^{7.7}$	2080

Кратко остановимся на наиболее востребованных моделях турбулентности. В настоящее время для моделирования турбулентности применяются такие методы:

- RANS-модель (Reynolds Averaged Navier-Stokes) – уравнения Рейнольдса получаются из системы уравнений Навье–Стокса путем использования процедуры осреднения, предложенной О. Рейнольдсом. Система уравнений, полученная после осреднения, является незамкнутой, поскольку связь между турбулентными составляющими тензора напряжений и вектора плотности теплового потока с параметрами осредненного течения неизвестна. Эта связь должна быть доопределена с помощью дополнительных соотношений, которые и представляют собой модель турбулентности.
- Полуэмпирические модель Спаларта-Аллмараса (содержит одно дифференциальное уравнение переноса для модифицированной турбулентной вязкости) и модель Ментера (содержит два дифференциальных уравнения переноса для кинетической энергии турбулентности и удельной скорости её диссипации). SA-модель, как правило, «затягивает» отрыв пограничного слоя, индуцированный неблагоприятным продольным градиентом давления. Обе эти модели занижают темп релаксации пограничного слоя, формирующегося вниз по потоку от точки присоединения, к своему равновесному состоянию и значительно завышают размеры так называемого «углового отрыва», т.е. отрыва от поверхности двухгранного угла при наличии неблагоприятного градиента давления. При этом, например, для течений с сильной кривизной линий тока и вращением ни одна из этих моделей не позволяет получить результаты, удовлетворяющие современным требованиям к точности расчета аэродинамических характеристик.
- Метод крупных вихрей (Large Eddy Simulation), уравнения которого могут быть получены из уравнений Навье–Стокса путем представления всех переменных в виде суммы крупно- и мелкомасштабной составляющих и последующего применения процедуры фильтрации.
- Метод отсоединенных вихрей (Detached Eddy Simulation), который был предложен в качестве альтернативы RANS- и LES-метода при расчете пристеночных течений с обширными отрывными зонами, для которых RANS модели не способны обеспечить приемлемую точность, а LES требует чрезмерно больших вычислительных ресурсов.
- Метод прямого численного интегрирования DNS, который базируется на единственном общепринятом в настоящее время допущении о том, что уравнения Навье–Стокса адекватно описывают не только ламинарные, но и турбулентные течения. В рамках этого метода расчет турбулентных течений производится путем непосредственного (без предварительного осреднения) численного решения уравнений Навье–Стокса. При этом, вне зависимости от характера рассматриваемого течения, должны использоваться трехмерные нестационарные уравнения Навье–Стокса, так как турбулентность является принципиально трехмерным и нестационарным явлением. Кроме того, DNS-метод подразумевает необходимость достаточно точного разрешения всех пространственно-временных масштабов турбулентности, из чего следует необходимость использования суперкомпьютеров при расчетах.

Для эффективного применения этих и других методов моделирования турбулентности и проведения расчетов на суперкомпьютерах существует большое количество различных программных комплексов. Подавляющее большинство таких комплексов

имеют параллельную версию для эффективного использования многопроцессорных компьютеров. Как правило, в параллельных версиях расчетная область разделяется на части, то есть проводится декомпозиция области. Затем построенные области распределяются между доступными процессорами.

Среди наиболее популярных и широко используемых программных комплексов, применяемых для моделирования турбулентных течений в задачах аэрогидродинамики (CFD – Computer Fluid Dynamics), можно выделить:

- Программный комплекс ANSYS. Это мощная CAE-система, позволяющая решать мультифизические задачи. Благодаря наличию препроцессора, комплекс позволяет не только создавать геометрические модели встроенными средствами, но и импортировать готовые, созданные в CAD-системах. В настоящее время в программный комплекс ANSYS интегрированы такие мощные программные системы для моделирования турбулентности, как FLUENT и CFX.
- Программная система Star-CD. Это многоцелевой единый CFD-комплекс, предоставляющий пользователю возможность решения задач механики жидкости и газов на всех типах сеток: стационарные и нестационарные течения, ламинарные течения – модели ньютоновской и неньютоновской жидкостей, турбулентные течения (применяются несколько наиболее известных моделей), около- и сверхзвуковые течения, задачи теплопереноса, горения, многокомпонентные течения, многофазные потоки, задачи со свободными поверхностями и др.
- Программная система CFD-FASTRAN. Это мощный многоотраслевой инструмент для анализа летательных аппаратов. Комплекс объединяет гидродинамический модуль, модуль конечно-элементного расчета динамики твердого тела, а также обеспечивает их согласованное взаимодействие. Помимо этого, в CFD-FASTRAN имеется алгоритм сеточной интерполяции для подвижных сеток.

В качестве наглядного примера численного расчета турбулентных течений можно рассмотреть классическую задачу об эволюции вихря Тейлора–Грина. Эта задача представляет собой пример распада первоначального одиночного стационарного вихря с последующим образованием каскада все более мелких вихревых образований, которые постепенно затухают за счет присущих течению диссипативных процессов. Данная задача хорошо изучена теоретически для малых и больших чисел Рейнольдса, поэтому она используется в качестве теста для проверки адекватности численных алгоритмов моделирования турбулентности. Результаты решения задачи с использованием метода LES представлены на рис. 1 [2].

Еще одним примером расчета турбулентных течений является работа сотрудников Санкт-Петербургского государственного политехнического университета, проведенная в 2006-2007 годах, посвященная исследованию течений в гидротурбинах. Полная проточная часть гидротурбины изображена на рис. 2а). Рассматривались как полномасштабный вариант расчета, то есть во всей проточной части, так и расчет по частям: спиральная камера, направляющий аппарат, сама турбина, а также так называемая отсасывающая труба. При тех вычислительных ресурсах, которые имелись на тот момент у разработчиков проекта (8- и 48-процессорные кластеры), можно было проводить достаточно ограниченные исследования достаточно сложной системы. Тем не менее, учитывая, что в проточной части имеется развитое турбулентное течение, проблема выбора тех или иных моделей турбулентности была весьма актуальной [3, 4]. На рис. 2б) показаны характерные мгновенные изоповерхности кинетической энергии турбулентности за гидротурбиной в области отсасывающей трубы. Заметим, что именно срывы вихрей, подобные приведенным на рис. 2б), служат одним из источников мощных пульсаций давления в проточной части турбины. В результате этой работы были проведены многовариантные расчеты течения в проточной части турбины. На момент её проведения

она была уникальной для России. При расчете течения рассматривались два базовых варианта такого ключевого параметра, как объёмный расход через турбину, – 368 м^3/с и 319.2 м^3/с. При этом проводились расчеты КПД системы. В таблице 2 приведены характерные параметры расчетов.

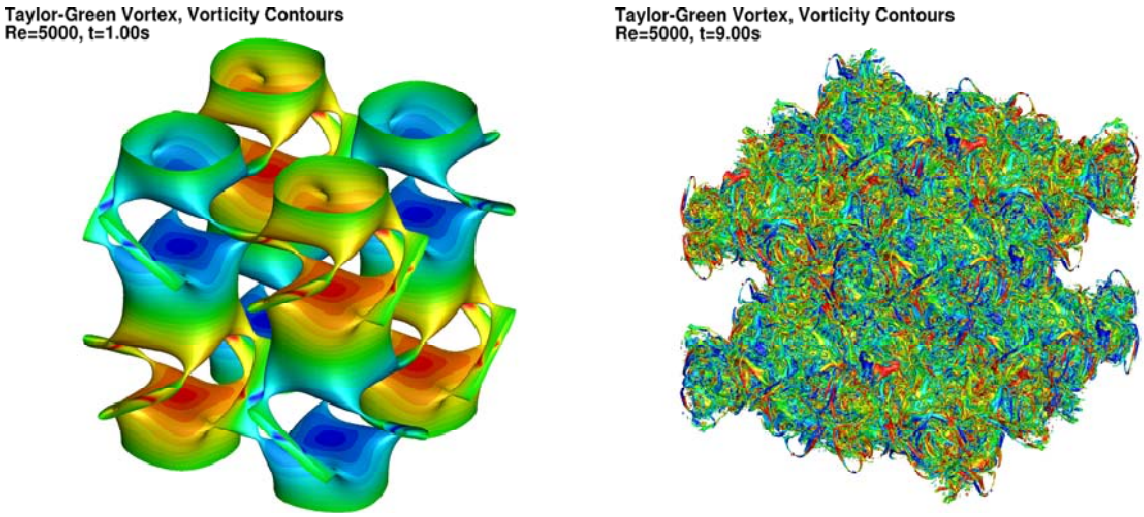
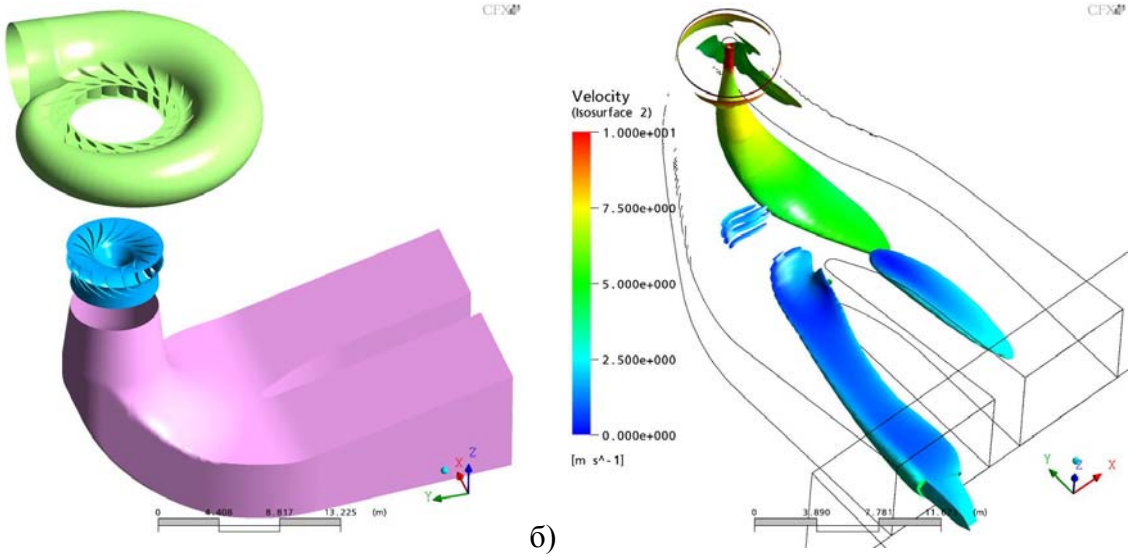


Рис. 1. Поверхности уровня завихренности поля скорости
для моментов времени t=1 с (слева) и t=9 с (справа)

Отметим, что величина экспериментального значения КПД при объёмном расходе 330 м^3/с равна 95.27% , что на рассматриваемом уровне моделирования хорошо согласуется с численными результатами. К сожалению, имевшиеся на тот момент времени вычислительные ресурсы не позволили решать задачу на более «мелких» сетках. Вместе с тем на сравнительно малых ресурсах удалось достаточно хорошо протестировать ряд наиболее употребительных моделей турбулентности в данной, крайне важной практически задаче. В расчетах использовались программные комплексы FLUENT и CFX.



а) б)

Рис. 2. Проточная часть (а), вихревая зона за турбиной (б)

В докладе приведен и ряд других задач моделирования турбулентности в разнообразных машинах и системах, при этом основным тезисом является то обстоятельство, что уровень адекватности моделирования физической реальности напрямую связан с доступными вычислительными мощностями.

Таблица 2

Общее число узлов расчётных сеток	Объёмный расход 368 $м^3/с$		Объёмный расход 319.2 $м^3/с$	
	КПД всей гидротурбины при расчёте по частям, %	КПД всей гидротурбины при расчёте проточного тракта в целом, %	КПД всей гидротурбины при расчёте по частям, %	КПД всей гидротурбины при расчёте проточного тракта в целом, %
2 547 010	83.46	85.41		86.18
5 624 563	90.65	93.37		93.73
38 099 030	94.98		95.46	

Литература

1. Гарбарук А.В., Стрелец М.Х., Шур М.Л. Возможности и ограничения современных подходов к моделированию турбулентности в аэродинамических расчетах. Учебное пособие. СПбГПУ, 2012. С. 88.
2. High Performance Computing in Science and Engineering ’11. Transactions of the High Performance Computing Center (Editors Wolfgang E. Nagel, Michael M. Resch, Dietmar B. Kröner). Stuttgart (HLRS), 2011. P. 649.
3. Болдырев Ю.Я., Викторов Е.Д., Шиндер Ю.К. и др. Отчет о НИР «Сравнительный анализ результатов расчета 3-мерного потока жидкости в типовых элементах гидромашин, основанный на применении пакетов гидродинамических программных комплексов Cadrun, Fluent и CFX». СПб.: СПбГПУ, 2005. 136 с.
4. Болдырев Ю.Я., Викторов Е.Д., Шиндер Ю.К. и др. Отчет о НИР «Разработка методических вопросов расчета потерь энергии в элементах проточной части гидротурбин с применением вычислительной гидродинамики». СПб.: СПбГПУ, 2007. 70 с.

ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА РАСПРЕДЕЛЕННЫХ ВС

А.А. Бондаренко, М.В. Якобовский

Институт прикладной математики им. М.В. Келдыша РАН

Рассматриваются проблемы обеспечения отказоустойчивости параллельных программ при расчетах на распределенных вычислительных системах. Приводится анализ видов отказа оборудования; определение методов диагностирования отказов различных видов; определение состава и объёмов данных, необходимых и достаточных для проведения локального перерасчета величин, значения которых утрачены в результате отказа оборудования.

Современные суперкомпьютеры состоят из десятков тысяч узлов, каждый из которых оснащен процессорами и, как правило, различными ускорителями. Увеличение количества компонентов системы естественным образом ведет к увеличению вероятности отказа некоторых элементов. Учитывая отмеченную специфику перспективных экзафлопсных вычислительных систем, особенно важным становится вопрос об обеспечении самой возможности осуществления длительного расчета на больших и сверхбольших распределенных системах. «Сами принципы отказоустойчивости надо пересмотреть: задача должна уметь сама себя восстанавливать после сбоя, без вмешательства человека» [1, с. 22].

Обеспечение отказоустойчивой работы на вычислительных системах больших масштабов связано с решением двух задач. Первая – повышение надежности системы на аппаратном уровне, дублирование основных управляющих компонентов, поддержка горячей замены отказавшего оборудования и т.п. Вторая – разработка программ, использующих новые подходы к организации отказоустойчивых распределенных вычислений.

Данная работа посвящена решению второй задачи.

Основной подход к обеспечению отказоустойчивости параллельных вычислений предполагает периодическое сохранение состояния расчета задачи на внешние носители, что позволяет при выходе из строя части вычислительного поля продолжить расчет с последней сохранённой глобальной контрольной точки [2, 3]. Однако, с увеличением числа обрабатывающих элементов и процессорных узлов время наработки на отказ вычислительной системы становится мало и сравнимо как со временем проведения сеанса расчета, так и, при дальнейшем увеличении числа процессоров, со временем сохранения контрольной точки. Фактически, в перспективе, с ростом числа процессоров сама возможность успешного сохранения глобальной контрольной точки окажется под вопросом.

Другой подход к организации отказоустойчивых вычислений основан на создании новых библиотек или использовании языков программирования, в том числе поддерживающих объекты с локальной синхронизацией и контроль состояния распределенного вычисления, включая параллелизм по данным. Реализацией данного подхода является библиотека «T sims» [4] для среды «T система». Еще одним примером является программный комплекс «Пирамида» [5], разрабатываемый МСЦ РАН совместно с НИИ «Квант». Данные комплексы обеспечивают работоспособность системы при выходе из строя узлов вычислительной системы. Однако «T sim» и «Пирамида» осуществляют

организацию вычислительного параллельного процесса исключительно для задач, в которых отсутствуют информационные обмены между процессами.

В данной работе развивается подход к организации работы вычислительной системы, в рамках которого могут присутствовать информационные обмены между процессами. Он основан на создании механизмов поддержания локальных контрольных точек – распределенных промежуточных данных (аналогичный подход развивается в НИИСИ РАН под руководством Бителина В.Б. и в [6]). Соответствующие данные, в случае выхода из строя части вычислительной мощности, должны обеспечивать возможность продолжения расчета, ценой повторного расчета состояния части моделируемого поля.

В рамках данного подхода рассмотрен отказ оборудования, при котором теряется связь с программой, выполняемой на отказавшем вычислительном узле (например, вследствие выхода из строя элементов процессорного узла, таких как сам процессор, диски, сетевые контроллеры).

Разработка методов обеспечения отказоустойчивости актуальна для высокопроизводительных программных комплексов, функционирующих на современных суперкомпьютерах. Это объясняет востребованность дальнейших исследований в этом направлении.

В работе развивается подход обеспечения отказоустойчивости параллельных вычислений, основанный на локальных контрольных точках.

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту 13-01-12073 офи_м.

Литература

1. Донгарра Дж. Экзафлопсное будущее суперкомпьютеров // Суперкомпьютеры. №1(1). 2010. – С. 21–23.
2. Plank J.S., Li K. Faster checkpointing with n+1 parity // FTCS, 1994, P. 288–297.
3. Sriram Sankaran, Jeffrey M. Squyres, Brian Barrett, Andrew Lumsdaine, Jason Duell, Paul Hargrove, and Eric Roman. The LAM/MPI Checkpoint/Restart Framework: System-Initiated Checkpointing // In LACSI Symposium, October 2003 (publication LBNL-53808 Proc.) – [http://crd.lbl.gov/groups-depts/ftg/projects/current-projects/BLCR/].
4. Тютляева Е.О., Московский А.А. Методы обеспечения отказоустойчивости в библиотеке шаблонных классов C++ для распараллеливания T Sim // Программные системы: теория и приложения: Электрон. научн. журн. 2011. № 3(7). С. 17–28 – [http://psta.psiras.ru/read/psta2011_3_17-28.pdf].
5. Баранов А.В., Киселёв А.В., Киселёв Е.А., Корнеев В.В., Семёнов Д.В. Программный комплекс «Пирамида» организации параллельных вычислений с распараллеливанием по данным – [http://agora.guru.ru/abrau2010/pdf/299.pdf].
6. Поляков А.Ю., Данекина А.А. Оптимизация времени создания и объёма контрольных точек вос-становления параллельных программ // Вестник СибГУТИ. – Новосибирск: СибГУТИ, 2010. – № 2. – С. 87-100.

МНОГОКРИТЕРИАЛЬНЫЙ МЕТОД ПОДЖСТРОЕНИЯ КАРТЫ ДИСПАРАТНОСТИ И ЕГО РЕАЛИЗАЦИЯ СРЕДСТВАМИ GPGPU

А.Н. Волкович

Объединенный институт проблем информатики НАН Беларуси

Предложен многокритериальный подход к построению плотных карт диспаратности, разработанный с целью повышения качества 3D-реконструкции. Кроме того, приведены результаты вычислительного эксперимента, основанного на сравнении реализаций метода на стандартных вычислительных системах и GPGPU.

Введение

Разработка математических методов и построение на их основе высокоскоростных алгоритмов и систем прецизионной цифровой обработки последовательности динамических сигналов и изображений видимого и ИК диапазонов, наблюдение которых проводилось при наличии различных помех, среди которых мультипликативные и аддитивные шумы, топологические и геометрические искажения, ошибки квантования и дискретизации исходных аналоговых сигналов при их цифровом представлении, девиационные воздействия внешних источников на регистрирующую аппаратуру, является одним из наиболее актуальных направлений исследований в современном компьютерном видении.

1. Применение цветовых метрик при поиске соответствий локальных областей изображений

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.

При переходе к работе с тремя компонентами изображения можно представить в виде «облака» точек в трехмерном пространстве с осями, соответствующими цветовым каналам изображения.

Следует отметить, что система RGB на самом деле не является ортогональной. Данная особенность связана с тем, что реальная чувствительность человеческого глаза по соответствующим каналам неравномерна. Неортогональность RGB-системы приводит к возникновению погрешностей при проведении математических операций над элементами цветового пространства, что привело к необходимости создания цветовой модели XYZ, которая являлась бы ортогональной.

CIE XYZ – линейная 3-компонентная цветовая модель, основанная на результатах измерения характеристик человеческого глаза. Она построена на основе зрительных возможностей «стандартного наблюдателя», гипотетического зрителя, возможности которого были тщательно изучены и зафиксированы в ходе длительных исследований че-

ловеческого зрения, проведённых комитетом CIE (фр. *Commission Internationale de l'Eclairage*).

Функции соответствия цветов – значения каждой первичной составляющей света (красной, зелёной и синей), которые должны присутствовать на изображении, для того чтобы человек со средним зрением мог воспринимать все цвета видимого спектра. Данным трём первичным составляющим R, G, B были поставлены в соответствие координаты X, Y и Z.

Основное свойство, присущее XYZ-системе, – положительная определённость, т.е. любой физически ощутимый цвет представляется в системе XYZ только положительными величинами. Также следует отметить, что не всем точкам в пространстве XYZ соответствуют реальные цвета в силу неортогональности функций соответствия цветов.

Поскольку мерой сравнения точек в трехмерном пространстве выступает евклидово расстояние, которое корректно применимо к ортогональным системам, следует производить ортогонализации пространства RGB в пространство XYZ.

Максимальное значение длины вектора в пространстве, построенном для 8-битного цветового канала, составляет порядка 441 единицы, что в 1,7 раза превосходит по информативности величину в 256 единиц – максимальное расстояние между точками при сравнении точек в полутоновом представлении изображения. Помимо того, при переходе к сравнению цветов количество уникальных значений, описывающих точки, увеличивается в 65 тысяч раз по сравнению с монохромным.

Численные эксперименты показали значительное улучшение адекватности восстановления модифицированным алгоритмом, который позволяет более четко определить границы объектов со сходными визуальными характеристиками, визуально сливающихся с фоном, а также объектов с мелкими деталями.

2. Использование оператора градиента в целях поиска информативных областей изображений и определения размерности сканирующего окна

Несмотря на увеличение диапазона сравниваемых значений за счет использования цветовой информации, сохраняется 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. Отсюда очевидно возникает задача поиска таких участков, что позволит исключить из обработки неинформативные области и, следовательно, сократить количество ошибок и число итераций алгоритма.

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. В связи со спецификой локальных вычислений способ определения «существенных» значений состоит в установлении порога. В свою очередь, для количественного выражения изменения яркости используют понятия первой и второй производных.

Определение точки изображения как точки перепада происходит в том случае, если ее двумерная производная первого порядка превышает некоторый заданный порог. Связное множество таких точек в соответствии с заранее заданным критерием связности есть перепад яркостей, а протяженный перепад яркостей является контуром.

Один из подходов к связыванию точек контура состоит в анализе характеристик пикселей в небольшой (3×3 или 5×5) окрестности каждой точки (x, y) изображения, которая была отмечена как контурная точка (точка перепада). Все точки, являющиеся сходными в соответствии с некоторыми заранее заданными критериями, связываются и

образуют контур, состоящий из пикселей, отвечающих этим критериям. При таком анализе используются следующие два основных параметра для установления сходства пикселей контура: величина отклика оператора градиента и направление вектора градиента.

Вычисление первой производной цифрового изображения основано на различных дискретных приближениях двумерного градиента. Направление вектора градиента совпадает с направлением максимальной скорости изменения функции f в точке (x, y).

Вычисление градиента изображения состоит в получении величин частных производных $Gx = df/dx$ и $Gy = df/dy$ для каждой точки. Один из способов нахождения первых частных производных Gx и Gy в конкретной точке состоит в применении следующего градиентного оператора Собеля:

$$Gx = (z7 + 2 * z8 + z9) - (z1 + 2 * z2 + z3),$$

$$Gy = (z3 + 2 * z6 + z9) - (z1 + 2 * z4 + z7).$$

Для оператора Собеля, который выявляет горизонтальные и вертикальные контуры (перепады яркости), определяют соответствующие маски для свертки с исходным изображением. Также при помощи масок можно изменить приведенные формулы таким образом, чтобы они давали максимальный отклик для контуров, направленных диагонально. У каждой из масок сумма коэффициентов равна нулю, таким образом данные операторы будут давать нулевой отклик на областях постоянной яркости, что характерно дифференциальному оператору. Рассмотренные маски применяются для получения составляющих градиента Gx и Gy. Для вычисления величины градиента эти составляющие необходимо использовать совместно, определяя величину градиента приближенно через абсолютные значения частных производных:

$$Gradient = |Gx| + |Gy|.$$

Обработка изображения оператором градиентов, а также последующая бинаризация результатов позволит составить карту вычислений.

Карту вычислений возможно использовать для обработки изображения при помощи динамических сканирующих окон. Данный процесс основывается на постепенном увеличении размеров окна, в том случае если в зоне его сканирования оказывается недостаточное количество информации.

Данный подход позволяет уменьшить число ошибок обработки однородных областей, однако следует отметить, что по мере увеличения размеров окна происходит «раздувание» границ объекта (при размерах окна более 50x50), что искажает реальные контуры объекта.

Также следует отметить усреднение результатов сравнения областей исходя из специфики функций мер сходства, что приводит к снижению результативности обработки в случаях, когда число точек в сканирующем окне начинает превышать размерность диапазона возможных значений функции сходства.

Таким образом, можно говорить, что существует возможность автоматического определения размерности окна в некотором диапазоне, который определяется в зависимости от битности исходных данных из требований неравенства для преобразованного XYZ цветового пространства:

$$S_{scan} \leq \sqrt{(\max(X) - \min(X))^2 + (\max(Y) - \min(Y))^2 + (\max(Z) - \min(Z))^2},$$

где S_{scan}– площадь сканирующего окна; $\max(X \ldots Y \ldots Z)$ – максимально возможное значение компонента; $\min(X \ldots Y \ldots Z)$ – минимально возможное значение компонента.

В случае когда размерность окна достигает возможного максимума не «набрав» в сканируемой области достаточного числа информации, следует произвести отсеивание

данной точки, в связи с тем, что информация о ее положении в пространстве неоднозначна.

3. Привязка точек малотекстурированных областей

По результатам работы этапов обработки изображения, описанных выше, формируется карта диспаратности со значительными пропусками. Данный факт вызван тем, что изображения реального мира обладают большим количеством малотекстурированных областей. Практика показывает необходимость поиска принципиально иного подхода к областям изображений, обладающих данными характеристиками.

Анализируя механизм оценки малотекстурированных областей, используемых в «живых» системах, в частности зрения человека, было определено, что помимо непосредственного анализа бинокулярных изображений человек обладает рядом вспомогательных средств, позволяющих уточнить объемные характеристики окружающего пространства, среди которых: аккомодация (изменение фокусного расстояния), конвергенция (сведение зрачков на близкие объекты интересов), саккадические движения (псевдохаотические рефлекторные мелкие движения глазами), ассоциативное восприятие, воображение.

Производя обработку заранее полученных изображений, следует отметить невозможность реализации механизма аккомодации. Учитывая уровень развития программно аппаратных систем искусственного интеллекта и компьютерного зрения, можно говорить о том, что, производя обработку ранее полученных изображений, возможно реализовать метод, близкий к механизму саккадических движений.

Данный механизм предлагается к реализации путем определения взаиморасположения точки в малотекстурированной области относительно информативных участков. Следует отметить, что определение полной конфигурации области вокруг каждой точки и поиск области аналогичной конфигурации требует значительных вычислительных ресурсов, кроме того, выделение области при помощи алгоритмов типа Region Growing может привести к неприемлемым результатам ввиду различия ракурсов и возможных различий в смыканиях замкнутости областей.

С целью вычислительного упрощения алгоритма и избежания формирования ложных конфигураций областей автором предлагается проведение «привязки» точки к информативным областям только по нескольким направлениям (в программной реализации количество направлений может определяться в зависимости от конфигурации системы). Таким образом формируется группа дескрипторов, включающих в себя значения пиксельного расстояния в некотором направлении от точки до точки со значением оператора градиента, превышающим некоторый определенный на этапе конфигурации порог. В свою очередь, на втором изображении производится поиск точки с аналогичными дескрипторами.

Основываясь на данном подходе производится заполнение оставшихся участков карты диспаратности.

4. Параллельная реализация метода средствами GPGPU

В рамках выполнения работ по созданию системы автоматической реконструкции трехмерных сцен по нескольким изображениям была протестирована реализация композиционного метода построения плотных карт диспаратности, основанного на описанных выше подходах.

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.

Тестирование проводилось на работчих станциях на базе процессора Intel i7 3.4 ГГц, 16384 Mb ОЗУ, графическая карта NVidia GeForce GTX450 и Intel Core 2 Duo 2.6 ГГц, 2048 Mb ОЗУ, графическая карта NVidia GeForce 9600GT.

Замеры на CPU выполнялись по пятьдесят раз для каждой пары изображений; в таблице 1 представлено среднее арифметическое время. В связи с тем, что результаты измерений на GPU значительно (более чем на 15%) отличались друг от друга, для оценки данного технологического подхода проводилось 300 испытаний, при этом исключались явные экстремумы значений, которые могли быть вызваны несбалансированным использованием ресурсов системой Windows.

Таблица 1

Стереоизображения	**Intel Core Duo 2.6 GHz Dual Mode**	**Intel i7 3.4 GHz**	**Intel Core Duo 2.6 GHz GPGPU**	**Intel i7 3.4 GHz GPGPU**
Спутниковое изображние	6,91	4,87	0,016	0,03
Стереопара общего типа (пейзаж)	1,64	1,64	0,009	0,024
Стереопара с коротким ГРИП (макроснимок)	12,61	8,46	0,027	0,052

В ходе вычислительного эксперимента использование композиционного метода позволило сократить число ошибок на карте диспаратности, а также уменьшить время обработки за счет использования GPGPU.

Заключение

В ходе работ исследована возможность использования цветовой информации изображений при поиске соответствий между пикселами изображений в задаче построения карты диспаратности. Исследована специфика использования RGB-модели и приведения ее к ортогональной XYZ-модели. Проведен вычислительный эксперимент на реальных изображениях, который показал значительное улучшение результатов построения карт диспаратности по сравнению с методами, основанными на использовании только интенсивностей пикселов изображений.

Также произведено изучение возможности использования оператора градиента в целях поиска информативных областей изображений, определения размерности сканирующего окна при построении карт диспаратности, и рассмотрен оригинальный подход к обработке малотекстурированных областей. Кроме того, проведено тестирование алгоритма на стереопарах для классических вычислительных систем различной производительности и вычислительной системы на основе GPGPU, в результате которого выявлено значительное преимущество GPGPU-систем для решения данных задач.

Литература

1. Borodach A., Tuzikov A. Automatic determination of matching points on two images // Proceedings of the 9th International Conference “Pattern Recognition and Information Processing”. 22-24 May, 2007, Minsk, Belarus. Vol. 1. P. 49-53.
2. Волкович А.Н. Использование цветовых характеристик при построении карт диспаратности // Материалы международного конгресса РОПИ-2011. Нижний Новгород: ННГУ, 2011. С. 112-117.

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.
4. Тузиков А.В., Шейнин С.А., Жук Д.В. Математическая морфология, моменты, стереобработка: избранные вопросы обработки и анализа цифровых изображений. Минск, Белорус. наука, 2006. – 198 с.

ГРИД-ЦЕНТР ИПХФ РАН ДЛЯ КОМПЕТЕНЦИИ ПРИКЛАДНОГО ПО И ПРОВЕДЕНИЯ КВАНТОВО-ХИМИЧЕСКИХ И МОЛЕКУЛЯРНО-ДИНАМИЧЕСКИХ РАСЧЕТОВ НА ОСНОВЕ «ГИБРИДНЫХ» И РАСПРЕДЕЛЕННЫХ ТЕХНОЛОГИЙ

В.М. Волохов[1], Д.А. Варламов[1,2], А.В. Пивушков[1], А.В. Волохов[1], Г.А. Покатович[1]

[1]Институт проблем химической физики РАН, Черноголовка

[2]Институт экспериментальной минералогии РАН, Черноголовка

Описаны перспективы создания в ИПХФ РАН проблемно-ориентированного грид-центра квантово-химического и молекулярно-динамического прикладного ПО, а также проведения расчетов с широким использованием «гибридных» и распределенных технологий вычислений. Центр предназначен для компетенции широкого спектра прикладных пакетов в области химии и смежных наук и проведения ресурсоемких расчетов с их использованием, а также организации упрощенного доступа пользователей через клиентские интерфейсы высокого уровня к высокопроизводительным вычислительным ресурсам. Помимо общей компетенции большого количества прикладных пакетов будут созданы доступные веб- и грид-ориентированные вычислительные сервисы для квантовой химии и молекулярной динамики.

Введение

Вычислительная (прежде всего квантовая) химия, молекулярная динамика и молекулярное моделирование, а также сопряженные с ними области знаний являются наиболее заинтересованными в высокопроизводительных вычислениях отраслями науки. Исследования, проводимые в области химии и смежных наук, в настоящее время, как правило, неэффективны без использования сверхмощных параллельных и распределенных вычислительных ресурсов для решения задач самых разных классов [1].

Современное мировое состояние вычислительной химии характеризуется тотальным использованием мощных вычислительных ресурсов – как параллельных (включая «гибридные» на основе графических процессоров и мультиядерных сопроцессоров), так и распределенных – для решения задач различных классов. В настоящее время в крупнейших мировых научных центрах, специализирующихся в теоретической химии и теории химических превращений, ведутся активные работы по разработке наиболее эффективных вычислительных процедур для параллельных, «гибридных» и распределенных вычислительных сред, позволяющих исследовать самые разнообразные химические процессы.

Потенциально задачи в области молекулярного моделирования и многоуровневого иерархического моделирования материальных объектов (от квантово-механического уровня до уровня сплошных сред и конструкций) могут выходить на уровень востребованности многих петафлопс. Некоторые задачи оптимизации крупных молекулярных структур требуют выполнения до 10^9 отдельных расчетов. Типичный докинг белковых лигандов с размерностью 200 атомов × 300 000 конфигураций × 1000 CPU-часов требует до 300 Пф вычислительных мощностей. Для построения многомерных потенциальных поверхностей, адекватно описывающих химические реакции, нужно провести 10^2 – 10^N независимых весьма ресурсоемких *ab initio* расчетов. Детальное моделирование даже отдельных этапов каталитических реакций в топливных элементах на основе водорода требует до 60000 CPU-часов. Для исследования динамики не очень сложной

химической реакции с использованием метода классических траекторий зачастую нужно рассчитать до 10^7-10^9 независимых траекторий. Численное исследование многопараметрических функций $F(x_1, x_2 \ldots x_n)$ в области изменения параметров x_1, $x_2 \ldots x_n$ при разбиении диапазона изменения каждого параметра на 10 ячеек требует проведения 10^N независимых расчетов F, и так далее.

Востребованность вычислительной химией все возрастающих вычислительных ресурсов подтверждается тем, что большинство суперкомпьютерных центров США и Европы предоставляют до 40% вычислительных мощностей для нужд биохимии, молекулярного моделирования, квантовой химии, нанотехнологических расчетов. При этом существует устойчивая тенденция на создание проблемно-ориентированных суперкомпьютерных центров (как вариант – выделение фиксированных пулов ресурсов в рамках петафлопсных суперкомпьютеров), специализирующихся исключительно на квантово-химических и молекулярно-динамических расчетах. При этом эти ресурсы часто входят в различные грид-полигоны и образуют достаточно обширные виртуальные организации (типа BO Gaussian в консорциуме EGI), объединяющие на разных уровнях ресурсы и пользователей.

К сожалению, при наличии в Российской Федерации достаточно мощных вычислительных суперкомпьютерных центров (МГУ, МСЦ, Саров, ЮжУрГУ и т.п.) подобные проблемно-ориентированные центры достаточной мощности в настоящее время практически отсутствуют. Данная публикация предназначена осветить их актуальность, а также предлагаемые технологии создания и аспекты эффективной эксплуатации.

1. Основы для создания грид-центра компетенции

Ранее, в результате проводимых в 2004-2012 годах в ИПХФ РАН (Научный центр Черноголовка) исследовательских работ в области распределенных вычислений был сформирован прототип предлагаемого к созданию проблемно-ориентированного вычислительного грид-центра. Работы с квантово-химическими прикладными пакетами в условиях суперкомпьютерных установок и грид-полигонов распределенных вычислений в ИПХФ РАН осуществлялись в течение более чем десятилетия по программам Президиума РАН, федеральным целевым научно-техническим программам, грантам РФФИ и Министерства науки и образования, программам Союзного Государства РФ-РБ [2, 3]. Исследования включали: 1) создание ресурсных сайтов ряда грид-полигонов (EGEE-RDIG/EGI-[RU-NGI], СКИФ-Полигон, Национальная нанотехнологическая сеть, Российская грид-сеть для высокопроизводительных вычислений); 2) адаптацию прикладного ПО в области вычислительной химии к работе на суперкомпьютерах и в различных грид-инфраструктурах, включая создание высокоуровневых грид-сервисов; 3) создание и апробацию новых методов вычислений на суперкомпьютерах и в распределенных средах, связанных со спецификой используемого ПО [4]; 4) проведение с использованием созданной инфраструктуры широкомасштабных вычислений в интересах пользователей грид-полигонов.

В 2012-2013 годах на базе созданного пула «гибридных» вычислительных узлов была проведена первичная компетенция квантово-химических и молекулярно-динамических пакетов, адаптированных на проведение расчетов с использованием GPU сопроцессоров, а также проведен ряд расчетов в области моделирования топливных элементов, молекулярной динамики, расчета кристаллических структур [2, 4].

В 2013 году в распоряжение авторов поступил вычислительный кластер с пиковой производительностью около 15 Тф (176 двухпроцессорных узлов HP Proliant на основе 4- и 6-ядерных процессоров Intel Xeon 5450 и 5670 частотой 3 ГГц и оперативной памятью 8 и 12 Гбайт – всего 1472 ядра; коммуникационная сеть Infiniband DDR на основе коммутаторов Voltaire и Cisco – скорость двунаправленного обмена данными 1400

Мбайт/с, латентность 3.2-4.5 мкс; транспортная и управляющая сети – Gigabit Ethernet; жесткие диски – не менее 36 Гбайт на узел), а также мини-кластер на базе двух высокопроизводительных «гибридных» узлов (1 узел – 12 вычислительных ядер [2x6 3.46GHz Intel® Xeon® X5675], 48 Гб RAM, 3 Тб HDD/SSD, 2 GPU Nvidia Tesla C2075, 3 сети Gigabit Ethernet, пиковая производительность узла – до 1.3 Тф для вычислений двойной точности и 2.3 Тф для вычислений ординарной точности). Пиковая производительность данного пула составляет около 2.6 Тф.

На базе данного оборудования начато создание (с использованием существующего прототипа) высокопроизводительного проблемно-ориентированного грид-сайта, направленного на решение «тяжелых» задач квантовой химии и молекулярной динамики с применением новейших технологий параллельных, распределенных и «гибридных» вычислений, а также на разработку и внедрение новых оригинальных методов вычислений для решения и оптимизации подобных задач.

2. Задачи центра

В процессе исследований давно стала очевидной потребность в создании высокопроизводительного ресурсного центра, узкоспециализированного на использовании в интересах вычислительной химии и смежных с ней областей науки.

На основе опыта предыдущих работ предложены следующие направления и методы, используемые для создания и успешной работы такого центра:

1) Отбор наиболее востребованных пользователями квантово-химических и молекулярно-динамических программных прикладных пакетов (ППП) – свободно распространяемых и лицензируемых, их жесткая конкуренция между собой, определение возможностей работы в качестве грид- и веб-ориентированных вычислительных сервисов, а также уровня реализации в них новейших вычислительных технологий;
2) Установка и отладка максимально широкого спектра выбранного ПО, эксплуатация его в интересах пользователей в рамках локального ресурса для решения научно-практических задач;
3) Разработка и внедрение в практику квантово-химических расчетов новейших вычислительных технологий («гибридные» вычисления на базе графических ускорителей и мультиядерных сопроцессоров, виртуализация вычислений и т.п.), использование адаптированных к данным технологиям версий ППП;
4) Интеграция центра в доступную грид-среду (в том числе с созданием соответствующих ресурсных сайтов и виртуальных организаций), адаптация прикладного ПО в качестве грид-сервисов для решения входящих задач, создание пользовательских интерфейсов для запуска ППП в качестве исходящих заданий грид-полигона;
5) Создание на базе центра веб-портала с интегрированными в него высокоуровневыми грид- и веб-интерфейсами к прикладным пакетам, предназначенными для формирования, запуска и мониторинга локальных и грид-заданий, и последующей доставки результатов пользователям. Портал обеспечивает корректную авторизацию пользователей, хранение их данных, учет и выбор ресурсов, а также поддерживает многие другие функции, максимально повышающие эффективность работы конечных пользователей.

Основным подходом является комбинированное использование новейших общеупотребительных технологий и методов проведения высокоинтенсивных расчетов (в первую очередь «гибридных» вычислений) совместно с ранее разработанными авторами методами расчетов в грид-средах. Результатом этого должно стать создание сбалансированного проблемно-ориентированного ресурса, объединяющего комплекс разработанных методов расчетов, центр компетенции ПО, собственно высокопроизводительный вычислительный ресурс (ориентированный на решение задач квантовой химии и

молекулярной динамики) с поддержкой «гибридной» модели вычислений, полноценный грид-сайт и высокоуровневые веб-интерфейсы.

3. Преимущества создания центра и проблемы его эксплуатации

Преимущества создания подобных центров очевидны:

1) создание пулов узлов, ориентированных именно на «тяжелые» квантово-химические вычисления (большие объемы RAM и HDD на ресурсных узлах, специфика коммуникационной среды и т.п.);
2) возможность установки и тестирования очень широкого спектра прикладных пакетов в выбранной области;
3) квалифицированная поддержка пользователей системными специалистами, хорошо знакомыми с особенностями конкретного прикладного ПО, а также ориентирующимися в весьма сложных системах входных данных и многозначности стратегии вычислений;
4) создание специализированных высокоуровневых интерфейсов к прикладному ПО для работы с локальными и грид-заданиями;
5) легкость (по сравнению с суперкомпьютерами общего назначения) интеграции в различные грид-полигоны с уже готовыми прикладными вычислительными сервисами, адаптируемыми к грид-средам.

Отдельно следует указать, что стремительное развитие «гибридных» (CPU+GPU или CPU+ускорители типа Intel Phi) вычислительных технологий достигло точки, когда множество существующих приложений в различных областях реализуются с их использованием и работают значительно быстрее, чем на обычных многоядерных системах. В настоящее время резко интенсифицировался перевод программного кода прикладных пакетов на «гибридные» технологии вычислений. При этом программирование обновленного или создаваемого заново «гибридного» кода для квантово-химических и молекулярно-динамических пакетов занимает одно из ведущих позиций в этом направлении. В течение последних лет существенно упростилось программирование соответствующей модели параллельных вычислений с использованием среды программирования CUDA™, которая обеспечивает набор абстракций, позволяющих выражать как параллелизм данных, так и параллелизм задач. Это позволяет проводить высокоинтенсивные вычисления с применением встраиваемых в расчетные узлы кластеров высокопроизводительных графических ускорителей (типа Nvidia Tesla и Kepler) и многоядерных сопроцессоров Intel Xeon Phi, что значительно повысит эффективность использования параллельных методов расчетов и снизит как требования к расчетным центрам, так и операционную стоимость расчетов, особенно при использовании большого числа расчетных узлов. Применение методов параллелизации с использованием GPU-устройств или ускорителей Intel Xeon Phi для прикладных пакетов в области квантовой химии и молекулярной динамики в большинстве случаев ведет к повышению эффективности расчетов от десятков процентов до нескольких порядков (PetaChem, Gromacs, NAMD, VMD) либо (как вариант) дает возможность существенного повышения точности или детальности расчетов.

К сожалению, в России при наличии в настоящее время достаточного количества ресурсов с технической поддержкой «гибридных» технологий пока гораздо меньше времени уделяется адаптации и тестированию прикладного ПО (в том числе с исходными кодами) к работе в подобных условиях, что в значительной мере снижает эффективность использования дорогостоящих ускорителей.

Заключение

Создаваемый проблемно-ориентированный грид-центр (базирующийся на достаточно мощном, минимум первые десятки терафлопс, вычислительном ресурсе) направ-

лен на разработку и внедрение новейших вычислительных (параллельных, распределенных, «гибридных») технологий с использованием широкого спектра прикладных пакетов. Он позволит решать на принципиально новом вычислительном уровне многочисленные «тяжелые» задачи в следующих областях: строение молекул и структура твердых тел, кинетика и механизмы сложных химических реакций, химическая физика процессов горения и взрыва, процессов образования и модификации полимеров, биологических процессов и систем, общие проблемы химической физики. Специализированный центр позволяет также разрабатывать новые оригинальные методы проведения локальных и распределенных вычислений.

Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, соглашение №8026 от 10.07.2012 г.

Литература

1. Суперкомпьютерные технологии в науке, образовании, промышленности. Вып. 3 (под ред. В.А.Садовничего). – М.: Изд-во МГУ, 2012. – 232 с.
2. Волохов В.М., Варламов Д.А., Пивушков А.В., Волохов А.В. Применение GRID технологий в области вычислительной химии // Известия Академии наук. Серия химическая. 2011. № 7. С. 1483-1490.
3. Волохов В.М., Варламов Д.А., Волохов А.В., Пивушков А.В., Покатович Г.А., Сурков Н.Ф. Грид-сервисы в вычислительной химии: достижения и перспективы // Вестник Уфимского государственного авиационно-технического университета. Серия Управление, вычислительная техника и информатика. 2011. Т.15, № 5 (45). С. 161-169.
4. Пивушков А.В., Волохов В.М., Варламов Д.А., Волохов А.В. Применение новых вычислительных технологий для повышения эффективности расчетов в грид-средах // Вестник Уфимского государственного авиационно-технического университета. Серия Управление, вычислительная техника и информатика. 2013. Т.17, № 2 (55). С. 117-124.

ПРИМЕНЕНИЕ КОНЕЧНЫХ АВТОМАТОВ ДЛЯ УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СЕТЬЮ РОБОТОВ

И.В. Воронин

Институт проблем лазерных и информационных технологий РАН, Шатура
E-mail: woronin05@yandex.ru

В данной работе обсуждаются вопросы применения различных алгоритмов управления распределенными устройствами на основе конечных автоматов. Анализируются примеры выполнения алгоритмов, с использованием которых происходит управление элементами распределенной сенсорной сети, интегрированных в роботизированные платформы. Выносится на дискуссию проект роботизированной платформы УМКИ, основанный на свободном программном обеспечении, что позволяет с минимальными затратами модернизировать его и развивать функционал.

ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ MPI/OPENMP-РЕАЛИЗАЦИИ С ВЫДЕЛЕНИЕМ ПОТОКОВ-«ПОЧТАЛЬОНОВ» ТРЕХМЕРНОЙ СХЕМЫ РАСЩЕПЛЕНИЯ ДЛЯ ЗАДАЧИ ТЕПЛОПЕРЕНОСА

***К.В. Воронин*[1,2]**

[1]*Институт вычислительной математики и математической геофизики СО РАН*
[2]*Новосибирский госуниверситет*

Представлены результаты исследования параллельных алгоритмов реализации векторных схем расщепления на основе технологий MPI и OpenMP для решения трехмерных задач теплопереноса. Проводится сравнение параллельных алгоритмов, использующих только MPI, простой MPI/OpenMP и MPI/OpenMP с выделением потоков-«почтальонов». Основная идея MPI/OpenMP-алгоритма с «почтальонами» заключается в выделении на каждом из узлов с общей памятью одного OpenMP-потока, отвечающего за реализацию обмена данными между процессами. При использовании такого подхода вычисления выполняются одновременно с обменами данными. Результаты проведенного исследования позволили заключить: несмотря на то, что использование «гибридного» подхода с выделением потоков-«почтальонов» позволяет значительно ускорить эффективность «прямолинейного» MPI/OpenMP-варианта, такой подход уступает «чистой» MPI-реализации для рассматриваемого класса алгоритмов.

Введение

При решении больших прикладных задач математического моделирования физических процессов естественным образом возникает потребность в использовании современных суперкомпьютеров с общей и разделенной памятью. Целью использования суперкомпьютеров является не только преодоление ограничения на оперативную память на одном вычислительном узле, но и ускорение общего времени выполнения алгоритма за счет распараллеливания вычислений на все доступные ядра. Одним из наиболее распространенных подходов является использование технологии MPI для проведения расчетов на системах с распределенной памятью. В то же время современные вычислительные узлы на самом деле всегда состоят из нескольких вычислительных ядер с общей памятью. Поэтому кажется естественным использовать «гибридные» алгоритмы на основе MPI и OpenMP, где коммуникации между узлами осуществляются с помощью MPI, а внутри узлов работают OpenMP-потоки на общей памяти. Достаточно хорошо известно, что «чистая» MPI-реализация для алгоритмов, где распределение данных по процессам возникает естественным образом (явные схемы для параболических уравнений, метод разделения переменных для эллиптических уравнений, методы декомпозиции области и т.п.), показывает высокую эффективность (почти линейное шкалирование времени от числа MPI-процессов). Для достижения сравнимой производительности для OpenMP-реализации внутри одного вычислительного узла обычно осуществляется значительная модификация кода с введением локальных массивов для каждого из OpenMP-потоков. В данной работе исследован другой способ повышения эффективности MPI/OpenMP-реализации - основная идея заключается в выделении на каждом из узлов с общей памятью одного потока («почтальона»), отвечающего за выполнение обменов данными между процессами (между узлами – MPI). Данный подход позволяет организовать одновременное выполнение полезных вычислений и обменов данными

[1]. Для этого необходимо дополнительно разбить данные на каждом узле на части меньшего размера. Эффективность подобной гибридной MPI/OpenMP-реализации сильно зависит, в том числе, и от размера этих частей, который подбирался (экспериментально) оптимальным для рассматриваемых задач. В итоге, общее время работы алгоритма сводится к максимуму из времени, затрачиваемого на вычисления, и времени, затрачиваемого на обмен данными.

Исследование эффективности такого гибридного подхода с выделением потоков-«почтальонов» было проведено для трехмерной схемы расщепления в смешанном методе конечных элементов в задачах теплопереноса [2, 3]. Одной из характерных особенностей данной схемы является то, что благодаря использованию параллелепипедальных сеток и конечных элементов низкого порядка реализация данной схемы на каждом временном шаге сводится к выполнению (на каждом из дробных шагов) независимых прогонок вдоль координатных линий сетки. Данная схема является лишь одним из примеров векторных схем расщепления для задачи теплопереноса, записанной в терминах «температура – тепловой поток», поэтому все сделанные выводы об эффективности использования гибридных алгоритмов имеют достаточно общий характер и распространяются на весь класс схем, предложенных в рамках упомянутого подхода. В настоящее время векторные схемы расщепления в смешанном МКЭ применяются, например, для моделирования геотермальных процессов в литосфере [4].

Тезисы организованы следующим образом – в первом разделе приведены кратко постановка задачи и вид трехмерной схемы расщепления, для которой был построен параллельный алгоритм. Во втором разделе описаны нумерация данных и схема распределения данных по MPI-процессам, а также пересылки, необходимые для реализации одного временного шага схемы. В разделе 3 изложены основные особенности исследуемой гибридной MPI/OpenMP-реализации. Наконец, в разделе 4 представлены сравнительные результаты, полученные при проведении численных экспериментов на кластере Сибирского суперкомпьютерного центра (ССКЦ СО РАН) [5], для гибридной реализации с «почтальонами», прямолинейной MPI/OpenMP-реализации и «чистой» MPI-реализации. В заключении еще раз формулируются основные результаты работы.

1. Постановка задачи и схема расщепления

Рассмотрим следующую систему дифференциальных уравнений первого порядка, описывающую процесс распространения тепла в трехмерной области Ω:

$$\begin{aligned} c_p \rho \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{w} &= f \\ \frac{1}{\lambda} \mathbf{w} &= -\nabla T \end{aligned} \qquad x \in \Omega, t > 0.$$

Здесь T и $\mathbf{w}$ – искомые функции температуры и теплового потока, c_p, ρ и λ – коэффициенты теплоемкости, плотности и теплопроводности соответственно, f – распределенные внутри области источники тепла. Первое уравнение представляет собой закон сохранения энергии, второе – закон Фурье. К системе присоединяются начальные данные для температуры, на границе ставятся краевые условия Неймана или Дирихле. После стандартных преобразований можно получить смешанную слабую (по пространству) постановку задачи

$$\begin{aligned} \int_\Omega c_p \rho \frac{\partial T}{\partial t} q dx + \int_\Omega \nabla \cdot \mathbf{w} q dx &= \int_\Omega f q dx, \forall q \in L_2(\Omega) \\ \int_\Omega \frac{1}{\lambda} \mathbf{w} \cdot \mathbf{u} dx &= \int_\Omega T \nabla \cdot \mathbf{u} dx - \int_{\partial\Omega} T \mathbf{u} \cdot \mathbf{n} d\gamma, \forall \mathbf{u} \in \mathbf{H}_{div}(\Omega) \end{aligned},$$

где искомые функции T и w ищутся как элементы функциональных пространств $C^1(0,T;L_2(\Omega))$ и $C(0,T;\mathbf{H}_{div}(\Omega))$ соответственно. Для пространственной аппроксимации применяется смешанный метод конечных элементов с элементами Равьяра-Тома для вектор-функций и кусочно-постоянными элементами для скаляров.

Таким образом, возникает следующая система обыкновенных дифференциальных уравнений

$$M\frac{\partial T_h}{\partial t}+\mathbf{B}\mathbf{w}_h=F_h \quad t>0,$$
$$\mathbf{A}\mathbf{w}_h=\mathbf{B}^T T_h$$

где матрица масс для потока $\mathbf{A}$ – блочно-диагональная с трехдиагональными блоками, M – диагональная матрица масс для температуры, прямоугольные матрицы $\mathbf{B}$ и $\mathbf{B}^T$ соответствуют операторам градиента и дивергенции.

Использованная в данной работе схема расщепления основана на схеме Дугласа и Гана для сеточной дивергенции потока. Эта схема имеет второй порядок точности по времени и пространству и обеспечивает абсолютно устойчивое вычисление температуры [4]. Реализация схемы на одном шаге по времени сводится к пяти дробным шагам, на каждом из которых необходимо выполнять прогонки вдоль линий сетки, правая часть для прогонки аппроксимирует вторые смешанные производные по пространству от промежуточных потоковых переменных.

2. Распределение данных по MPI-процессам

Для всех сеточных функций используется следующая нумерация: скалярные величины нумеруются в порядке возрастания индексов *y* - *z* - *x*; *x*-компоненты векторных величин – в порядке возрастания индексов *x* - *y* - *z*, *y*-компоненты – *y* - *z* - *x*, *z*-компоненты – *z* - *y* - *x*. При такой нумерации естественным образом организовывается распределение данных по процессам - данные «нарезаются» по внешнему индексу.

Для реализации данной схемы требуется три обмена данными типа «all-to-all» на каждом шаге по времени – пересылки компонент потоков по *x*, *y* и *z*. Выполнение каждого из дробных шагов схемы сводится к обращению трехдиагональных матриц на ассемблируемых векторах правой части, т.е. к выполнению независимых прогонок вдоль линий сетки, что позволяет на каждом MPI процессе выполнять решение соответствующих систем маленького размера независимо. В силу несимметричности схемы выбор внешних индексов нумерации был оптимальным с точки зрения количества необходимых обменов данных на каждом временном слое.

3. MPI/OpenMP-реализация с «почтальонами»

Коммуникации между вычислительными узлами осуществляются с помощью процедур MPI, поэтому данные между вычислительными узлами распределяются так, как это описано в предыдущем разделе. На каждом из вычислительных узлов используется распараллеливание с помощью OpenMP. Главным для исследуемой MPI/OpenMP-реализации с «почтальонами» является организация наложения вычислений и обменов данными на каждом из вычислительных узлов. С этой целью при выполнении прогонок вдоль каждого из направлений данные на каждом процессе дополнительно разбиваются на части меньшего размера. Каждая такая часть обрабатывается параллельно потоками-«решателями», в то время как поток-«почтальон» выполняет динамически пересылку обработанных на каждом MPI-процессе к текущему моменту времени частей.

Выбор размера маленькой части для обработки каждым из OpenMP-потоков-«решателей» должен быть оптимальным с точки зрения следующих ограничений. С

одной стороны, слишком маленький размер блока может привести к зависимости по памяти и многократной перекачке данных между кэшами каждого из потоков, с другой – слишком большой размер блока увеличивает время возможного простоя из-за необходимости синхронизации и уменьшает возможный выигрыш во времени из-за уменьшения гибкости гибридной части алгоритма. При исследовании эффективности алгоритма размер маленькой части подбирался экспериментально. Также необходимо отметить, что, помимо разбиения процесса вычислений на части, следует разбивать и процесс сборки векторов правой части для прогонок, причем размер частей разбиения для правой части также зависит от размера кэша и от размера блоков для вычислений (для тех слагаемых, в которых участвуют данные, полученные в ходе межузловых коммуникаций).

Кратко основная идея подобной MPI/OpenMP-реализации может быть также сформулирована в виде следующего псевдокода:

```
if (my_thread_ID != 0) {
 // "solver"-threads:
 for (int count = 0; count < chunks_number; count++) { /* loop for chunks */
 #pragma omp parallel
 /* solving each small chunk in parallel */
 foo1();
 }
 #pragma omp flush
 /*telling the "postman"-thread that several chunks are ready to be exchanged*/
 foo2();
}
else { //my_threadID = 0
 // "postman"-thread:
 while /*not all chunks are exchanged */ {
 #pragma omp flush
 /* checking if there are ready chunks to be exchanged, if yes - transfer */
 foo3();
 }
}
```

4. Сравнение MPI-, MPI/OpenMP- и MPI/OpenMP-реализации с «почтальонами»

В данном разделе представлены результаты сравнения MPI-реализации с «прямолинейной» MPI/OpenMP- и MPI/OpenMP-реализацией с «почтальонами». Под «прямолинейной» MPI/OpenMP-реализацией подразумевается простое использование *#pragma omp* директив в MPI-коде. Представленные результаты были получены на кластере ССКЦ СО РАН [5] на двойных блейд-серверах HP BL2x220 G7, ОП модуля – 24 Гбайт, каждый узел состоит из двух 6-ядерных процессоров Intel Xeon X5670 2.93 GHz (Westmere). Расчеты проводились на тестовом решении, соответствующем правой части формулой $f = \left(f_{i,j,k}\right)$, где $f_{i,j,k} = h_x^i h_y^j h_z^k \frac{t}{t+1}\frac{i+j+k}{i+j+k+2}10^{-4}$.

Для гибридных MPI/OpenMP-реализаций число MPI-процессов совпадало с числом вычислительных узлов. Т.к. основное отличие от MPI-реализации состоит в использовании общей памяти внутри узла, использовались только два вычислительных узла. При этом для MPI/OpenMP-реализации с «почтальонами» был проведен ряд оптимизаций, связанных с локальной перенумерацией данных. На рис.1 приведены результаты сравнения MPI-, MPI/OpenMP- и MPI/OpenMP-реализаций для сетки 384x384x384, при этом оптимальный размер частей для реализации с «почтальонами», на которые делились данные внутри узла, был подобран экспериментально.

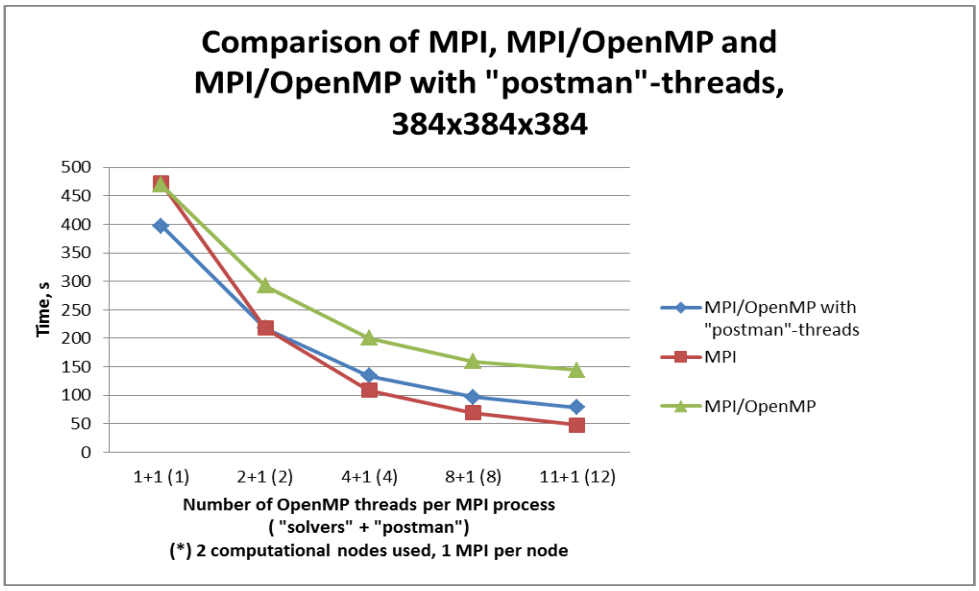


Рис. 1. Зависимость времени выполнения от числа OpenMP-потоков на вычислительном узле (числа MPI-процессов на узле для MPI-реализации), 2 вычислительных узла, сетка 384x384x384

Как можно заметить на рис. 1, наложение вычислений и обменов данными позволило существенно повысить эффективность простой MPI/OpenMP-реализации, однако MPI-реализация все равно показывает лучший результат. Это происходит не столько за счет возникновения дополнительных накладных расходов организации «гибридности» кода, сколько из-за того, что значительно увеличивается время, затрачиваемое на пересылки (т.к. в данном случае было использовано всего 2 MPI-процесса для MPI/OpenMP и 24 для MPI). Следует также заметить, что чистая MPI-реализация показывают очень высокую эффективность (80% при использовании 12 процессов на узле по сравнению с одним). Более того, при увеличении размера задачи шкалирование MPI-реализации еще улучшается. Необходимо отметить также одну из важных особенностей MPI/OpenMP-реализации с «почтальонами», а именно сильную зависимость эффективности от размера частей, на которые дополнительно делятся данные на каждом процессе. При этом априорный выбор оптимального размера требует тщательного анализа особенностей устройства общей памяти на вычислительном узле.

Заключение

Результаты проведенного численного исследования позволяют сделать вывод, что для рассматриваемого класса алгоритмов использование гибридного MPI/OpenMP-подхода с выделением потоков-«почтальонов» значительно улучшает «простую» (без локальных массивов для каждого OpenMP-потока) MPI/OpenMP-реализацию, но не приводит к повышению эффективности по сравнению с реализацией на основе MPI.

Данная работа поддержана грантами РФФИ №13-01-00019 и №12-01-31046.

Литература

1. Rabenseifner R., Hager G., Jost G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes // Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing. 427-436 (2009).

2. Воронин К.В., Лаевский Ю.М. Схемы расщепления в смешанном методе конечных элементов решения задач теплопереноса // Матем. Моделирование. 24(8). 109-120 (2012).
3. Воронин К.В., Лаевский Ю.М. О схемах расщепления в смешанном методе конечных элементов // Сиб. журн. вычисл. Математики. 15(2). 101-107 (2012)..4. Верниковская А.Е., Даценко В.М., Верниковский В.А., Матушкин Н.Ю., Лаевский Ю.М., Романова И.В., Травин А.В., Воронин К.В., Лепехина Е.Н. Эволюция магматизма и карбонатит-гранитная ассоциация в неопретерозойской активной континентальной окраине Сибирского кратона: термохронологические реконструкции // Доклады Академии наук. 448(5). 555-562 (2013).
5. Сибирский суперкомпьютерный центр – http://www2.sscc.ru/.

ПОДХОДЫ К ОРГАНИЗАЦИИ СИСТЕМЫ УПРАВЛЕНИЯ ЗАЩИЩЕННЫМИ КАРТОГРАФИЧЕСКИМИ БАЗАМИ ДАННЫХ

Р.Ф. Гибадуллин, С.В. Пыстогов

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

Рассматривается серверная часть системы управления защищенными картографическими базами данных на основе СУБД MySQL. Приводятся два архитектурных подхода к построению такой системы с применением вычислительных кластеров: монокластер и мультикластер. Выделяются и анализируются два способа построения монокластера. Даются предпосылки к разработке системы управления защищенными картографическими базами данных в среде PostgreSQL.

Введение

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

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.

1. Серверная часть системы управления защищенными картографическими базами данных на основе СУБД MySQL

Принципы формирования защищенных картографических баз данных для разных типов объектов подробно рассматриваются в работах [1] и [2].

На рис. 1 представлена серверная часть системы.

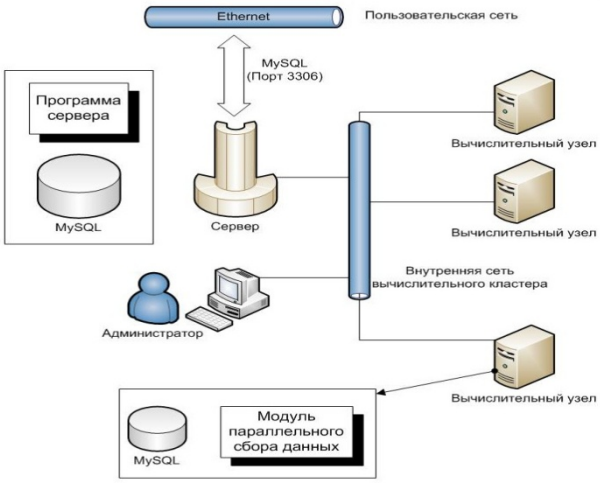


Рис. 1. Серверная часть системы

Как видно из рисунка, серверная часть – это вычислительный комплекс, состоящий из управляющего узла и множества вычислительных узлов. Управляющий 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, кроме того совместно выполняют роль распределенного хранилища.

К серверу по корпоративной сети подключается множество пользователей.

В системе за основу берется «интеллектуальная» файл-серверная организация взаимодействия. Сервер выполняет 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 (результат).

При построении такой системы с применением вычислительных кластеров можно выделить два архитектурных подхода.

– ***Монокластер*** («весь кластер на запрос»). В реализации данного подхода все вычислительные узлы серверной части в один момент времени обрабатывают только один запрос. Основные особенности:

- Все базы данных равномерно распределяются по узлам кластера.
- Каждый вычислительный узел выполняет поиск удовлетворяющих условиям запроса элементов в своей локальной части БД.
- Конечный результат получается путем конкатенации результатов со всех узлов, где были получены промежуточные результаты.
- Серверу нет необходимости выполнять функцию роутера. Запросы обрабатываются в порядке очередности их поступления на сервер.
- Возможна задержка барьерной синхронизации.

– ***Мультикластер*** («один узел на один запрос»): параллельное выполнение множества клиентских запросов на кластере. Система способна обрабатывать одновременно до N запросов, где N – число вычислительных узлов кластера. Особенности:

- Репликация баз данных по узлам. Каждый вычислительный узел хранит полную копию всех защищенных баз данных картографических сцен (ЗБД КС).
- На управляющий узел возлагаются дополнительные функции распределения запросов между узлами – функции роутера.
- Необходимость контролирования когерентности данных в ЗБД между узлами при изменении информации.

Можно выделить два способа построения монокластера:

1. Запрос отправляется для выполнения на каждый вычислительный узел кластера, где происходит полный просмотр всех фрагментов и поиск необходимых объектов. При этом, в случае селективного запроса, часть узлов кластера загружается «вхолостую».
2. На сервере хранится информация о расположении конкретных фрагментов по узлам вычислительного кластера. Селективный запрос направляется только на определенные узлы кластера, где просматриваются лишь предписанные фрагменты.

Для реализации способа 2 в разработанную структуру ЗБД КС необходимо ввести дополнительное отношение, которое хранит в себе пары связей «Номер фрагмента картографической сцены – Номер (адрес) узла кластера». Запросы для обработки берутся из очереди запросов, как и в способе 1. В ходе предварительной обработки, кроме выполнения раскрытия сокрытых параметров, синтаксического анализа и проверки прав пользователя на выполнение запроса, выполняется поиск узлов кластера с искомой информацией. Для этого используется табличное отношение с информацией о расположении всех фрагментов картографической сцены на узлах кластера.

Разработанные программные модули для обоих способов тестировались с использованием сгенерированных ЗБД КС. В качестве тестовых запросов выбраны:

1. select * from db; – выборка всех объектов сцены;
2. select * from db where x>1562340; – выборка участка сцены;
3. select * from db where x<710276; – выборка участка сцены.

Любой из них применим ко всем БД: DB1, DB2, DB3. Тестирование проводилось на 10 узлах суперкомпьютера КНИТУ-КАИ (с характеристикой узла: 2 Quad-Core Intel(R) Xeon(R) E(5450) (2*6 MB L2 cache, 3.00 GHz, 1333 MHz FSB), 32 GB Memory DDR2-667 MHz, 2*Ethernet 10/100/1000 BASE-T, 2*InfiniBand с пропускной способностью до 20 Gbit/s).

Полученные времена обработки выбранных запросов по первому и второму способам приведены в табл. 1. Для первого способа даны замеры времени при работе системы на кластере и на одном вычислительном узле (в скобках). Видно, что при выборке участков сцены второй способ более скоростной. Но при выборке данных всей сцены преимуществ не наблюдается.

Таблица 1. Замеры времени для способов 1 и 2 архитектуры монокластера

	DB1		DB2		DB3	
	Способ 1	Способ 2	Способ 1	Способ 2	Способ 1	Способ 2
Запрос 1	37 сек. (45 сек.)	39 сек.	41 сек. (57 сек.)	42 сек.	59 сек. (71 сек.)	60 сек.
Запрос 2	36 сек. (48 сек.)	17 сек.	43 сек. (54 сек.)	20 сек.	55 сек. (75 сек.)	26 сек.
Запрос 3	40 сек. (45 сек.)	18 сек.	50 сек. (59 сек.)	22 сек.	61 сек. (66 сек.)	28 сек.

Это объясняется тем, что в последнем случае поиск ведется по всем узлам. Использование вычислительного кластера по способу 1 дает лишь небольшой выигрыш по сравнению с обработкой тех же запросов на одном вычислительном узле, несмотря на десятикратное преимущество в вычислительных ресурсах.

Для дальнейшего ускорения процесса обработки потока запросов по способу 2 было предложено его развитие – способ 3 (рис. 2):

- На управляющем узле формируется начальная очередь клиентских запросов: n>N, где n – размер очереди, N – число узлов кластера (рис. 2, левый столбец).
- Производится предварительная обработка запросов согласно случаю 2 (рис. 2, правый столбец). Формируется очередь предобработанных запросов с информацией о целевых узлах кластера.

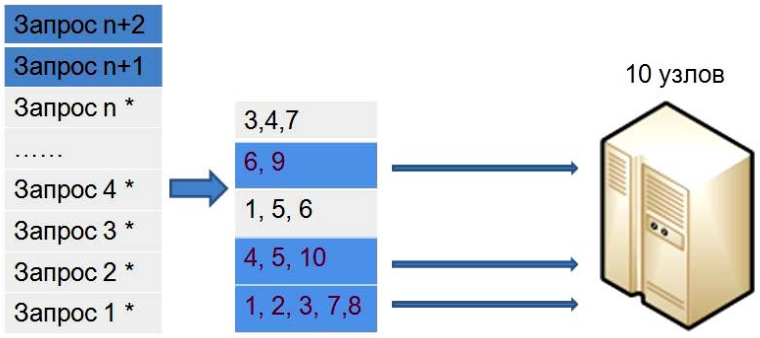


Рис. 2. Развитие способа 2

- Определяются запросы для начала обработки на кластере при возможной загрузке всех вычислительных узлов с учетом порядка следования запросов.

- Как только какой-либо из запросов будет обработан и освободит вычислительные узлы, в очереди предобработанных запросов будет искаться следующий запрос, удовлетворяющий условию максимальной загрузки кластера.
- После очередного считывания запроса из очереди происходит сдвиг, и новые запросы с номерами > n проходят предварительную обработку.

Были проведены эксперименты и для способа 3 на тех же БД. В качестве очереди запросов использовался пул из 20 запросов селективного типа. Шаблон запросов: select * from db{1-3} [where x {<,>,<=,>=} … [and y {<,>,<=,>=} …]]. Полученные результаты:

- Для точечных объектов: обработка 20 запросов без распределения потока запросов – 365 с, с распределением – 256 с. Выигрыш 30%.
- Для линейных объектов: без распределения – 403 с, с распределением – 280 с. Выигрыш 31%.
- Для площадных объектов: без распределения – 418 с, с распределением – 293 с. Выигрыш 33%.

Выигрыш времени на предварительной обработке нивелируется накладными расходами при поиске подходящих запросов для отправки на расчет.

2. Предпосылки к разработке системы управления защищенными картографическими базами данных в среде PostgreSQL

В целях коммерциализации разработанных модулей были проведены хоздоговорные работы с ООО «Геоинформационный центр «Зенит» (г. Казань). В рамках этих работ была установлена целесообразность построения системы управления защищенными картографическими базами с ориентацией на малые предприятия на базе одного многопроцессорного сервера и тонких клиентов. Правомерность этого отмечается и описанными выше результатами: рост производительности «в разы» от использования вычислительного кластера не наблюдается.

Для реализации такой системы в качестве базы выбрана СУБД PostgreSQL, которая обладает следующими преимуществами сравнительно с СУБД MySQL:

- обладает полным набором пространственных возможностей (соответствующих стандартам OGC [3]) и позволяет осуществлять любые виды операций над геоданными;
- включает поддержку пространственных индексов R-Tree/GiST, повышающую скорость обработки SQL-запросов к пространственным базам данных;
- поддержка баз данных практически неограниченного размера, надежность и устойчивость на очень больших нагрузках (что очень критично при организации системы на базе одного сервера);
- поддержка системы аутентификации Kerberos версий 4 и 5.

Таким образом инициирован проект по разработке системы управления защищенными картографическими базами данных в среде PostgreSQL. Ставятся и решаются следующие задачи:

1. Разработка модуля формирования защищенной картографической базы данных, включающая в себя полное множество атрибутов геоданных в соответствии со стандартом OGC.
2. Разработка параллельных модулей управления защищенной картографической базой данных, ориентированных на один вычислительный сервер с многоядерной архитектурой.
3. Разработка интерфейса управления защищенными картографическими базами данных с применением PostGIS и веб-интерфейса клиента для управления защищенной картографической базой данных, по сути реализация идеологии тонкого клиента.

По третьей задаче следует отметить, что использование связки PostgreSQL/PostGIS предоставляет довольно широкие возможности по работе с пространственными данными. Рассмотрим некоторые основы работы с PostGIS [4].

Таблицы метаданных OGC

При создании пространственной базы данных автоматически создаются две таблицы метаданных – SPATIAL_REF_SYS и GEOMETRY_COLUMNS. Они создаются в соответствии со спецификацией Open Geospatial Consortium Simple Features for SQL specification, выпущенной OGC и описывающей стандартные типы объектов ГИС, функции для манипуляции ими и набор таблиц метаданных.

Таблица GEOMETRY_COLUMNS хранит информацию о таблицах базы данных, содержащих пространственную информацию. Ее заполнение осуществляется вручную либо как следствие выполнения специальной процедуры OGC AddGeometryColumn().

Таблица SPATIAL_REF_SYS содержит числовые идентификаторы и текстовые описания систем координат, используемых в пространственной базе данных. Одним из полей этой таблицы является поле SRID – уникальный идентификатор, однозначно определяющий систему координат. SRID представляет из себя числовой код, которому соответствует некоторая система координат. Например, распространенный код EPSG 4326 соответствует географической системе координат WGS84.

Пространственные запросы

Для создания таблицы, содержащей пространственные данные, применим нижеприведенные SQL запросы:

create table points (pt geometry, name varchar);
insert into points values ('POINT(0 0)', 'Origin');
insert into points values ('POINT(4 0)', 'X Axis');
insert into points values ('POINT(0 3)', 'Y Axis');
select name, ST_AsText(pt), ST_Distance(pt, 'POINT(4 3)') from points;

В данном примере мы создали таблицу *points*, содержащую два поля: поле *pt* типа *geometry* и поле *name* типа *varchar*, после чего добавили (*insert*) в нее три записи, содержащие информацию о точках. Затем осуществили выборку с использованием функций PostGIS: *ST_Distance()* и *ST_AsText()*. В качестве параметра обе функции используют объект типа geometry. Функция *ST_Distance()* рассчитывает расстояние между двумя указанными точками плоскости, а *ST_AsText()* возвращает геометрию объекта в текстовом формате *WKT (Well-Known Text)*. Формат *WKT* включает информацию о типе объекта и координаты, составляющие объект.

Литература

1. 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.

2. Гибадуллин Р.Ф., Пыстогов С.В. Параллельная система управления полнообъектными защищенными базами данных картографических сцен // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2012): Материалы 12-й Всерос. конф. – Нижний Новгород: ННГУ, 2012. С. 91–95.

3. Стандарты Open Geospatial Consortium – [URL: http://www.opengeospatial.org].

4. Основы работы с PostGIS – [URL: http://gis-lab.info/qa/postgis-work.html].

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ЛАЗЕРНОГО ИЗЛУЧЕНИЯ С ПЛАЗМОЙ МЕТОДОМ ЧАСТИЦ В ЯЧЕЙКАХ

А.А. Голованов, Е.Н. Неруш, В.Ф. Башмаков, И.Ю. Костюков

Нижегородский госуниверситет им. Н.И. Лобачевского,
Институт прикладной физики РАН, Нижний Новгород

Описаны различные алгоритмы и реализующая их программа, используемая для моделирования взаимодействия лазерного излучения экстремально высокой интенсивности с веществом с учётом эффектов квантовой электродинамики.

Введение

Расчёт ускорения заряженных частиц в сверхсильных световых полях требует огромных затрат компьютерных ресурсов, и очень важно из всех методов моделирования выбрать наиболее эффективный. Так, использование конечно-разностных методов для решения уравнения для функции распределения частиц в плазме требует выделения памяти на описание огромной области в шестимерном пространстве (импульсы и координаты). При этом во многих задачах функция распределения соответствует горячей плазме с достаточно хорошо коллимированными пучками частиц с энергией, много большей средней температуры плазмы (например, в задаче ускорения электронов плазменной волной). Таким образом, почти всюду в шестимерной области моделирования функция распределения близка к нулю, и существенны лишь её значения в небольшом объёме. Напротив, в методе частиц в ячейках расчёт эволюции системы производится только в тех областях, где значения функции распределения существенны, поскольку в этом методе функция распределения аппроксимируется совокупностью квазичастиц, движущихся по тем же траекториям, что и реальные частицы, что дает колоссальный выигрыш в производительности по сравнению с конечно-разностными методами.

1. Используемые алгоритмы

В методе частиц в ячейках функция распределения представляется как сумма квазичастиц. Для расчёта плазменной динамики необходимо решать уравнения движения квазичастиц, а также уравнения, описывающие динамику электромагнитного поля. В основу разработанной программы были положены алгоритмы, используемые в коде VLPL [1] (например, явный алгоритм решения уравнений Максвелла NDFX, свободный от численной дисперсии в одном направлении). Однако для расчёта траекторий частиц был использован обладающий рядом преимуществ алгоритм, недавно предложенный в работе [2].

Остановимся подробно на процессе излучения (для примера – излучения фотонов электронами), описываемом в разработанной программе с использованием метода Монте-Карло. В данном методе специальная функция («генератор событий») на каждом шаге по времени и для каждого электрона с использованием генератора случайных чисел решает, следует ли производить излучение фотона данным электроном и какова энергия излученного фотона, если излучение происходит. Естественно, генератор событий должен быть устроен таким образом, чтобы в среднем давать распределение излученных фотонов, соответствующее реальной спектральной плотности вероятности излучения. В разработанной программе используется оригинальный подход, схожий с

методом Монте-Карло для численного интегрирования. Энергия излученного фотона лежит в пределах от 0 до энергии электрона, излучившего фотон. Спектральная плотность вероятности излучения в этом интервале может быть ограничена некоторым максимальным значением. Таким образом, геометрически имеем кривую – зависимость спектральной плотности вероятности излучения от энергии излучаемого фотона, вписанную в прямоугольную область с углами в точках (0,0) и (энергия электрона, выбранный максимум спектральной плотности распределения вероятности излучения). С использованием генератора случайных чисел можно генерировать точки, имеющие равномерное распределение в этой прямоугольной области. В методе численного взятия интеграла площадь под кривой вычисляется по количеству точек, попавших под кривую. В предложенном генераторе событий излучение фотона производится, если сгенерированная точка попадает под кривую, при этом энергия излученного фотона равна x-координате сгенерированной точки. Очевидно, что вероятность попадания случайно сгенерированной точки под кривую тем выше, чем выше (при данной частоте) значение спектральной плотности распределения вероятности излучения. Строго говоря, можно показать, что при соответствующем выборе максимального уровня спектральной плотности вероятности излучения и нормировок распределение излученных с использованием такого генератора событий фотонов даёт результат, достаточно близкий к реальному.

Таким образом, в разработанной программе на каждом шаге по времени для каждого электрона с использованием равномерного распределения вычисляются два случайных числа: частота излучения и число, сравниваемое с вероятностью излучения на данной частоте. Если это число оказывается меньше этой вероятности излучения, то производится излучение фотона – энергия электрона уменьшается, а в область моделирования добавляется новая квазичастица – фотон. С использованием данного генератора событий в программе описывается некогерентное синхротронное излучение гамма-квантов электронами, движущимися в сверхсильном лазерном поле. Данное излучение становится важно при интенсивностях выше 1023 Вт/см^2, поскольку приводит к существенным потерям энергии электронами.

2. Используемые технологии

Написанная программа использует многопоточность, что легко реализуемо за счет «локальности» выбранных алгоритмов. Под «локальностью» здесь подразумевается то, что величина полей в узлах сетки и кинематические величины квазичастиц на следующем шаге по времени зависят только от величины полей в узлах, соседних для данного узла или данной квазичастицы. При этом сохраняется конечная скорость распространения взаимодействий, за счёт чего область моделирования можно разделить на пересекающиеся подобласти, в каждой из которых можно производить расчёты независимо от других определённое время. Длительность этого промежутка времени выбирается меньшей, чем половина промежутка времени, необходимого для распространения возмущения через область пересечения. При таком выборе неверно будут посчитаны только значения полей в узлах сетки, близких к границе области. Однако за счёт обмена значениями полей в области пересечения можно восстановить правильные значения и продолжить вычисления. Таким образом, можно производить расчёты динамики в разных областях на разных процессорах, лишь изредка производя обмен небольшим количеством данных между ними (см. рис. 1), что идеально для создания многопотоковых программ.

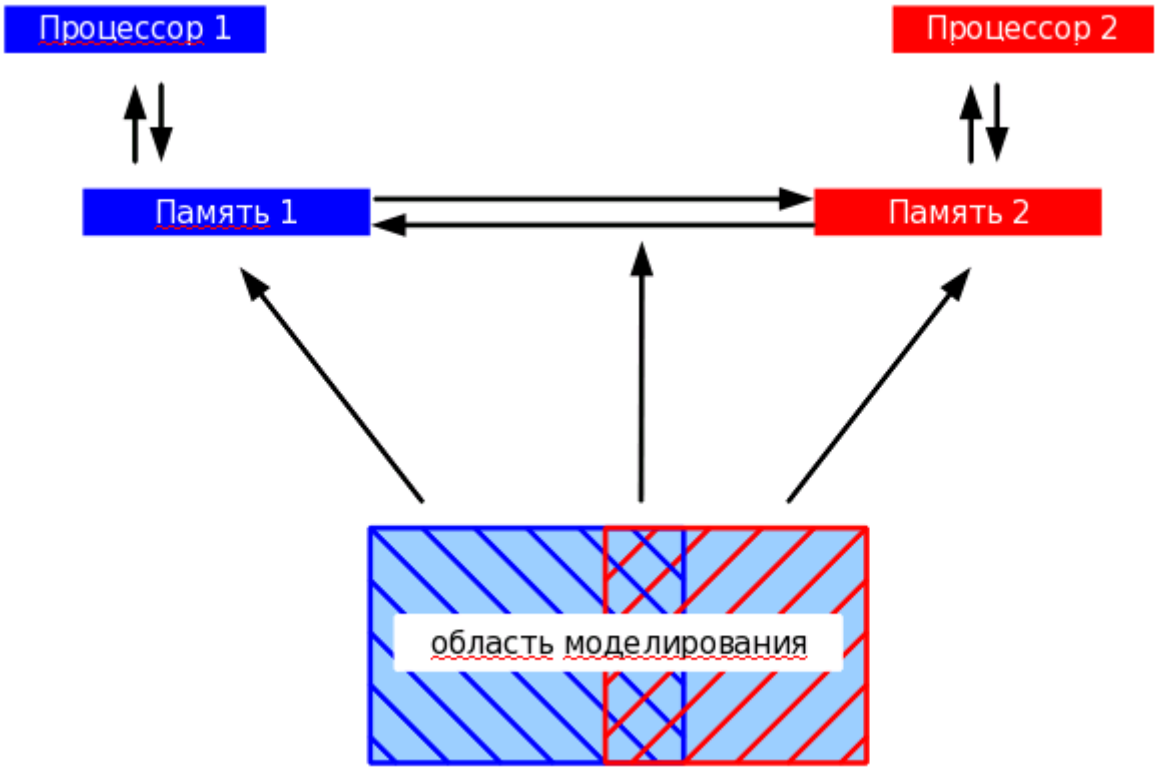


Рис. 1. Схема разделения области моделирования между двумя потоками в случае применения «локальных» численных схем

Для разделения программы на потоки были использованы технологии pthreads (POSIX threads) и NUMA (Non-uniform memory access, неоднородный доступ к памяти). Библиотека pthreads определяет набор функций и типов, позволяющий создавать и синхронизировать потоки, а также управлять ими.

Технология неоднородного доступа к памяти является в настоящее время очень перспективной. В архитектуре с неоднородным доступом к памяти для каждого процессора выделен отдельный канал для доступа к отдельной секции памяти (процессор со «своей» памятью называется узлом). Таким образом, доступ процессора к памяти внутри одного узла происходит быстрее и на большей скорости, чем доступ к памяти соседних узлов. При использовании технологии неоднородного доступа к памяти достаточно разбить область моделирования на части, для каждой из частей выделить память на отдельном узле и обрабатывать данные в каждой области отдельным потоком. Операционная система (использовалась ОС Linux) автоматически распределит потоки именно по тем процессорам, с которых доступ к памяти будет наиболее быстрым, то есть на каждом узле производится моделирование именно в той подобласти, данные для которой записаны в память данного узла. И лишь изредка производится обмен небольшим объёмом данных между узлами.

Таким образом, технология неоднородного доступа к памяти увеличивает пропускную способность памяти по отношению к технологии симметричного доступа к памяти, что оказалось существенным при использовании 2-процессорной 8-ядерной рабочей станции с двумя NUMA-узлами (получено двукратное ускорение выполнения программы).

Тестирование программы и последующее моделирование проводилось для параметров, близких к параметрам экспериментов по ускорению электронов, проводящихся на установке PEARL, созданной в ИПФ РАН, а также для параметров строящихся и проектируемых сверхмощных лазерных систем.

Работа поддержана грантом правительства Российской Федерации (проект No. 14.B25.31.0008).

Литература

1. Pukhov A. The virtual laser plasma laboratory // J. Plasma Phys. 1999. V. 61. No 10. P. 425-428.
2. Vay J.-L. Simulation of beams or plasmas crossing at relativistic velocity // Physics of Plasmas. 2008. 15. 056701.

ОТКРЫТАЯ ГРИД-СИСТЕМА НА ОСНОВЕ СЕМАНТИЧЕСКОЙ СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

Н.А. Гонтарь

Запорожский национальный технический университет

Открытая грид-система на основе семантической сервис-ориентированной архитектуры использует концептуальную модель, состоящую из грид-сервисов и семантического сервиса. Грид-сервисы имеют две спецификации: синтаксическую на языке GWSDL и семантическую на языке OWL-S. Семантический сервис проверяет согласованность и разрешимость предоставляемых онтологий и выполняет семантический поиск. Использование подобного представления увеличивает релевантность поиска необходимых сервисов, позволяет автоматизировать процесс композиции подходящих сервисов для решения поставленных задач, увеличивает эффективность использования сервисов и их интероперабельность, а также повышает прозрачность, расширяемость и адаптируемость грид-систем.

Введение

Грид можно рассматривать как сеть сервисов, которые предоставляют пользователям доступ к множеству распределенных вычислительных ресурсов [1]. Для эффективного использования грида необходимо обеспечить сервисам гибкость и интероперабельность, т.е. возможность эффективного взаимодействия вычислительных и информационных ресурсов, реализованных на разнородных программно-аппаратных платформах. Конвергенция сервис-ориентированной архитектуры (Service Oriented Architecture, SOA) и грид-систем воплощена Глобальным грид-форумом (Global Grid Forum, GGF) в открытой архитектуре грид-сервисов (Open Grid services Architecture, OGSA). Основной документ [2], фиксирующий архитектуру OGSA, описывает общие принципы построения сервис-ориентированного грида в терминах необходимых функциональных возможностей. OGSA определяет грид-сервис как веб-сервис, который предоставляет набор корректно определенных интерфейсов на языке GWSDL, и следует специфическим конвенциям для их создания и композиции сложных распределенных систем.

Программный язык, такой как GWSDL, рассматривает только синтаксическую сторону реализации и не отражает семантическую информацию о грид-сервисе [3]. Таким образом, пользователь должен предоставить или получить дополнительные разъяснения о сервисе. В настоящее время возможно применение семантических технологий (СТ) в интерпретации грид-сервисов, что сокращает усилия и повышает эффективность в управлении ресурсами и поиске необходимой информации [4].

Постановка задачи

Ввиду большого количества необходимых стандартов для создания открытой грид-системы на основе СТ актуальной является задача создания эталонной модели, отражающей семантическое описание грид-сервисов и регулирующей отношения между ними и пользователи на основе семантической сервис-ориентированной архитектуры (Semantic Service Oriented Architecture, SSOA) [5].

Наличие такой модели позволит разработчикам, пользователям и поставщикам программно-аппаратных средств использовать единообразный подход при решении соответствующих задач.

Основная часть

OGSA поддерживает основные принципы SOA, следовательно основана на взаимодействии между тремя основными сторонами: поставщиком услуг (Service Provider), который публикует описания сервиса и обеспечивает реализацию услуги; потребителем (Service Consumer), который может использовать универсальный идентификатор ресурса (URI) или найти описание сервиса в реестре; Service Broker, который обеспечивает и поддерживает реестр сервисов [6]. Метамодель, показывающая эти отношения, изображена на рис. 1.

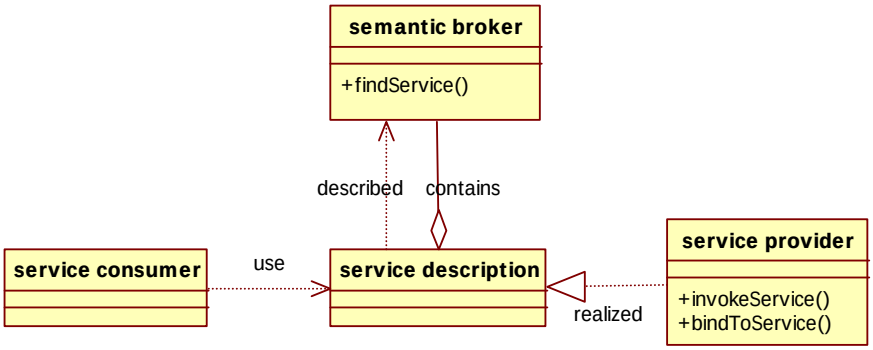


Рис. 1. Концептуальная модель SOA

Мы рассматриваем случай, когда в основе OGSA лежит SSOA, отличительной стороной которой является наличие семантического сервиса и онтологий [5]. Каждый грид-сервис реализует свою собственную онтологию. Семантический сервис производит релевантный поиск и верификацию предоставленных онтологий. В этом случае грид-сервис провайдера и потребителя может развиваться независимо от других грид-сервисов или их онтологий. При таком построении архитектуры сравнительно легко вносить изменения, добавлять новые сервисы и т.д. (улучшаются качественные характеристики архитектуры: расширяемость, масштабируемость, способность к эволюционным изменениям). Однако отсутствие общего словаря предметной области или онтологии верхнего уровня делает затруднительным процесс сравнения онтологий различных грид-сервисов. Концептуальная модель SSOA показана на рис. 2.

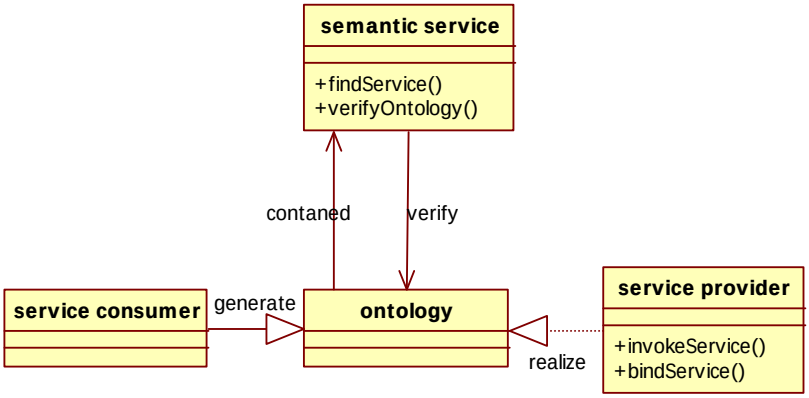


Рис. 2. Концептуальная модель SSOA

Создание онтологий позволяет получить в распоряжение семантического сервиса модель грид-сервиса, которая объединит его функциональные и нефункциональные характеристики. В таком случае необходимо с помощью технологий представления информации выразить шесть аспектов описания грид-сервиса (рис. 3). Metadata включает специфические нефункциональные описания грид-сервиса, такие как местоположение, характеристики, информация от провайдера и т.п. Usage описывает информацию об обслуживании грид-сервиса. IS-A отражает иерархическую структуру расположения грид-сервиса внутри грид-системы. Reference – ссылки на ресурсы, которые будут задействованы этим грид-сервисом. Input / Output указывает на входные / выходные данные.

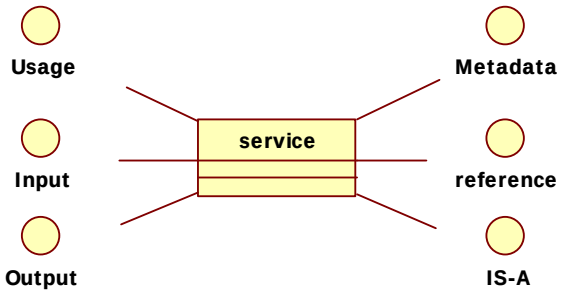


Рис. 3. Характеристики грид-сервиса

При создании сервиса для описания аспектов Input / Output достаточно использования языка GWSDL, а для всех характеристик мы используем семантический язык разметки OWL-S, который достаточно выразителен и непротиворечив. Источники информации для разработки описания грид-сервиса и последовательность действий показаны на рис. 4 [7].

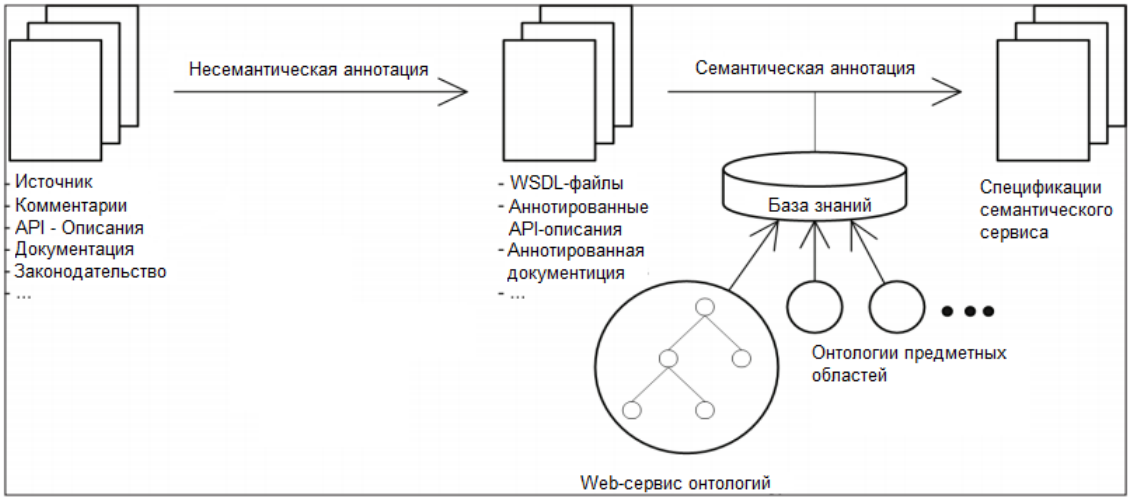


Рис. 4. Процесс создания SWS

Выводы

Таким образом, создание грид-сервиса в SSOA – это итерационный процесс, на каждом шаге которого используется необходимый язык аннотирования данных и информации. В результате грид-сервис имеет две спецификации: синтаксическую на языке GWSDL и семантическую на языке OWL-S. Использование подобного представления увеличивает релевантность поиска необходимых грид-сервисов, позволяет автоматизировать процесс композиции подходящих грид-сервисов для решения поставленных за-

дач, увеличивает эффективность использования грид-сервисов и их интероперабельность. Еще одним важным элементом SSOA является семантический сервис, который проверяет согласованность и разрешимость предоставляемых онтологий и выполняет семантический поиск.

Использование SSOA направлено на улучшение OGSA и эффективное взаимодействие грид-сервисов внутри системы. С использованием семантических аспектов и сервисной ориентации повышается прозрачность, расширяемость и адаптируемость грид-систем.

Литература

1. Bagherifard K. Measuring Semantic Similarity in Grids Using Ontology [Text] / K. Bagherifard, Mehrbakhsh Nilashi, Othman Ibrahim, Norafida Ithnin, and Lasisi Ayodele Nojeem // International Journal of Innovation and Applied Studies. Mar. 2013. ISSN 2028-9324. Vol. 2. No. 3. P. 230-237.
2. Foster I. Open Grid Services Architecture (OGSA) v1.0 [Electronic resource] / I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui and other, Jan. 2005. – Access mode: http://www.gridforum.org/documents/GFD.30.pdf.
3. Foster I. The Grid: Blueprint for a New Computing Infrastructure [Text] / I. Foster, C. Kesselman. – Publisher: Morgan Kaufmann Inc. 1999.
4. Ludwig S.A. Semantic approach to service discovery in a Grid environment [Text] / Simone A. Ludwig, S.M.S. Reyhani. Web Semantics: Science, Services and Agents on the World Wide Web. Jan. 2006. Vol.4. No.1. P. 1-13.
5. Гонтарь Н.А., Кудерметов Р.К. Организация семантической сервис-ориентированной архитектуры [Текст] // Системный анализ и информационные технологии: труды 14-й Международной научно-технической конференции SAIT 2012, Киев, 24 апреля 2012 г. / УНК “ИПСА” НТУУ “КПИ”. – К.: УНК “ИПСА” НТУУ “КПИ”, 2012. С. 337-339.
6. Arsanjani A. Service-oriented modeling and architecture. How to identify, specify, and realize services for your COA [Electronic resource] / Ali Arsanjani, Nov. 2004. – Access mode: http://www.ibm.com/developerworks/library/ws-soa-design1/.
7. Нестеренко А.С., Гонтарь Н.А. Создание семантического веб-сервиса // 17-й Международный молодежный форум «Радиоэлектроника и молодежь в XXI веке», г. Харьков (Украина), ХНУРЭ, 22 – 24 апреля 2013 г. Харьков: ХНУРЭ, 2013. С. 222-223.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ В ЗАДАЧАХ ОПТИЧЕСКОЙ ДИФФУЗИОННОЙ СПЕКТРОСКОПИИ НА ВЫЧИСЛИТЕЛЯХ С ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРОЙ

А.В. Горшков[1,2], М.Ю. Кириллин[2], В.П. Гергель[1]

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

Кратко описан алгоритм моделирования распространения зондирующего излучения методом Монте-Карло, адаптированный для решения задач оптической диффузионной спектроскопии. Дана оценка эффективности и масштабируемости параллельной версии алгоритма для систем с общей и распределенной памятью. Описаны результаты переноса и подходы к оптимизации алгоритма на графические процессоры и ускорители Intel Xeon Phi. Выполнено сравнение версий алгоритма для различных архитектур с точки зрения времени его работы.

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 компонент 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 этом объеме [1].

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.

Наиболее подходящими методами решения задач оптической диффузионной спектроскопии являются алгоритмы на базе метода Монте-Карло [2, 3]. Такие алгоритмы позволяют решать достаточно общие задачи моделирования распространения излучения в различных средах, включая биологические ткани, с учетом сложной геометрии исследуемых объектов. Основной недостаток таких методов – большая вычислительная

трудоемкость. Для повышения их эффективности исследователи работают как над оптимизацией алгоритмов моделирования [4-6], так и над использованием вычислителей с параллельной архитектурой [7-9].

1. Моделирование распространения излучения методом Монте-Карло

Идея алгоритма моделирования распространения излучения методом Монте-Карло состоит в трассировке набора фотонов в среде. Для улучшения качества метода рассматриваются не отдельные фотоны, а так называемые пакеты фотонов. Каждому пакету ставится в соответствие определенный начальный вес. Обычно, для упрощения расчетов, начальный вес принимают за единицу. Далее понятия фотона и пакета фотонов будут отождествляться.

Среда описывается набором слоев, каждый слой обладает рядом оптических характеристик и определенной геометрией границ.

Начальное положение и направление фотона определяется соответствующими параметрами источника излучения. Рассчитывается случайная величина, определяющая длину свободного пробега фотона в среде. На каждом шаге часть фотонов поглощается средой, что проявляется в уменьшении веса пакета фотонов на соответствующую величину. По окончании акта поглощения рассматривается рассеяние пакета фотонов. При этом фотоны меняют направление своего движения случайным образом.

Описанная последовательность шагов повторяется до тех пор, пока большинство фотонов пакета не будут поглощены либо не покинут исследуемый объект. В соответствие с тем, каким образом прекращается моделирование фотона, все фотоны можно разделить на 3 класса – диффузионно-отраженные, поглощенные и прошедшие (рис. 1).

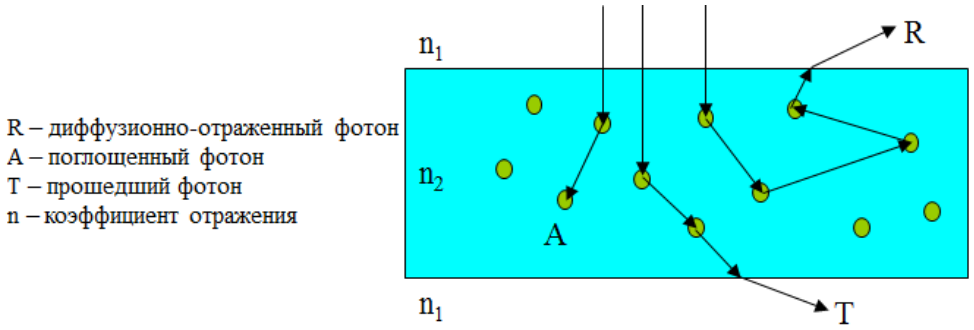


Рис. 1. Траектории движения и типы фотонов

В задачах оптической диффузионной спектроскопии дополнительно вводится понятие детектора – некоторой области на внешней границе среды, которая способна улавливать прошедшие через нее фотоны. И наиболее интересными являются траектории диффузионно-отраженных фотонов, прошедших через тот или иной детектор [10]. Полученная информация позволяет определить взаимное расположение источника и детекторов с тем, чтобы траектории фотонов, попавших в эти детекторы, проходили через интересующую исследователя область.

Отметим также, что для учета сложной геометрии тканей используется следующий подход. Объект моделирования разбивается на слои, границы которых представляются в виде триангулированных поверхностей. И на каждом шаге моделирования проверяется, пересекла ли траектория движения фотона границу текущего слоя. Для ускорения алгоритма поиска пересечения используются BVH-деревья [11].

2. Параллельный алгоритм для систем с общей и распределенной памятью

Методы Монте-Карло по своей природе хорошо распараллеливаются, так как проводимые в рамках метода испытания случайной величины обычно не зависят друг от

друга. В задаче моделирования распространения излучения трассировку фотонов также можно проводить независимо, что позволяет создавать эффективные параллельные алгоритмы для решения уравнения переноса излучения методом Монте-Карло.

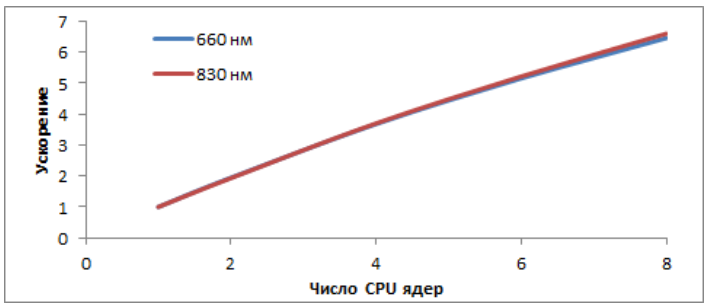


Рис. 2. Ускорение алгоритма моделирования распространения света на системах с общей памятью (2 процессора Intel Xeon L5630 (4 ядра в каждом), 24 ГБ RAM) при разных длинах волн источника излучения, 320 000 фотонов

Очевидным подходом к ускорению алгоритма моделирования является использование возможностей современных многоядерных процессоров. В силу высокой степени параллелизма задачи применение многоядерных CPU позволяется существенно сократить время моделирования. Проведенные эксперименты показывают, что эффективность распараллеливания описанного алгоритма на одном процессоре составляет 92%, а на двух CPU, установленных в рамках одного узла, 81%. Снижение эффективности во втором случае обусловлено увеличением числа запросов к оперативной памяти при постоянной ширине шины данных (рис. 2). Отметим, что все тесты проводились на геометрии головы человека, полученной по данным МРТ (~2 000 000 треугольников).

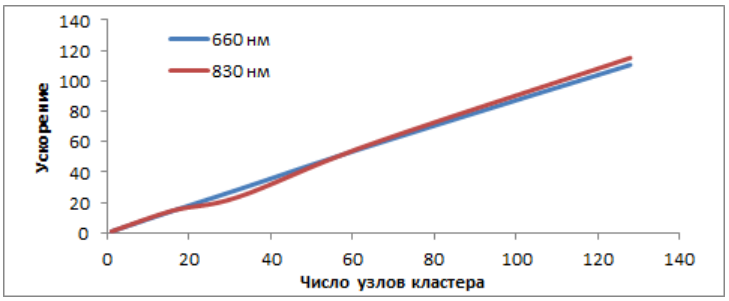


Рис. 3. Ускорение алгоритма моделирования распространения света на системах с распределенной памятью (128 узлов, на каждом 2 процессора Intel Xeon X5570, 12 ГБ RAM) при разных длинах волн источника излучения, 16 000 000 фотонов

Следующий шаг – использование систем с распределенной памятью. Принципиальной особенностью таких систем является необходимость организации обменов данными между отдельными узлами системы. Причем от степени эффективности обменов существенно зависит и скорость работы программы в целом. Отметим, что в рассматриваемой задаче передачу данных нужно делать только в начале и в конце работы алгоритма моделирования, а значит, данный алгоритм теоретически должен показывать хорошую эффективность распараллеливания и близкую к линейной масштабируемость.

Это подтверждается проведенными экспериментами. В частности, эффективность распараллеливания алгоритма на 128 узлах кластера составляет 90% (рис. 3).

3. Параллельный алгоритм для ускорителей Intel Xeon Phi

Intel Many Integrated Core (MIC) – архитектура многоядерной процессорной системы на базе x86 CPU ядер. Используется в сопроцессорах Intel Xeon Phi, предназначенных для ускорения задач общего назначения и составляющих в этом плане конкурен-

цию GPU. Основное достоинство данной архитектуры состоит в том, что для ее использования программисту не требуется изучать новые технологии и применять новые средства разработки. Основные отличия сопроцессора от CPU заключаются в следующем. Во-первых, число одновременно выполняющихся потоков на Intel MIC на порядок больше, чем на процессоре. И во-вторых, время доступа к памяти на сопроцессоре существенно больше, а, значит, нужно обратить особое внимание на эффективное использование кэш-памяти.

В качестве модели программирования для переноса алгоритма моделирования распространения света был выбран нативный (native) режим, когда весь код программы исполняется на сопроцессоре без использования CPU. Альтернативный вариант – offload режим, когда программа запускается на процессоре, а критические с точки зрения времени выполнения участки кода копируются для выполнения на сопроцессор. Выбор нативного режима обусловлен тем, что для использования ускорителя в этом случае не требуется модификации исходного кода.

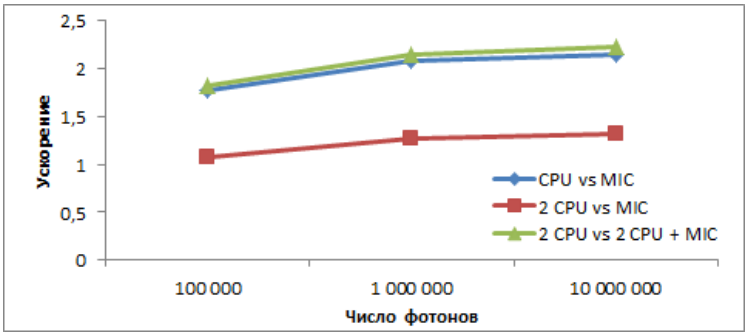


Рис. 4. Ускорение алгоритма моделирования распространения света на Intel MIC (2 процессора Intel Xeon X5680 (6 ядер), 32 ГБ RAM, Intel Xeon Phi SE10X)

Прямой перенос кода на Intel Xeon Phi позволил получить производительность на уровне 6-ядерного процессора Intel Xeon Phi X5680. Дальнейшие оптимизации, направленные на улучшение работы с памятью, позволили ускорить начальную версию программы на Intel MIC в 4 раза. Отметим, что те же оптимизации применялись и к CPU-версии кода, что позволило получить двукратный прирост производительности на процессоре. Полученные результаты приведены на рис. 4.

4. Параллельный алгоритм для графических процессоров

Алгоритм моделирования переноса излучения был перенесен также и на графические процессоры. За основу была взята оптимизированная версия, полученная на предыдущем этапе при работе с Intel Xeon Phi. Следует отметить, что те улучшения, направленные на работу с памятью, что были сделаны ранее в расчете на сопроцессор, оказались полезными и при работе с GPU. При этом использование графического процессора позволило получить производительность программы на уровне Intel Xeon Phi (в 3 раза быстрее процессора Intel Xeon L5630 с 4 ядрами). Результаты переноса приведены на рис. 5.

Заключение

В данной работе приведены результаты сравнения производительности параллельных алгоритмов для моделирования распространения света в гетерогенных средах с произвольной геометрией. Показано, что эффективность распараллеливания описанного здесь алгоритма на системах с общей и распределенной памятью близка к максимальной, масштабируемость на системах с распределенной памятью близка к линейной. Продемонстрированы результаты переноса алгоритма на графические процессоры

и ускорители Intel Xeon Phi. Показано, что производительность обоих типов ускорителей в данной задаче примерно одинакова.

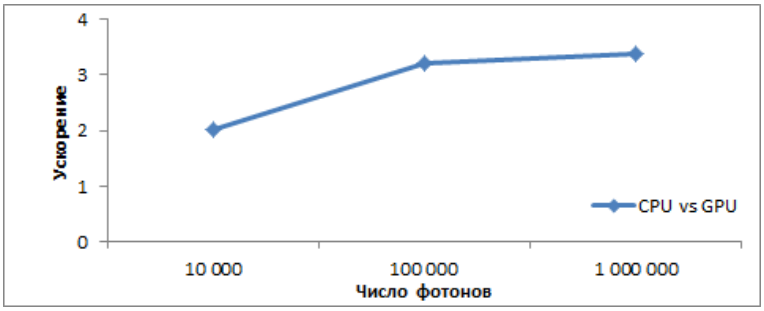


Рис. 5. Ускорение алгоритма моделирования распространения света на GPU (2 процессора Intel Xeon L5630 (4 ядра), 24 ГБ RAM, NVidia Tesla X2070)

Работа выполнена при поддержке Министерства образования и науки Российской Федерации (соглашение 8741) и гранта Президента РФ МК-1652.2012.2 при организационной поддержке Лаборатории информационных технологий ННГУ.

Литература

1. Sorvoja H.S.S., Myllylä T.S., Kirillin M.Yu., Sergeeva E.A., Myllylä R.A., Elseoud A.A., Nikkinen J., Tervonen O., Kiviniemi V. Non-invasive, MRI-compatible fibreoptic device for functional near-IR reflectometry of human brain // Quantum Electorinics. – 2010. – Vol. 40, No. 12. – P. 1067-1073.
2. Chuang C.C., Chen C.M., Hsieh Y.S., Liu T.C., Sun C.W. Brain structure and spatial sensitivity profile assessing by near-infrared spectroscopy modeling based on 3D MRI data // J. Biophotonics. – 2013. – Vol. 6, No. 3. – P. 267-274.
3. Franceschini M.A., Boas D.A. Noninvasive measurement of neuronal activity with near-infrared optical imaging // Neuroimage. – 2004. – Vol. 21, No. 1. – P. 372-386.
4. Wang L., Jacques S.L. Hybrid model of Monte Carlo simulation and diffusion theory for light reflectance by turbid media // J. Opt. Soc. Am. A. Opt. Image. Sci. Vis. – 1993. – Vol. 10, No. 8. – P. 1746-1752.
5. Alerstam E., Andersson-Engels S., Svensson T. White Monte Carlo for time-resolved photon migration // J. Biomed. Opt. – 2008. – Vol. 13, No. 4. – P. 041304.
6. Chen N. Controlled Monte Carlo method for light propagation in tissue of semi-infinite geometry // Appl. Opt. – 2007. – Vol. 46, No. 10. – P. 1597-1603.
7. Majumdar A. Parallel performance study of Monte Carlo photon transport code on shared-, distributed-, and distributed-shared-memory architectures // International Parallel and Distributed Processing Symposium. – 2000. – P. 93.
8. Luu J., Redmond K., Yip Lo W.Ch., Chow P., Lilge L., Rose J. FPGA-based Monte Carlo computation of light absorption for photodynamic cancer therapy // 17th IEEE Symposium of Field-programmable Custom Computing Machines. – 2009. – P. 157-164.
9. Fang Q., Boas D.A. Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units // Opt. Express. – 2009. – Vol. 17, No. 22. –P. 20178–20190.
10. Gorshkov A.V., Kirillin M.Yu. Monte Carlo simulation of brain sensing by optical diffuse spectroscopy // Journal of Computational Science. – 2012. – Vol. 3. – P. 498-503.
11. Горшков А.В., Коршунова А.Л. Оптимальный алгоритм поиска пересечений в задаче Монте-Карло-моделирования распространения зондирующего излучения в головном мозге человека // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2012. – Т. 5, Ч. 2. – С. 73-80.

РАСЧЕТ ОПТИЧЕСКИХ СВОЙСТВ ПОЛУПРОВОДНИКОВЫХ НАНОКРИСТАЛЛОВ В РАМКАХ ТЕОРИИ ФУНКЦИОНАЛА ПЛОТНОСТИ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ НА ГЕТЕРОГЕННЫХ КЛАСТЕРНЫХ СИСТЕМАХ

А.В. Горшков, А.В. Линев, Д.В. Осокин, А.М. Сатанин, А.В. Швецов

Нижегородский госуниверситет им. Н.И. Лобачевского

Кратко описан алгоритм расчета оптических свойств нанокристаллов методом Кона–Шэма. Рассматривается динамика электронных состояний в сферической потенциальной яме с бесконечными стенками. Данная модельная система отражает свойства сферической квантовой точки с достаточно большим потенциальным барьером. Описаны подходы к оптимизации алгоритма для гетерогенных кластерных систем.

Введение

Вычисление 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 [1-4].

Для описания возбуждений в наноструктурах необходимо решить уравнение Шредингера, описывающее многоэлектронную волновую функцию, заданную 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, определяемые только электронной плотностью. В основе теории Кона-Шэма (теории функционала плотности) лежит тот факт, что основные характеристики многоэлектронной 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 [5] и позволяет рассчитывать динамический отклик многоэлектронных систем.

Однако расчеты даже в рамках теории функционала плотности в случае большого числа частиц обладают большой вычислительной трудоемкостью. Одним из выходов является разработка параллельной реализации данного метода, адаптированной для эффективного использования гетерогенных кластерных систем.

1. Расчет спектров нанокристаллов методом Кона-Шэма

Для расчета спектров нанокристаллов необходимо решать нестационарное уравнение Шредингера. Для этого выбирается система невзаимодействующих электронов, называемая KS-системой, определенная таким образом, что точно воспроизводит электронную плотность истинной взаимодействующей системы.

Электронная плотность взаимодействующей системы может быть определена как

$$n(\vec{r},t)=\sum_{j=1}^{N_p}\left|j_j(\vec{r},t)\right|^2, \quad (1)$$

где $\varphi_j(\vec{r},t)$ являются решениями уравнения Шредингера с потенциалом $\upsilon_{KS}(\vec{r},t)$:

$$i\hbar\frac{\partial}{\partial t}\varphi_j(\vec{r},t)=\left[-\frac{\nabla^2}{2m_e^*}+\upsilon_{KS}\left[n;\Phi_0\right](\vec{r},t)\right]\varphi_j(\vec{r},t), \quad (2)$$

$$\upsilon_{KS}\left[n;\Phi_0\right](\vec{r},t)=\upsilon\left[n;\psi_0\right](\vec{r},t)+\int d^3\vec{r}\,'\frac{e^2n(\vec{r}\,',t)}{\left|\vec{r}-\vec{r}\,'\right|}+\upsilon_{XC}\left[n;\psi_0,\Phi_0\right](\vec{r},t). \quad (3)$$

Здесь $\upsilon(\vec{r},t)$ – потенциал внешнего поля, второе слагаемое представляет собой потенциал Хартри $\upsilon_H(\vec{r},t)$, третье слагаемое – обменно-корреляционный потенциал, KS-состояние Φ_0 невзаимодействующих электронов выбирается в виде детерминанта одночастичных орбиталей $\varphi_j(\vec{r},0)$.

На первом шаге происходит расчет электронной плотности согласно формуле (1). Далее решением уравнения Пуассона определяется потенциал Хартри для данной $n(\vec{r},t)$:

$$\Delta\upsilon_H(\vec{r},t)=-4\pi\left(n(\vec{r},t)-n_+(\vec{r},t)\right). \quad (4)$$

Здесь $n_+(\vec{r},t)$ – плотность положительных зарядов. Появление $n_+(\vec{r},t)$ в уравнении Пуассона вытекает из требования электронейтральности нанокристалла. Соответственно интегралы по объему от $n_+(\vec{r},t)$ и $n(\vec{r},t)$ должны быть равны. Плотность $n_+(\vec{r},t)$ выбирается в виде равномерно размазанного по всему объему нанокристалла положительно заряженного желе. Уравнение Пуассона, записанное в конечных разностях, представляет собой систему линейных алгебраических уравнений с *N* неизвестными значениями потенциала Хартри в узлах сетки. Для решения этой системы используется метод сопряженных градиентов. После этого определяется обменно-корреляционный потенциал по аппроксимирующей формуле:

$$\upsilon_{XC}(\vec{r},t)=-\frac{1,222}{r_S(n)}-0,066\ln\left(1+\frac{11,4}{r_S(n)}\right),\ r_S(n)=\sqrt[3]{\frac{3}{4\pi n(\vec{r},t)}}. \quad (5)$$

Здесь использовано приближение локальной плотности. Последнее хорошо зарекомендовало себя при расчетах спектров многоэлектронных систем и заключается в том, что значение обменно-корреляционного потенциала в точке *r* определяется значением электронной плотности *n*(*r*,*t*) в этой же точке *r* в тот же момент времени *t*. Далее решаются нестационарные уравнения Шредингера для KS-системы на одном шаге по времени классическим явным методом Рунге–Кутты 4-го порядка. 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*(*r*,*t*) на следующем шаге по времени.

2. Параллельный алгоритм для гетерогенных кластерных систем

Для декомпозиции области моделирования выбираются два направления и рассматривается прямоугольник в двухмерном пространстве координат *x* и *y*; этот прямоугольник разбивается на блоки как по оси *x*, так и по оси *y*; каждый блок со всеми содержащимися в нем узлами сетки хранится и обрабатывается на соответствующем процессоре.

При распределенной схеме вычислений требуется на разных этапах алгоритмов производить обмены граничными элементами по направлениям x и y между соседними по вычислительной решетке процессорами.

Для более эффективного выполнения операций обмена, был выбран такой порядок хранения данных, при котором границы области рассчета по x представляют собой одну непрерывную область памяти, а границы области по y-направлению лежат непрерывно для каждого слоя по z-направлению. Таким образом, позиция в массиве элемента с координатами (x_i, y_i, z_i) вычисляется по следующей формуле:

$$index_{x_i, y_i, z_i} = z_i + x_i \cdot \dim Z + y_i \cdot \dim X \cdot \dim Z ,$$

где $\dim X$ и $\dim Y$ – число элементов по оси x и z соответственно.

Одна итерация метода сопряженных градиентов включает в себя одно матрично-векторное умножение, три пересчета значений векторов и два скалярных произведения векторов. Все эти шаги могут быть эффективно распараллелены.

Однако для подсчета скалярного произведения векторов необходимо выполнить операцию редукции посчитанных частичных сумм со всех узлов, что является точкой синхронизации, замедляющей параллельные вычисления.

Для увеличения степени параллелизма была использована модификация метода сопряженных градиентов для систем с распределенной памятью: при помощи переупорядочивания этапов метода можно выполнить оба скалярных произведения вместе, тем самым сведя число точек синхронизации до одной [6].

Для достижения дополнительного ускорения все вычисления, производимые на одном узле кластера, были перенесены на графический процессор. Следует отметить, что те улучшения, направленные на работу с памятью, что были сделаны ранее, оказались полезными и при работе с GPU.

Для сравнения была реализована параллельная версия алгоритма для систем с распределенной памятью с использованием технологии OpenMP.

На рис. 1 приведено время работы параллельной CPU- и GPU-версий программы для одного узла кластера.

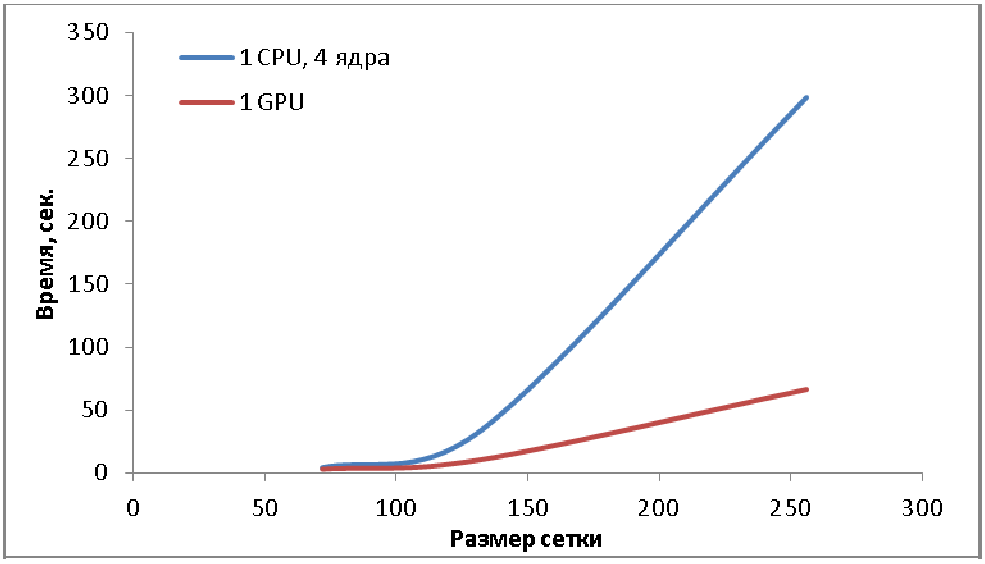


Рис. 1. Время работы параллельной CPU- и GPU-версии в зависимости от размера сетки

Результаты экспериментов при запуске на нескольких узлах кластера приведены на рис. 2.

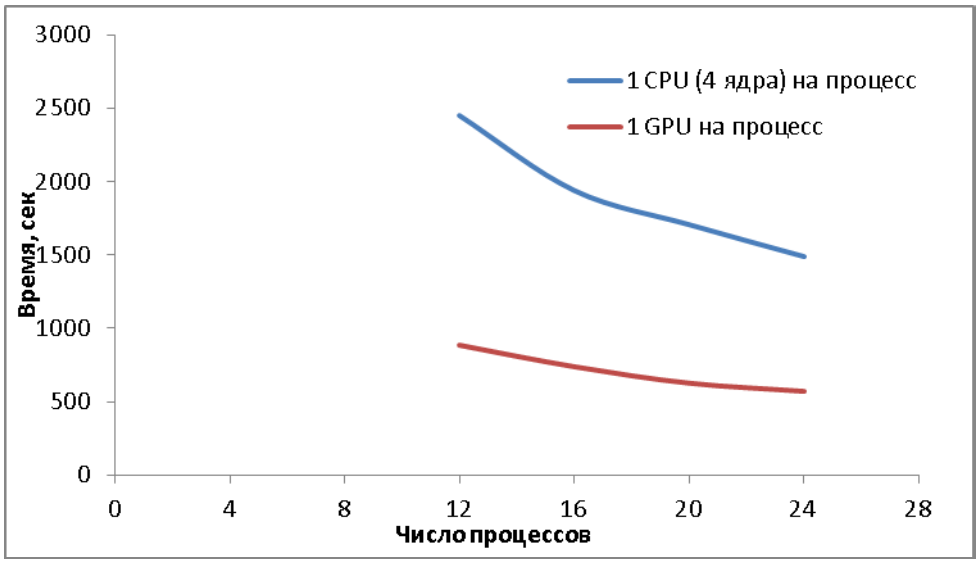


Рис. 2. Время работы параллельной CPU- и GPU-версий для систем с распределенной памятью в зависимости от числа процессов

Как видно из результатов тестов, при увеличении числа процессов ускорение GPU версии сначала снижается, а затем выходит на некоторой постоянный уровень и становится примерно равно 2,5. Это означает, что использование нескольких графических процессоров, установленных на разных узлах гетерогенного кластера, является также оправданным.

Заключение

В данной работе приведены результаты оценки производительности параллельного алгоритма для расчета оптических свойств полупроводниковых нанокристаллов в рамках теории функционала плотности.

Численные эксперименты показали эффективность использования графических ускорителей при решении задач данного типа. Алгоритм расчета на гетерогенном вычислительном кластере продемонстрировал хорошую масштабируемость.

Работа выполнена при организационной поддержке Лаборатории информационных технологий ННГУ.

Литература

1. Shimizu K.T. et al. Surface-Enhanced Emission from Single Semiconductor Nanocrystals // Phys. Rev. Lett. – 2002. – Vol. 89. – P. 117401.
2. Lee J. et al. Bioconjugates of CdTe Nanowires and Au Nanoparticles: Plasmon−Exciton Interactions, Luminescence Enhancement, and Collective Effects // Nano Lett. – 2004. – Vol. 4. – P. 2323–2330.
3. Zhang W. et al. Semiconductor-Metal Nanoparticle Molecules: Hybrid Excitons and the Nonlinear Fano Effect // Phys. Rev. Lett. – 2006. – Vol. 97. – P. 146804.
4. Ratchford D. et al. Manipulating Coupling between a Single Semiconductor Quantum Dot and Single Gold Nanoparticle // Nano Lett. – 2011. – Vol. 11. – P. 1049-1054.
5. Runge E. et al. Density-Functional Theory for Time-Dependent Systems // Phys. Rev. Lett. – 1984. – Vol. 52. – Num. 12. – P. 997-1000.
6. D'Azevedoy E.F., Eijkhoutz V.L., Rominey C.H. Conjugate Gradient Algorithms with Reduced Synchronization Overhead on Distributed Memory Multiprocessors. 1999.

СУПЕРКОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ НЕЛИНЕЙНОГО ВЗАИМОДЕЙСТВИЯ ЛАЗЕРНОГО ИЗЛУЧЕНИЯ С НАНОСТРУКТУРИРОВАННЫМИ СРЕДАМИ МЕТОДОМ FDTD

А.В. Горшков[1,2], А.В. Линев[1], А.В. Пикулин[1,2]

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

Кратко описан метод конечных разностей во временной области для моде-лирования нелинейного взаимодействия лазерного излучения с наноструктурирован-ными средами. Дана оценка эффективности и масштабируемости параллельной вер-сии алгоритма для систем с общей и распределенной памятью. Описаны результаты переноса и подходы к оптимизации алгоритма для графических процессоров. Выполнено сравнение версий алгоритма для различных архитектур с точки зрения времени его работы.

Введение

В последнее время огромный интерес вызывают исследования искусственных сред, обладающих необычными электродинамическими свойствами [1-3]. Наиболее известный пример подобных сред – это вещества с отрицательным показателем преломления или среды Веселаго [4]. Современные нанотехнологии позволяют создавать композиционные материалы, содержащие включения разных масштабов и свойств, которые обладают заданными электродинамическими характеристиками (диэлектрическими и магнитными проницаемостями), коренным образом отличающимися от характеристик компонент [4].

Подобные искусственные среды или метаматериалы в настоящее время широко используются для создания безотражательных покрытий и различных оптических элементов (адаптивных линз, перестраиваемых зеркал, конвертеров и т.д.) [5-7].

Разработка эффективных методов генерации когерентного терагерцового излучения и манипуляции его характеристиками – одна из наиболее «горячих» проблем современной физики. Острая потребность в решении этой проблемы обусловлена перспективами широких практических приложений терагерцового излучения, среди которых важное место занимают спектроскопия и сенсорика биологических конституентов, детектирование взрывчатых и ядовитых веществ по их терагерцовым «отпечаткам пальцев», неразрушающий контроль фармацевтических продуктов, ближнепольная терагерцовая микроскопия микрочипов и других объектов. Несмотря на взрывной характер роста числа работ по метаматериалам, лишь небольшое число групп в мире может реально изготавливать метаматериалы, тем более для такого коротковолнового диапазона, как терагерцовый. Как правило, для этого используется сложный литографический процесс. Очевидно, что ввиду большого числа технологических параметров, такие процессы требуют предварительного выполнения обширных и затратных численных расчетов.

Современные технологии позволяют также создавать искусственные полупроводниковые наноразмерные кристаллы – квантовые точки, составляющие элементную базу квантовой электроники и оптоэлектроники [1-3]. В последнее время активно разрабатываются методы легирования нанокристаллов, методы соединения их с омическими

контактами, развита технология встраивания нанокристаллов в диэлектрические матрицы, метод упаковки нанокристаллов в одно-, двух- и трехмерные структуры. Такие среды не только представляют огромный интерес для бурно развивающейся наноэлектроники, но и имеют более широкую область применимости. Например, кремниевые квантовые точки можно вводить в различные биологические объекты (протеины, мембраны, клетки и т.д.) для локальной диагностики и модификации их свойств.

Несмотря на имеющийся прогресс в проектировании и создании наноструктур для оптоэлектронных приложений существует большой круг проблем, которые еще ждут своего решения. Чтобы создать наноматериалы с заданными электрическими и оптическими свойствами, необходимо продолжить развитие теоретических методов моделирования нанообъектов, разработку методов симуляции реальных технологических процессов, конструирование наноструктур нового типа. Решение подобных задач возможно только при использовании суперкомпьютерных технологий, позволяющих осуществить численные эксперименты с огромными массивами атомов. Исходя из вышесказанного, наиболее важной и актуальной на данный момент является разработка методов, алгоритмов и комплекса программных средств для суперкомпьютерного моделирования, в частности моделирования нелинейных эффектов (многофотонное поглощение, пробой и др.), имеющих место при взаимодействии мощного лазерного излучения с микро- и наноструктурированными средами.

1. Метод конечных разностей во временной области

Одним из методов, широко используемых для моделирования нелинейного взаимодействия лазерного излучения с наноструктурированными средами, является метод конечных разностей во временной области (FDTD, Finite-Difference Time-Domain) [8]. В рамках данного проекта на основе метода FDTD были разработаны, теоретически и экспериментально исследованы алгоритмы суперкомпьютерного моделирования нелинейного взаимодействия лазерного излучения с наноструктурированными средами, учитывающие эффекты ионизации, двухфотонного поглощения и самофокусировки на нановключениях с керровской нелинейностью.

Метод FDTD позволяет проводить численное интегрирование уравнений Максвелла в средах с практически произвольной геометрией, благодаря чему используется для решения широкого класса электродинамических задач.

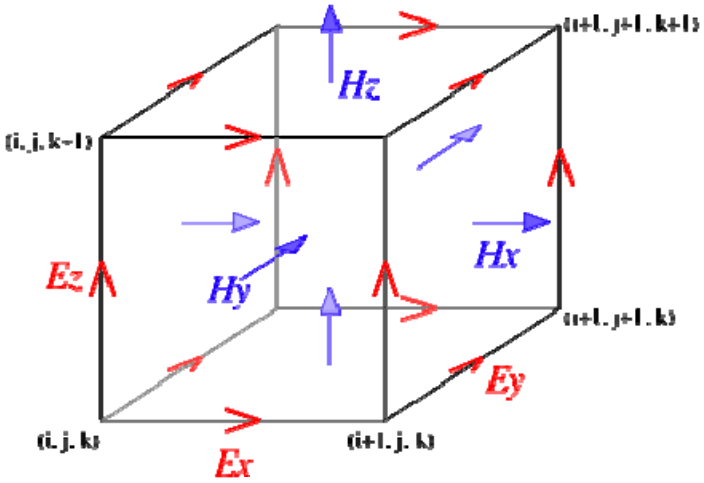


Рис. 1. Поля в ячейке сетки, используемой для расчетов методом конечных разностей во временной области

В уравнениях Максвелла изменение электрического поля E (частная производная) зависит от распределения в пространстве магнитного поля H (ротор). Аналогично, изменение поля H зависит от распределения в пространстве поля E.

На этом наблюдении основан алгоритм Йи (Yee) [9]. Сетки для полей E и H смещены по отношению друг к другу на половину шага дискретизации времени и по каждой из пространственных переменных. Конечно-разностные уравнения позволяют определить поля E и H на данном временном шаге на основании известных значений полей на предыдущем.

При заданных начальных условиях алгоритм Йи дает эволюционное решение во времени от начала отсчета с заданным временным шагом.

Для избегания краевых эффектов в конечной области симуляции (в т.ч. отражений волн от стенок) границы изолируются с помощью согласованных поглощающих слоев (PML, Perfectly Matched Layer) [10]. Использованный здесь PML представляет собой слой неизотропной среды с временной дисперсией, параметры которого подобраны так, чтобы одновременно обеспечить согласованность слоя с внутренним пространством и затухание волн при распространении их в таком слое.

2. Параллельный алгоритм для систем с общей памятью

Основная идея метода FDTD с вычислительной точки зрения состоит в следующем. Задается равномерная трехмерная сетка, узлы которой содержат некоторые начальные значения. На каждом шаге по времени происходит пересчет значения в каждом из узлов сетки на основании значений в соседних узлах, полученных на предыдущем временном шаге. Поскольку для вычисления значений на текущем шаге используются данные только предыдущего шага, каждый узел сетки может быть обработан независимо и параллельно.

На основании данной идеи был разработан параллельный алгоритм моделирования для многоядерных процессоров. Были проведены эксперименты по оценке производительности полученного алгоритма с целью оценки эффективности распараллеливания (рис. 2). Сравнивался последовательный и параллельный алгоритм, работающий на 4 ядрах процессора.

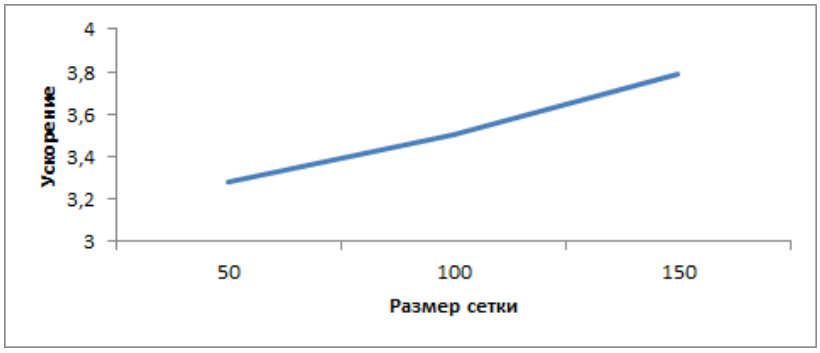


Рис. 2. Ускорение параллельного алгоритма моделирования методом FDTD (Intel Xeon L5630 (4 ядра), 24 ГБ RAM) 100 шагов по времени

Результаты проведенных экспериментов показывают, что эффективность распараллеливания в данной задаче существенно зависит от размеров сетки и при увеличении размера эффективность растет. Такой рост ускорения объясняется тем, что распараллеливание производится в рамках одного шага по времени и при увеличении размеров задачи растет время одного шага, а значит, уменьшается доля накладных расходов на распараллеливание.

Следующий шаг ускорения процесса моделирования – перенос алгоритма на графические процессоры. Рассматриваемая задача имеет достаточно большую степень па-

раллелизма, что позволяет эффективно задействовать все вычислительные ресурсы GPU. Обратить внимание следует только на работу с памятью. В качестве графического процессора для переноса использовались GPU от NVidia с архитектурой Fermi. Данная архитектура поддерживает автоматическое кеширование данных из глобальной памяти, поэтому основной идеей оптимизации в данном случае было размещение данных и порядок обсчета узлов, при которых обеспечивается максимально эффективное использование кеш-памяти GPU. Результаты численных экспериментов приведены на рис. 3.

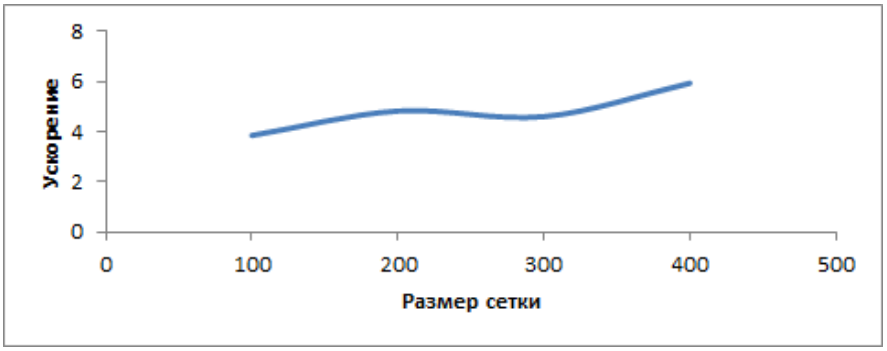


Рис. 3. Ускорение алгоритма моделирования на GPU относительно 4-ядерного процессора (Intel Xeon L5630 (4 ядра), NVidia Tesla M2070, 24 ГБ RAM), 80 шагов по времени

3. Параллельный алгоритм для систем с распределенной памятью

Основная идея распараллеливания при работе на системах с распределенной памятью в данной задаче состоит в следующем. Трехмерная сетка разбивается на области по двум направлениям, каждая область хранится в рамках отдельного процесса. В дополнение к основной области расчета каждый процесс использует граничные узлы, которые необходимы для получения состояния системы в следующий момент времени.

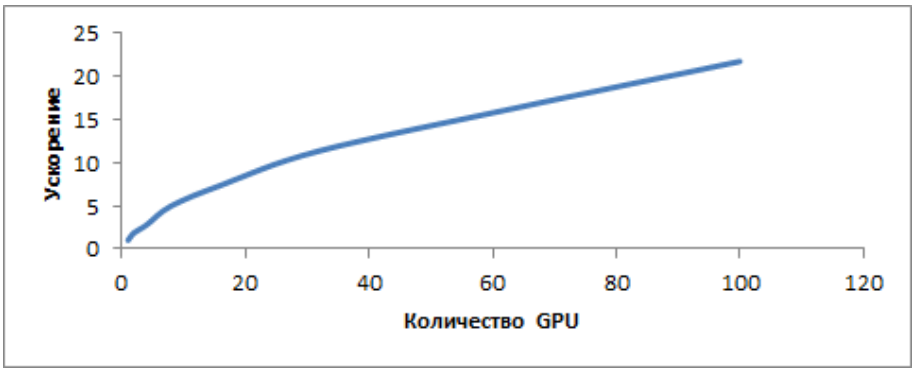


Рис. 4. Ускорение алгоритма моделирования на системах с распределенной памятью (2 процессора Intel Xeon X5570 (4 ядра), 24 ГБ RAM, NVidia X2070), 100 шагов по времени, размер сетки 1000 элементов по каждому направлению

Алгоритм моделирования в данном случае состоит из следующего набора шагов для каждого момента времени:

- каждый процесс вычисляет новые значения узлов сетки в хранимой на нем части области;
- каждый процесс посылает часть значений своих узлов соседнему процессу в качестве граничных;
- каждый процесс принимает дополнительные граничные значения узлов от соседнего процесса; эти дополнительные значения используются только на следующем шаге для вычисления нового состояния системы, после чего удаляются.

Для обеспечения эффективных обменов данными необходимо использовать такую структуру хранения узлов сетки, чтобы для обмена границами по одному из направлений можно было брать данные, лежащие в едином блоке памяти, а по второму – лежащие в виде набора блоков. Это позволяет как существенно сократить количество операций обмена данными между процессами, так и уменьшить объем накладных расходов при копировании данных на GPU.

Результаты численных экспериментов, демонстрирующих эффективность и масштабируемость предложенного решения, приведены на рис. 4.

Заключение

В данной работе приведены результаты оценки производительности параллельных алго-ритмов для моделирования нелинейного взаимодействия лазерного излучения с нано-структурированными средами. Показано, что эффективность рассматриваемого алго-ритма на многоядерных процессорах приближается к максимальной при увеличении размера сетки. Показана применимость графических процессоров и систем с распреде-ленной памятью для ускорения вычисления в данной задаче. Продемонстрирована ли-нейная масштабируемость на гетерогенных кластерах с графическими процессорами на узлах.

Работа выполнена при организационной поддержке Лаборатории информационных технологий ННГУ.

Литература

1. Agranovich V.M. Linear and nonlinear wave propagation in negative refraction metamaterials / V.M. Agranovich, Y.R. Shen, R.H. Baughman, and A.A. Zakhidov // Phys. Rev. B. –2004. –Vol. 69. – P. 165112.
2. Zharov A. A. Nonlinear Properties of Left-Handed Metamaterials / A.A. Zharov, I.V. Shadrivov, and Yu. S. Kivshar // Phys. Rev. Lett. – 2003. – Vol. 91.– P. 037401.
3. Shvets G. Photonic approach to making a material with a negative index of refraction / G. Shvets // Phys. Rev. B. – 2003- Vol. 67. – P. 035109-1 – 035109-8.
4. Veselago V. G. Electrodynamics of materials with negative index of refraction / V. G. Veselago // Phys. Usp..- 2003. - Vol. 46. – No. 7. – P. 764.
5. Zhang S. Demonstration of metal-dielectric negative-index metamaterials with improved performance at optical frequencies / S. Zhang, W. Fan, K. J. Malloy, S. R. J. Brueck, N. C. Panoiu and R. M. Osgood // J. Opt. Soc. Am. B. – 2006. – Vol. 23. – P. 434.
6. Ziolkowski R. W. Ultrathin, metamaterial-based laser cavities / R.W. Ziolkowski // J. Opt. Soc. Am. B. – 2006. –Vol. 23. – P.451.
7. Korobkin D. Enhanced near-field resolution in midinfrared using metamaterials / D. Korobkin, Ya. Urzhumov, and G. Shvets // J. Opt. Soc. Am. B. – 2006. – Vol. 23. – P.468.
8. Taflove A. Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. C. Hagness. – 2nd ed. – Boston, London: Artech House, 2000.
9. Yee K. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media // IEEE Transactions on Antennas and Propagation. -1966. –Vol. 14, No. 3. – P. 302–307.
10. Berenger J. A perfectly matched layer for the absorption of electromagnetic waves // Journal of Computational Physics. -1994. –Vol. 114, No. 2. – P. 185–200.

ИСПОЛЬЗОВАНИЕ МОДУЛЯРНОЙ АРИФМЕТИКИ ДЛЯ СИНТЕЗА СИСТЕМ ОБРАБОТКИ МНОГОРАЗРЯДНЫХ ЧИСЕЛ НА FPGA

М.А. Дерябин, А.А. Зайцев

Северо-Кавказский федеральный университет, Ставрополь

Рассмотрены возможности приложения модулярной арифметики при синтезе интегральных схем на ПЛИС архитектуры FPGA для работы с числами большой разрядности. Проведен анализ архитектурных особенностей FPGA, и выработаны требования к алгоритмам, реализующим операции с целыми многоразрядными числами. Приведен пример синтеза устройства на основе системы остаточных классов.

Введение

Программируемые пользователем вентильные матрицы (FPGA) являются перепрограммируемыми кремниевыми чипами. Использование FPGA-чипов во всех отраслях промышленности обусловлено тем, что данные ПЛИС сочетают в себе лучшие части специализированных интегральных схем (ASIC) и систем, основанных на базе традиционных процессоров [1]. За счет архитектурных особенностей и внутреннего параллелизма FPGA способны обеспечивать аппаратно-тактируемую скорость и надежность, не требуя при этом таких же объемов, как стандартный микропроцессор, и дополнительных накладных расходов на индивидуальный дизайн ASIC.

Однако при реализации операций с длинными числами, превышающими разрядную сетку конкретной схемы, возникают определенные сложности, связанные с ограниченностью возможностей функциональных блоков FPGA. Данная особенность элементарных вычислительных узлов ПЛИС дает возможность выполнять простые операции с высокой скоростью в реальных ситуациях, однако понижает возможности при реализации сложных алгоритмов. При этом отдельный отпечаток накладывает абсолютный параллелизм FPGA. В соответствии с этим реализация многоразрядной арифметики на ПЛИС данной архитектуры требует индивидуального подхода и анализа. Один из эффективных методов решения данной проблемы открывает модулярная арифметика и ее приложения.

1. Использование модулярной арифметики при обработке многоразрядных чисел

Система остаточных классов (Residue number system, RNS, СОК) является непозиционной системой представления чисел [2]. Числа в ней определяются кортежем разрядов, являющихся остатками от деления данного числа на модули системы – набор взаимно простых чисел, определяющих СОК. Согласно китайской теореме об остатках (КТО), такое представление является единственным, если исходное число не превышает заранее известного диапазона СОК – произведения оснований. При этом значение любого разряда не влияет на значения остальных, что дает данной системе определенные преимущества перед более традиционными позиционными системами счисления.

Учитывая отсутствие межразрядных связей, модулярная арифметика позволяет проводить декомпозицию системы большого динамического диапазона на ряд параллельных независимых каналов меньшей разрядности. Использование такого подхода

увеличивает эффективность вычислений, в частности при реализации операций сложения, умножения, возведения в степень.

Особенностью адаптации алгоритмов для СОК является необходимость перевода данных из позиционной системы в СОК и обратно. 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.

Обратное преобразование числа из модулярного представления в двоичную форму базируется на классической теореме из теории чисел, которая называется китайской теоремой об остатках. Предлагается метод восстановления чисел на основе совместного использования КТО и обобщенной позиционной системы счисления (ОПСС) [3], в которой числа заменяются кортежем цифр небольшой разрядности.

Использование данных методов перевода позволяет обеспечить возможность обработки данных без использования на вычислительном устройстве операций с многоразрядными числами. Благодаря чему в алгоритмах достигается параллелизм и высокая скорость работы основных арифметических операций, что продемонстрировано в работе [4]. Согласно данным исследованиям, ускорение, получаемое при использовании СОК для решения задачи модульного возведения в степень на персональном компьютере, относительно последовательного аналога данного алгоритма, превосходит количество параллельных потоков. При соблюдении определенных условий СОК позволяет достичь высокой скорости работы алгоритма при использовании даже небольшого числа параллельных потоков выполнения. Данный эффект достигается за счет снижения разрядности обрабатываемых операндов и перехода от многоразрядной арифметики к стандартной, поддерживаемой вычислительной системой на архитектурном уровне.

2. Синтез схем многоразрядной арифметики на FPGA

Основываясь на данных исследованиях, можно перенести имеющиеся алгоритмы на FPGA. Однако при этом на алгоритмы накладываются определенные ограничения, связанные с архитектурными особенностями вычислительной системы. Во-первых, FPGA реализует реальный параллелизм (true parallelism), что следует учитывать при реализации алгоритмов. При этом система остаточных классов за счет параллелизма на уровне операций позволяет максимально эффективно использовать ресурсы ПЛИС. Во-вторых, накладывается существенное ограничение на размерность элементарных операндов математических преобразований. Чаще всего разрядность FPGA ограничена 16, 32 или 64 битами на число. В связи с этим необходимо особым образом подходить к выбору оснований СОК. Важно учитывать, что элементарные операции не должны приводить к переполнению простого типа, иначе неизбежны потери данных. В-третьих, использование совместно с ПЛИС стандартного микропроцессора реального времени позволяет вынести ряд незначительных операций и тем самым снизить загруженность конечной схемы. В-четвертых, для реализации действий с константными данными особо эффективным методом при использовании ПЛИС являются LUT (look-up tables), предоставляющие способ для быстрого однотактового доступа к данным, хранящимся в табличном виде.

На основе описанного подхода к представлению чисел была синтезирована система, реализующая базовые арифметические операции. Данная система базируется на FPGA фирмы XILINX версии Spartan 6, являющейся 32-битной вычислительной системой. В связи с чем выбор модулей был обусловлен ограничениями системы – были подобраны девять 16-битных чисел. Общая разрядность обрабатываемых чисел, таким образом, не превышает 144 бит.

На рис. 1 представлена функциональная схема блока перевода из позиционного представления числа в СОК. Перевод осуществляется путем дробления на 16-битные форматы, каждый из которых масштабируется и накапливается в общую сумму. Преобразование производится параллельно одновременно по всем модулям.

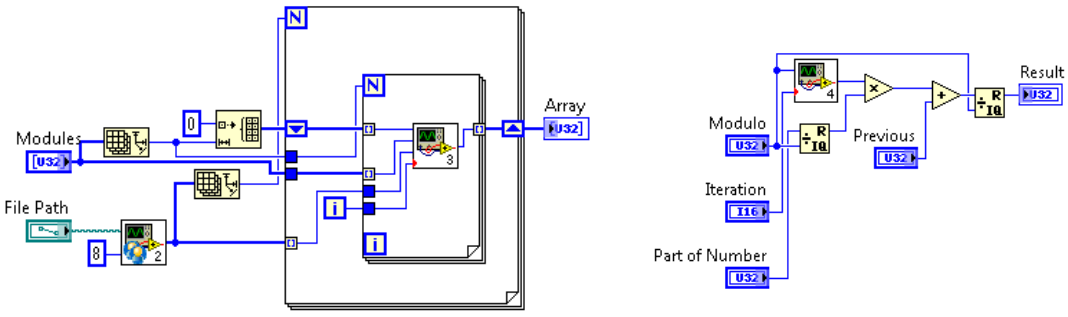


Рис. 1. Чтение и перевод шестнадцатеричного числа в СОК
в среде программирования LabVIEW

На рис. 2 представлен блок, в рамках которого выполняется умножение вектора числа в СОК на матрицу базисов, производится учет межразрядных переносов, в результате на выходе получается представление числа в ОПСС.

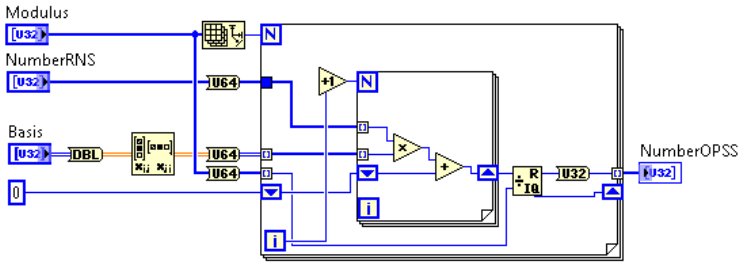


Рис. 2. Функциональная схема перевода числа из СОК в ОПСС

На рис. 3 представлен блок модульного возведения в степень, реализующий алгоритм быстрого модульного возведения, представленный в работе [5].

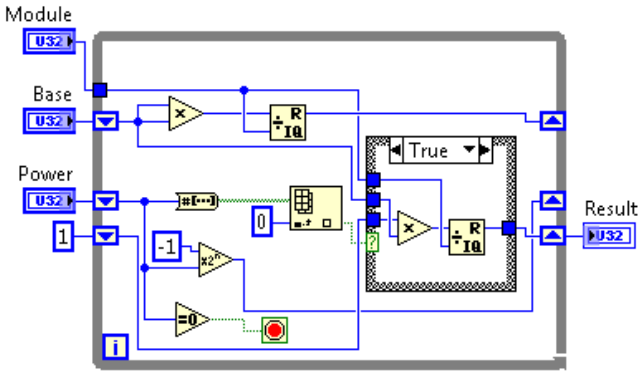


Рис. 3. Модульное возведение в степень

Разработанный модуль способен производить операции над многоразрядными числами в реальном времени, используя основные преимущества FPGA. Основной результат – снижение загруженности самой схемы при реализации арифметических операций над длинными числами.

Вывод

Работа с многоразрядными числами в FPGA затруднена аппаратными и архитектурными ограничениями самой платформы. В этом случае наиболее эффективное использование всех возможностей ПЛИС данного вида достигается с использованием параллельных методов обработки данных на уровне арифметических операций. Показано, что использование СОК является параллельной альтернативой классическим методам работы с большими числами. Система остаточных классов делает возможным реализация многоразрядной арифметики на ПЛИС с жестко ограниченной разрядной сеткой.

Литература

1. Баран Е.Д. LabVIEW FPGA. Реконфигурируемые измерительные и управляющие системы. – М.: ДМК Пресс, 2009. – 448 с.
2. Omondi A., Premkumar B. Residue number systems. Theory and Implementation. – London: Imperial College Press, 2007. – 258 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.
4. Дерябин М.А., Зайцев А.А. Использование модулярной арифметики для ускоре-ния выполнения операций с числами большой разрядности // Материалы XII Всероссийской конференции «Высокопроизводительные вычисления на кластерных системах». Нижний Новгород: Издательство Нижегородского университета, 2012. С. 129-133.
5. Schneier Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. – New York: John Wiley & Sons, 1995. – 662 p.

МИНИМИЗАЦИЯ КОНВЕКТИВНОГО И РАДИАЦИОННОГО ТЕПЛОВОГО ПОТОКА ПРИ ВХОДЕ АППАРАТА В АТМОСФЕРУ

В.В. Дикусар, Н.Н. Оленёв

Вычислительный центр им. А.А. Дородницына РАН, Москва

Принцип максимума в задаче оптимального управления по минимизации конвективного и радиационного теплового потока при входе аппарата в атмосферу сводит поиск решения к краевой задаче для системы обыкновенных дифференциальных уравнений. Рассмотрена схема параллельного расчета численных значений матрицы Якоби на каждом шаге расчета траектории. Приведены некоторые результаты расчетов.

Введение

Задача оптимального управления состоит в минимизации конвективного и радиационного теплового потока при входе космического летательного аппарата (КЛА) в атмосферу. При этом нужно учесть ограничения на динамический режим полета. Если угол атаки КЛА постоянный, а распределение коэффициента давления c_p на поверхности КЛА не изменяется или почти не изменяется в процессе полета, то ограничения $X < X_{доп}$ и $Y < Y_{доп}$, налагаемые на аэродинамические силы по условиям прочности конструкции КЛА, сводятся к ограничению на скоростной напор:

$$\frac{pV^2}{2} = q \le q_{доп} = \min\left(\frac{X_{доп}}{c_x S}, \frac{Y_{\text{доп}}}{c_y S}\right).$$

Здесь c_x и c_y практически постоянны, а масса КЛА играет второстепенную роль. Статистические аэродинамические нагрузки уравновешиваются распределенными силами инерции, пропорциональными массовой плотности КЛА, поэтому при пропорциональном изменении этой плотности условия нагрузки на конструкцию КЛА не изменяются, роль может сыграть лишь перераспределение массовой плотности.

В случае когда угол атаки КЛА регулируется, а условия обтекания претерпевают существенные изменения в процессе полета, прочностные ограничения могут иметь достаточно сложный характер и определяться неравенствами типа

$$F(q, \alpha, M) \le 0.$$

Если на борту КЛА находится экипаж, важно обеспечить выполнение ограничений на значения перегрузки для членов экипажа. Полная перегрузка n определяется отношением аэродинамической силы к весу КЛА на Земле:

$$n = \frac{R}{mg_3},$$

где R – аэродинамическая сила.

Важно учесть направление перегрузки. Если угол атаки КЛА почти не меняется, а условия обтекания неизменны, то направление действия перегрузки определено, и условие $n \le n_{доп}$ дает

$$\frac{q}{m} \le \left(\frac{q}{m}\right)_{доп}.$$

Принцип максимума сводит задачу оптимального управления к краевой задаче для системы обыкновенных дифференциальных уравнений [1]. Применение метода Ньюто-

на при решении краевой задачи включает трудоемкую процедуру численного расчета матрицы Якоби. Расчет естественно распараллеливается в силу независимости частных производных. Полученный параллельный алгоритм применяется для решения предложенной задачи оптимального управления. В численной реализации параллельного алгоритма расчета матрицы Якоби на языке C++ используется MPI [2, 3].

1. Описание динамического и теплового режимов при входе в атмосферу

Рассмотрим задачи выбора угла атаки аппарата, тормозящегося в атмосфере, при минимизации суммарного теплового потока с учетом ограничений на величину полной перегрузки скоростного напора. Решение задач определяет маневренность аппарата [4].

Суммарное количество тепла

$$Q = \int_0^T CV^3 \rho^{1/2} dt\,. \tag{1}$$

Требуется выбрать управление $C_y(t)$, доставляющее минимум $q(T)L(t)$ (1) при следующих ограничениях:

$$n_\sum = \sqrt{C_x^2 + C_y^2}\, q \frac{S}{G} \le N, \qquad q = \frac{\rho V^2}{2}, \qquad G = mg, \tag{2}$$

$$C_y^{\min} \le C_y \le C_y^{\max}, \qquad C_x = C_{xo} + kC_y^2, \tag{3}$$

$$\rho = \rho_0 e^{-\beta H}, \qquad g = g_0 \frac{R^2}{(R+H)^2}, \qquad V^\bullet = -C_x q \frac{S}{m} - g\sin\theta\,, \tag{4}$$

$$\theta^\bullet = C_y q \frac{S}{mV} + \left(\frac{V}{R+H} - \frac{g}{V}\right)\cos\theta, \qquad H^\bullet = V\sin\theta\,,\ L^\bullet = \frac{RV\cos\theta}{R+H} \tag{5}$$

где $n_\sum$ – полная перегрузка, q – скоростной напор, ρ – плотность атмосферы, V – скорость аппарата, θ – угол наклона траектории, H – высота полета, G – вес аппарата, m – масса, g_0 – ускорение силы тяжести на поверхности планеты, R – радиус планеты, C_x – коэффициент лобового сопротивления, C_y – коэффициент подъемной силы, S – характерная площадь аппарата, C_{xo}, k, ρ_0, β, C, $C_y^{\min}$, $C_y^{\max}$, N – постоянные величины.

Для системы (1)–(5) заданы начальные условия

$$V(0) = V_0, \quad \theta(0) = \theta_0, \quad H(0) = H_0, \quad L(0) = L_0, \quad Q(0) = 0\,. \tag{6}$$

Граничные условия имеют вид

$$L(T) = a,\ V(T) = V_1, \quad \theta(T) = \theta_1, \quad H(T) = H_1,\ T \text{ – не фиксировано}, \tag{7}$$

где a – параметр.

Заметим, что ограничение (1) выполняется автоматически из условий (2), (3).

Пусть спускаемый аппарат приходит из начального состояния (6) в конечное положение оптимальным образом в смысле минимума или максимума дальности в предположении, что на оптимальной траектории выполнено условие регулярности [1, 5]

$$\frac{\partial n_\Sigma}{\partial C_y} \ne 0, \quad n_\Sigma = N. \tag{8}$$

В этом случае принцип максимума имеет следующий вид:

$$\Pi = P_\theta \theta^\bullet + P_H H^\bullet + P_V V^\bullet + P_L L^\bullet, \qquad \Pi_1 = \Pi - \lambda(t)\left(n_\Sigma - N\right), \tag{9}$$

$$P_\theta^\bullet = -\frac{\partial \Pi}{\partial \theta}, \quad P_V^\bullet = -\frac{\partial \Pi}{\partial V}, \quad P_H^\bullet = -\frac{\partial \Pi}{\partial H}, \quad P_L^\bullet = -\frac{\partial \Pi}{\partial L}, \quad P_Q^\bullet = -\frac{\partial \Pi}{\partial q}. \tag{10}$$

Здесь $\lambda(t)$ – множитель Лагранжа, который определяется из условия Блисса [1, 5].

Π – функция Понтрягина, Π_1 – функция Лагранжа,

$$\frac{\partial \Pi}{\partial C_y} - \lambda(t)\frac{\partial n_\Sigma}{\partial C_y} = 0, \tag{11}$$

где $P_\theta, P_V, P_H, P_L, P_Q$ – соответствующие сопряженные переменные. Для ограничения типа неравенств выполнено условие дополняющей нежесткости

$$\lambda(t)\left(n_\Sigma - N\right) = 0. \tag{12}$$

Так как наша система автономна и на время спуска никаких ограничений не накладывается, то функция Понтрягина тождественно равна нулю, т.е.

$$\Pi\left(P, x, u\right) \equiv 0, \quad u = C_y, \quad x = \left(\theta, V, H_y, L\right), \quad P = \left(P_\theta, P_V, P_H, P_L, P_Q\right). \tag{13}$$

Сопряженная переменная $P_Q(T)$ нормируется условием

$$P_Q(T) = -1. \tag{14}$$

Из $P_Q^\bullet = 0$ следует $P_Q(t) \equiv -1$ на всей оптимальной траектории.

Начальные условия для системы (10) неизвестны и являются параметрами задачи. Условия $P_Q(t) \equiv -1$ и $\Pi(P, x, u) \equiv 0$ (13) по существу определяют три свободных параметра

$$P_\theta(0) = C_1, \quad P_V(0) = C_2, \quad P_L(0) = C_3, \tag{15}$$

так как $P_H(0)$ определяется из условия $\Pi(P, x, u) \equiv 0$.

В этом случае число контролируемых в конце траектории функций совпадает с числом свободных параметров задачи (1)–(7), (9), (10), поскольку время T не фиксировано и является свободным параметром.

Согласно принципу максимума программа управления выбирается из условия

$$\Pi \to \max_{C_y} \quad \text{при} \quad Q(T) \to \min. \tag{16}$$

Часть функции Понтрягина (9), которая явно зависит от управления $C_y(t)$

$$\Pi_0 = P_\theta \frac{C_y \rho V S}{2m} - P_V \frac{C_x \rho V^2 S}{2m}. \tag{17}$$

Управление $C_y(t)$ может принимать не только концевые значения (3), но также и промежуточное, которое определяется из условия

$$\frac{\partial \Pi_0}{\partial C_y} = 0, \qquad C_y^* = \frac{P_\theta}{2kP_V V} \quad , \qquad C_y^{\min} \le C_y^* \le C_y^{\max}. \tag{18}$$

Вычислим теперь три значения Π_0:

$$\Pi_1 = \Pi_0\left(C_y^{\min}\right), \quad \Pi_2 = \Pi_0\left(C_y^{\max}\right), \quad \Pi_3 = \Pi_0\left(C_y^*\right)$$

и определим соответствующие минимальные и максимальные величины Π_0

$$\Pi_0^{\min} = \min\left\{\Pi_1, \Pi_2, \Pi_3\right\}, \ \Pi_0^{\max} = \max\left\{\Pi_1, \ \Pi_2, \ \Pi_3\right\}. \tag{19}$$

Соотношения (19) определяют характер оптимального управления для задачи Понтрягина, т.е. при условии $n_{\sum} < N$. Решение поставленной задачи значительно упрощается, если правый конец траектории контролируется условием

$$H(T) = H_1. \tag{20}$$

В этом случае решение краевой задачи определяется граничными условиями

$$\theta(T) = \theta_1, \qquad V(T) = V_1,\ L(T) = a \tag{21}$$

и зависит от трёх произвольных постоянных C_1, C_2 и C_3.

Таким образом, исходная задача сводится к трёхпараметрической краевой задаче (1), (6), (10), (15), (21), а оптимальное управление $C_y(t)$ определяется в каждой точке t согласно принципу максимума (19).

Учет ограничений на перегрузку (2) существенно увеличивает трудности получения решения даже в регулярном случае. Первая проблема связана с вычислением множителя Лагранжа $\lambda(t)$ (11). При итеративном поиске оптимальной траектории наблюдается значительный рост множителей Лагранжа при $C_y(t) \to 0$. Указанную трудность можно преодолеть в рамках теории сингулярно-возмущенных систем [1, 5].

В поставленной задаче трудность определения геометрии оптимальной траектории связана с определением момента схода с ограничения $n_{\Sigma} = N$ (2).

Заметим, что суммарная перегрузка n_{Σ} (2) имеет два компоненты n_x и n_y. Первая называется продольной перегрузкой, а вторая – нормальной:

$$n_y = \frac{\rho V^2 S}{2mg_0} C_y, \qquad n_x = \frac{\rho V^2 S}{2mg_0} C_x, \qquad n_{\sum} = \sqrt{n_x^2 + n_y^2}. \tag{22}$$

Новое ограничение

$$|n_y| + n_x \le N_1, \qquad |n_y| + n_x - N_1 = \varphi(x,u) \le 0. \tag{23}$$

При соответствующем выборе N_1 из справедливости неравенства (23) заведомо будет выполнено стандартное ограничение. Указанный факт следует из неравенства

$$N_1 \ge \left[|n_y| + |n_x| \right] \ge \sqrt{n_x^2 + n_y^2}, \tag{24}$$

причем равенство достигается при $C_y = 0$.

Вычислим теперь производную $\varphi(x,u)$ по C_y

$$\frac{\partial \varphi}{\partial C_y} = \frac{\rho V^2 S}{2mg_0} \left[\operatorname{sign} C_y + 2kC_y \right]. \tag{25}$$

В этом случае множитель Лагранжа $\lambda(t)$ для ограничения $\varphi(x,u) \le 0$ (23)

$$\lambda(t) = \frac{2\left(\frac{P_\theta}{2} - kP_V C_y V \right) g_0}{V \left[\operatorname{sign} C_y + 2kC_y \right]}. \tag{26}$$

Краевая задача решалась методом продолжения решений по параметру t [6].

2. Параллельные вычисления

При реализации метода Ньютона решения краевой задачи, к которой сводится представленная выше задача оптимального управления, на каждом шаге расчета требуется трудоемкая процедура расчета матрицы Якоби. Параллельные вычисления в этом расчете реализованы с использованием технологии MPI на языке C++, с использованием функций, описанных в [2, 3]. В каждый заданный момент времени при движении

КЛА расчет каждой частной производной матрицы Якоби происходил параллельно. Применение параллельных вычислений ускорило расчет траекторий, удовлетворяющих принципу оптимальности, на порядок.

При входе аппарата в атмосферу, как правило, учитывают три основных ограничения: прочностное, ограничение на полную перегрузку и ограничение на нагрев. Указанное ограничение можно свести к задаче полета аппарата на минимум и максимум дальности с учетом ограничений на величину полной перегрузки, а в качестве функционала рассматривается минимум нагрева. При этом дальность в конце полета зависит от параметра. Выбор параметра производится из условия минимума максимального нагрева аппарата. Так как на величину угла атаки наложено ограничение, то отсюда следует оценка на прочностное ограничение.

Краевая задача решалась с использованием параллельного варианта метода Ньютона. Использовалась естественная параллелизация расчета элементов матрицы Якоби для каждого шага траектории КЛА на кластерном суперкомпьютере МСЦ РАН.

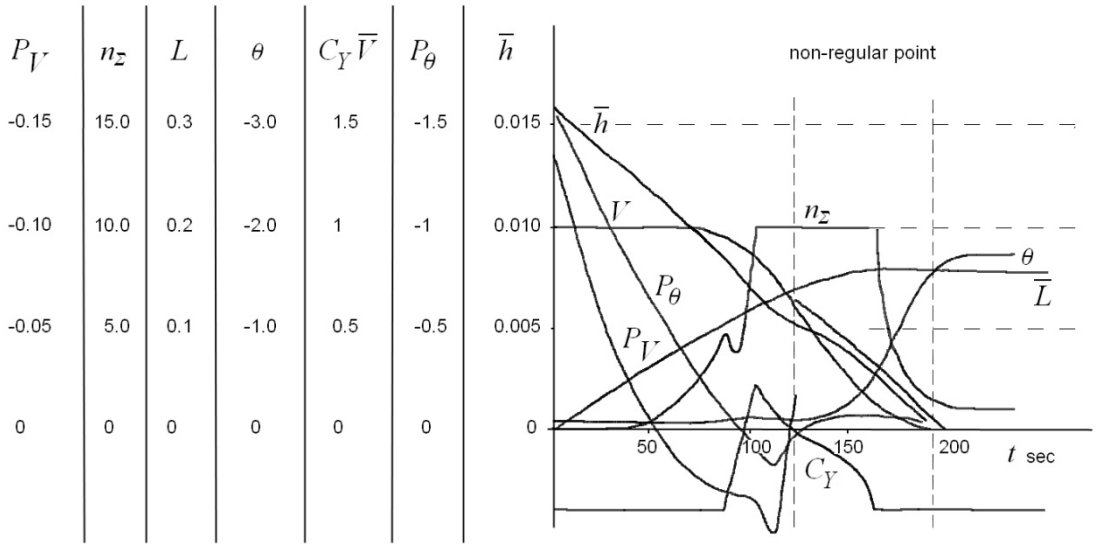


Рис. 1. Динамика траектории спуска при наличии нерегулярной точки

Результаты параллельных расчетов на высокопроизводительной вычислительной технике оптимальной траектории спуска аппарата в атмосферу с учетом ограничений на величину полной перегрузки в случае нерегулярного принципа максимума приведены на рис.1. При этом применялась общепринятая в аэродинамике полета нормировка переменных.

Получено четырехкратное ускорение счета матрицы Якоби при расчете на 16 узлах.

Работа выполнена при поддержке РФФИ (проекты №№ 11-07-00201, 12-01-00916, 13-07-01020), ПФИ Президиума РАН № 15, ПФИ ОМН РАН №3.

Литература

1. Афанасьев А.П., Дикусар В.В., Милютин А.А., Чуканов С.В. Необходимое условие в принципе максимума. – М.: Наука, 1990. – 320 с.
2. Оленев Н.Н. Основы параллельного программирования в системе MPI. М.: ВЦ РАН, 2005. – 80 с.

3. Долматова А.И., Оленев Н.Н. Параллельные вычисления в моделировании региональной экономики: учебное пособие. – Киров: ВятГУ, 2012. – 125 с.
4. Ярошевский В.А. Вход в атмосферу космических летательных аппаратов. – М.: Наука, 1988.
5. Дикусар В.В., Милютин А.А. Количественные и качественные методы в принципе максимума. – М.: Наука, 1989.
6. Дикусар В.В, Кошька М., Фигура А. Продолжение решений в прикладных задач оптимального управления. МФТИ, 2001.

АВТОМАТИЗИРОВАННАЯ НАСТРОЙКА ПАРАМЕТРОВ БИБЛИОТЕКИ S-MPI ДЛЯ КЛАСТЕРНЫХ СИСТЕМ С РАЗЛИЧНОЙ АРХИТЕКТУРОЙ

Д.В. Донцов

ООО «Центр компетенций и обучения», Саров
e-mail: dvdontsov@compcenter.org

Представлен инструмент, предназначенный для настройки параметров библиотеки S-MPI, реализующей стандарт MPI-2. Данный инструмент позволяет автоматизировать подбор оптимальных значений параметров библиотеки S-MPI с учетом как специфики архитектуры кластерной системы, так и специфики параллельного приложения. Доклад посвящен рассмотрению подходов, применяемых для подбора оптимальных значений параметров, включая возможность анализа зависимости производительности параллельного приложения от значений настраиваемых параметров без анализа его исходного кода.

Введение

Библиотека S-MPI [1], являющаяся отечественной реализацией стандарта MPI-2 [2], предназначена для запуска параллельных приложений на кластерах с различной архитектурой. Она имеет большое количество параметров, которое контролирует ее поведение и влияет на ее производительность. Оптимальные значения этих параметров могут сильно варьироваться в зависимости от среды исполнения и от архитектуры параллельного приложения. Еще одной важной особенностью является наличие фактора взаимного влияния параметров библиотеки друг на друга. Поэтому процедура поиска оптимальных значений параметров библиотеки S-MPI является достаточно сложной, долгой и ресурсоемкой.

Представляемый инструмент предназначен для автоматизации поиска оптимальных значений параметров библиотеки S-MPI с учетом как специфики кластера, так и специфики приложения.

Инструменты автоматической настройки

В настоящее время уже существуют инструменты автоматической настройки MPI-реализаций:

- MPI Tuner, являющийся коммерческим продуктом, входящим в поставку библиотеки Intel MPI [3];
- OTPO (Open Tool for Parameter Optimization [4]), являющийся свободно распространяемым продуктом и предназначенный для настройки OpenMPI [5].

Обе реализации инструментов настройки обладают рядом недостатков, которые не позволяют их использовать для настройки библиотеки S-MPI. Если MPI Tuner предназначен только для настройки Intel MPI, то OTPO достаточно ограничен в его функциональности: отсутствие возможности настройки с учетом специфики кластера, отсутствие возможности настройки параметров с композиционными значениями, изначальная поддержка ограниченного набора тестов. В связи с этим для библиотеки S-MPI был разработан собственный инструмент, получивший название MPIBoost, позволяющий проводить настройку как с учетом специфики кластера, так и с учетом специфики па-

раллельного приложения. Каждый из режимов работы разработанного инструмента настройки обладает рядом своих преимуществ и не только не исключает их совместное использование, а позволяет добиться максимальной эффективности библиотеки S-MPI в ходе их совместного использования.

Режим настройки на специфику кластера

Режим настройки на специфику кластера предназначен для поиска оптимальных значений параметров S-MPI с учетом среды запуска (совокупности программных и аппаратных средств, включая коммуникационное оборудование). Этот режим работы базируется на использовании IMB (Intel® MPI Benchmarks [6]), который позволяет получать необходимые данные о выполнении MPI-операций для различных размеров сообщений. Получаемые с его помощью данные позволяют провести оценку производительности MPI-операций, независимо от архитектуры параллельного приложения, и на ее основе определить оптимальные значения параметров библиотеки S-MPI.

Параметры S-MPI можно разделить на три группы:

- могут принимать значения из предопределенного списка допустимых значений;
- могут принимать значения из заданного диапазона;
- могут принимать композиционные значения.

На рис. 1 показан результат подбора параметра библиотеки S-MPI, отвечающего за правило выбора алгоритма коллективной операции Allreduce и принимающего композиционное значение.

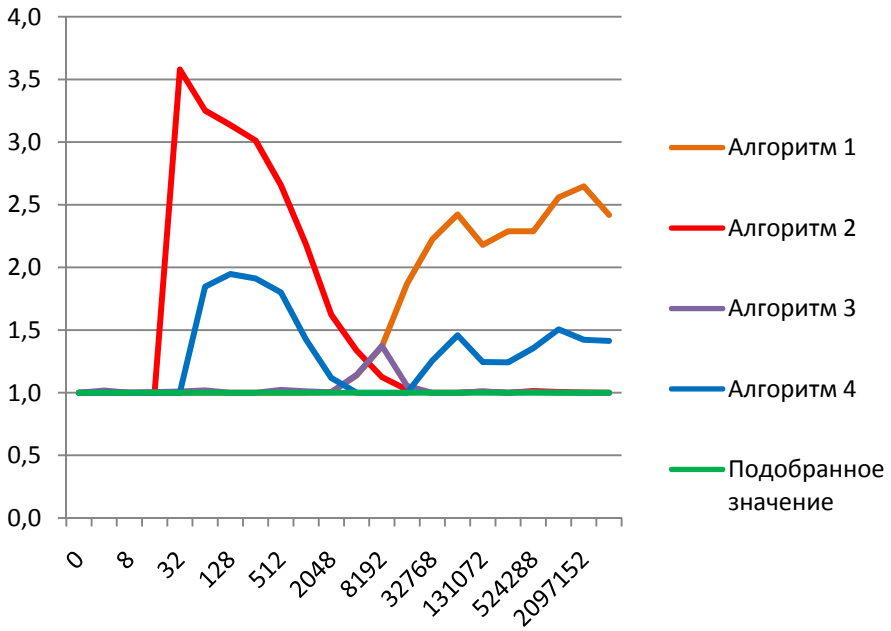


Рис. 1. Оптимизация MPI-операции Allreduce. N1:P8

Режим настройки на специфику приложения

Основной особенностью режима настройки на специфику приложения является задача определения необходимого набора параметров, подлежащих настройке, в условиях отсутствия сведений о том, как оно реализовано. MPIBoost не требует от пользователя каких-либо сведений об архитектуре приложения (набор используемых MPI-операций, размеры передаваемых сообщений и др.). В качестве информационной базы инструмент использует статистическую информацию, предоставляемую библиотекой S-MPI. Такая информация позволяет не только определить эффективность выполнения MPI-операций, но и исключить из процесса настройки параметры, которые не могут

повлиять на производительность. Настройка каждого параметра S-MPI производится с учетом возможности влияния его значения на другие параметры библиотеки.

Запуск MPIBoost с настройкой на тест IS (класс B) из набора NPB (NAS Parallel Benchmarks [7]) позволил получить настройки, улучшающие его производительность на 3–5%.

Процедура автоматической настройки параметров библиотеки S-MPI

Процедура автоматической настройки параметров библиотеки S-MPI является многоуровневой. Она позволяет достичь оптимального соотношения между затраченным на настройку параметров временем и достигнутой производительностью приложений:

- Настройка умалчиваемых значений параметров библиотеки S-MPI для наиболее распространенных архитектур кластеров и параметров запуска приложений. Этот уровень настройки позволяет библиотеке эффективно работать сразу после ее установки на большинстве кластерных систем;
- Настройка параметров библиотеки S-MPI на специфику конкретной кластерной системы. Предполагается, что эту настройку будет производить администратор кластера при установке пакета библиотеки S-MPI и при изменениях архитектуры кластера. Этот уровень настройки позволяет учесть специфические особенности конкретной кластерной системы и обеспечить высокую производительность большинства приложений, которые будут выполняться на данном кластере. Наибольший эффект этот уровень настройки может дать на кластерах со специфической архитектурой, настройки параметров для которой не были включены в библиотеку S-MPI как умалчиваемые значения;
- Настройка параметров библиотеки S-MPI на специфику конкретного приложения. Этот уровень настройки позволяет достичь максимальной производительности выполнения конкретного приложения на конкретной кластерной системе.

Литература

1. Воронов Г.И., Трущин В.Д., Шумилин В.В., Ежов Д.В. Программный комплекс S-MPI для обеспечения разработки, оптимизации и выполнения высокопараллельных приложений на суперкомпьютерных кластерных системах // Вопросы атомной науки и техники, сер. "Математическое моделирование физических процессов", 2013, №3, с.55-60.
2. Message Passing Interface Forum, “MPI: A Message Passing Interface” // In Proceedings of Supercomputing ’93. IEEE Computer Society Press, November 1993. P. 878–883.
3. Библиотека Intel MPI. – [Электронный ресурс]: http://software.intel.com/en-us/intel-mpi-library.
4. Инструмент настройки OpenMPI (OTPO). – [Электронный ресурс]: http://www.open-mpi.org/projects/otpo/.
5. Gabriel E., Fagg G.E., Bosilca G., at el. Open MPI: Goals, concept, and design of a next generation MPI implementation // Proceedings of the 11th European PVM/MPI Users’ Group Meeting, Budapest, Hungary, September 2004. P. 97–104.
6. Тест на производительность Intel MPI Benchmarks. – [Электронный ресурс]: http://software.intel.com/en-us/articles/intel-mpi-benchmarks.
7. Тест на производительность NAS Parallel Benchmark. – [Электронный ресурс]: http://www.nas.nasa.gov/publications/npb.html.

MALEENA – ИНТЕРАКТИВНАЯ СИСТЕМА МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ БИБЛИОТЕКИ OPENCV

К.А. Дробных, Н.А. Краснояров

Нижегородский госуниверситет им. Н.И. Лобачевского

Описывается интерактивная программная система MaLeEnA, реализующая графический интерфейс доступа к алгоритмам машинного обучения из библиотеки компьютерного зрения OpenCV. Система позволяет: загружать данные, настраивать параметры моделей, тренировать, загружать, сохранять и сравнивать модели, а также предсказывать значения на новых входных данных.

Введение

В настоящее время машинное обучение [1] применяется при решении многих задач компьютерного зрения, анализа речи, компьютерной лингвистики, медицинской диагностики, а также в процессе разработки интеллектуальных игр. Существует большое количество программных библиотек, содержащих реализацию алгоритмов машинного обучения. OpenCV [2] является одной из таких библиотек. Наличие интерактивной программной системы зачастую облегчает процедуру проведения экспериментов с использованием алгоритмов машинного обучения. Задача состоит в том, чтобы разработать графическое приложение, которое упрощает использование алгоритмов машинного обучения из библиотеки OpenCV, позволяет проводить эксперименты и сравнивать полученные результаты.

В настоящей работе описывается MaLeEnA – Machine Learning Environment A – интерактивная программная система для запуска алгоритмов машинного обучения из библиотеки компьютерного зрения OpenCV. MaLeEnA – это кроссплатформенное приложение, реализованное с использованием открытых библиотек OpenCV и Qt. Отличие данной системы от аналогов состоит в том, что она не требует установки дополнительного программного обеспечения и использует широко известную библиотеку компьютерного зрения OpenCV.

Существующие решения

Интерактивные приложения для работы с алгоритмами машинного обучения уже существуют на базе других библиотек, например, утилита rattle [3] для пакета R и приложение Weka [4]. Также необходимо отметить, что существуют и самостоятельные утилиты визуализации и анализа данных (Orange [5], RapidMiner [6] и др.).

Наше приложение имеет возможность автоматического определения формата записи данных (разделитель, имена переменных) в CSV-файле и позволяет задавать необходимое разбиение на обучающую и тестовую части. Поддержка работы с алгоритмом машинного обучения Gradient Boosting Trees является одной из отличительных особенностей разрабатываемой системы, т.к. такая возможность имеется только в системах Weka и RapidMiner.

Основные возможности

MaLeEnA позволяет выполнить полный цикл действий для решения задачи машинного обучения:

- загрузка данных (в известном формате CSV);
- настройка переменных (тип, подмножество используемых переменных);
- настройка параметров модели (реализована для алгоритмов Gradient Boosting Trees [7], Random Trees [8], Extremely Randomized Trees [9], Support Vector Machine [10], Classification And Regression Tree [11]);
- обучение модели, сохранение/загрузка в XML/YAML форматах, которые используются OpenCV;
- интерфейс для сравнения моделей, обученных с помощью предложенных алгоритмов на одних и тех же данных с различными параметрами;
- предсказание на новых данных.

Высокоуровневая архитектура

Система состоит из следующих компонент: Controller, Visualizator, Datafile, Models, Parameters + History.

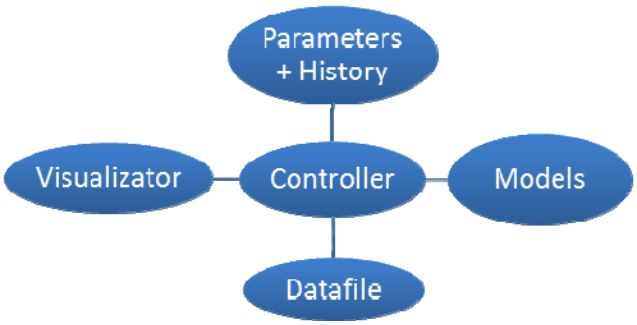


Рис. 1. Диаграмма взаимодействия компонент

Controller обеспечивает связь между компонентами (передача данных, вызовы).

Visualizator отвечает за отображение данных и взаимодействие с пользователем.

Компонент *Datafile* предоставляет возможности загрузки и настройки входных данных.

Models содержит функционал, обеспечивающий работу с моделями с использованием средств OpenCV (загрузка, сохранение, обучение, предсказание и т.п.).

Parameters + History реализует хранение параметров моделей и организует историю параметров обученных моделей (хранит параметры и ошибки обученных ранее моделей для дальнейшего сравнения их качества).

Иллюстрация основных возможностей

Рассмотрим решение задачи машинного обучения на данных Statlog (Heart) Data Set из коллекции UCI Machine Learning Repository [12].

- В процессе загрузки файла heart.dat система определяет формат записи данных (разделитель, имена переменных, признак отсутствующих значений).
- Диалог Variables позволяет выбрать response-переменную и сменить её тип (в данном примере это последняя переменная, categorical). При необходимости также можно изменить тип остальных переменных.
- Осуществляется выбор модели и настройка её параметров.
- Для запуска тренировки входные данные разбиваются на основную и тестовую части, с помощью одного из предложенных типов разбиения: можно указать пропорции разбиения, либо задать его вручную.

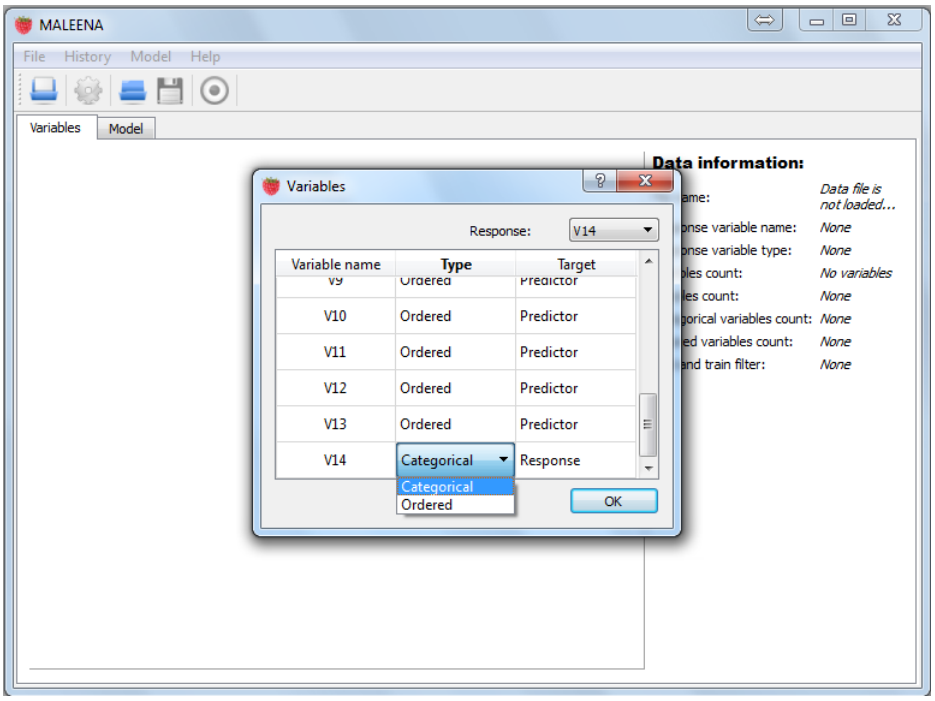


Рис. 2. Настройка переменных

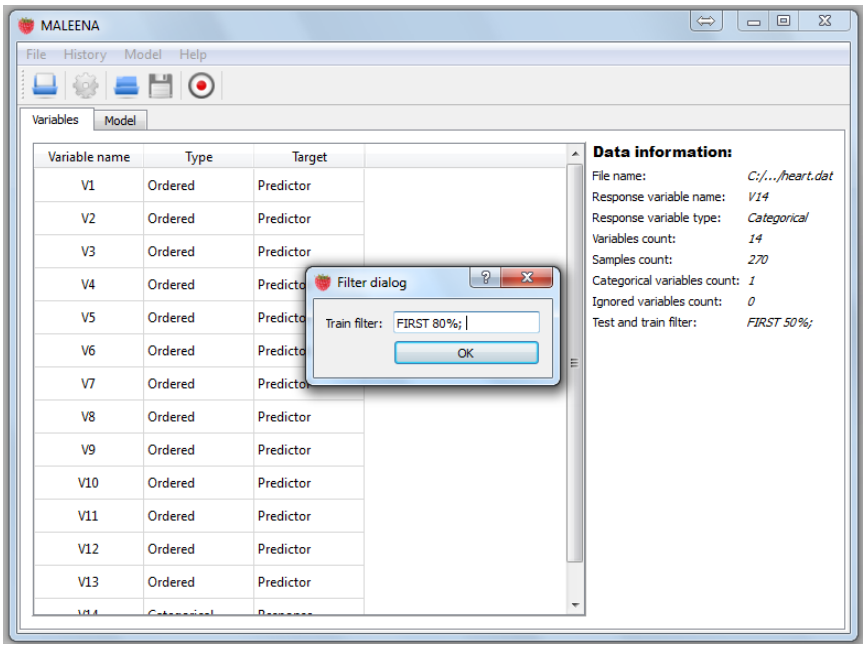


Рис. 3. Выбор разбиения данных на основную и тестовую части

- После тренировки нескольких моделей можно сравнить их и выбрать наиболее подходящую. На рис. 4 представлена диаграмма, используемая для сравнения моделей. Цветом на ней обозначается ошибка модели на тестовой части данных.
- Можно сохранить выбранную модель в файл или использовать её для предсказания значений на новых данных.

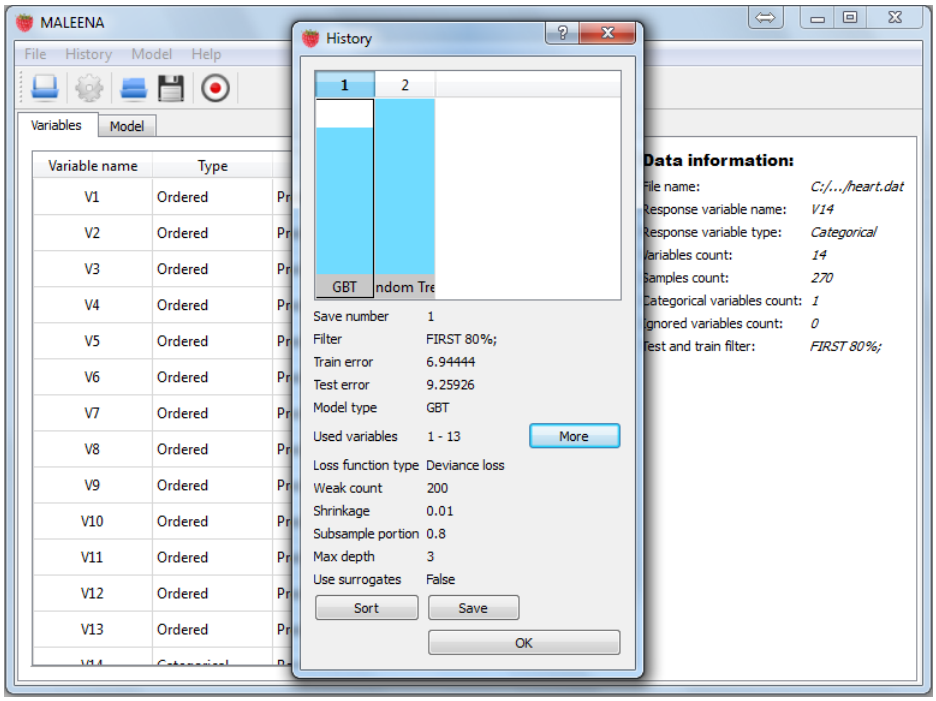


Рис. 4. Выбор разбиения данных на основную и тестовую части

Заключение

В работе описана система MaLeEnA – Machine Learning Environment A – интерактивная программная система, реализующая графический интерфейс доступа к алгоритмам машинного обучения из библиотеки компьютерного зрения OpenCV. Система позволяет загружать данные, настраивать параметры моделей, тренировать, загружать, сохранять и сравнивать модели, предсказывать значения на новых входных данных. Дальнейшее развитие системы подразумевает пользовательскую поддержку, исправление ошибок, доработку и расширение возможностей. Приложение будет выложено в открытый доступ в ближайшее время на сайт: http://ml.vmk.unn.ru/.

Работа выполнена в лаборатории «Информационные технологии» факультета ВМК ННГУ им. Н.И. Лобачевского.

Литература

1. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. – Springer, 2008.
2. OpenCV Documentation – [http://docs.opencv.org/].
3. Rattle – [http://rattle.togaware.com/].
4. Weka – [http://www.cs.waikato.ac.nz/~ml/weka/index.html].
5. Orange – [http://orange.biolab.si/].
6. RapidMiner – [http://sourceforge.net/projects/rapidminer/].
7. Friedman J. Greedy function approximation: the gradient boosting machine // Annals of Statistics. 2001. V. 29, N. 5. P. 1189–1232.
8. Breiman L. Random Forests // Machine Learning. 2001. V. 45, N. 1. P. 5–32.
9. Geurts P., Ernst D., Wehenkel L. Extremely randomized trees // Machine Learning. 2006. V. 63, N. 1. P. 342.
10. Burges C. A Tutorial on Support Vector Machines for Pattern Recognition // Data Mining Knowledge Discovery. 1998. V. 2, N. 2. P. 121–167.

11. Breiman L., Friedman J.H., Olshen R.A., Stone C.J. Classification and Regression Trees. Wadsworth & Brooks, 1984.
12. UCI Machine Learning Repository – [http://archive.ics.uci.edu/ml/].

ОБМЕН ДАННЫМИ В РАСПРЕДЕЛЕННЫХ ПРОГРАММАХ, СОЗДАННЫХ В ТЕХНОЛОГИИ ГРАФО-СИМВОЛИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В.В. Жидченко, П.В. Аболмасов

Самарский государственный аэрокосмический университет им. С.П. Королёва

Приведены концепции технологии графо-символического программирования, позволяющей разрабатывать параллельные программы в визуальной форме. Описан способ реализации модели общей памяти, используемый в технологии при создании программ для распределенных вычислительных систем.

Введение

В настоящее время параллельные суперкомпьютеры рассматриваются как один из основных инструментов для решения задач моделирования при проектировании сложных систем. Однако на практике их применение затрудняется высокой сложностью разработки программ для суперкомпьютерных систем. Так, создание программы для распределенного кластера требует от разработчика глубоких знаний в области программирования, как минимум, знание стандарта MPI и языка C. Ясно, что специалист в предметной области, например проектирования авиационных двигателей, не всегда обладает такими знаниями (а может быть, и не должен обладать). В результате появляется тандем специалиста в предметной области и программиста, а вместе с ним – проблемы коммуникаций, взаимопонимания и следующее за ними увеличение сроков разработки. В такой ситуации представляется актуальным создание простого, наглядного способа описания параллельных алгоритмов и средства автоматической генерации параллельных программ для целевой платформы, скрывающих от пользователя сложности реализации.

Технология графо-символического программирования (технология ГСП), разработанная и развиваемая на кафедре программных систем СГАУ, ставит своей целью решение указанной задачи [1]. Технология ГСП определяется как технология проектирования и кодирования алгоритмов программного обеспечения на основе графического представления программ, с целью полной или частичной автоматизации процессов проектирования, кодирования и тестирования программного обеспечения.

Существенную трудность при создании программы для распределенной вычислительной системы представляет задача управления потоками данных. В сравнении с моделью распределенной памяти, модель общей памяти является более привычной для прикладного программирования. Основополагающей целью при проектировании механизма организации памяти в технологии ГСП являлось желание избавить разработчика от необходимости ручного управления данными в вычислительных системах с распределенной памятью.

В настоящей статье описывается реализация модели общей памяти в технологии ГСП. Для реализации поставленной задачи был использован шаблон проектирования «свойство», который не реализован в языке C++, но который может быть реализован доступными средствами языка.

Концептуальное описание технологии ГСП

Модель параллельной программы в технологии ГСП (граф-модель или граф-программа) представляется четверкой <D, F, P, G>, где D – множество данных некоторой предметной области, F – множество операторов, определенных над данными предметной области, P – множество предикатов, действующих над структурами данных предметной области, G = {A, Ψ, Φ, R} – ориентированный помеченный граф, называемый агрегатом (рис. 1). A = {A1, A2, …, An} – множество вершин графа. Каждая вершина Ai помечена локальным оператором fi(D)∈F. На графе задано множество дуг управления Ψ = { Ψ1i1, Ψ1i2, …, Ψjm} и множество дуг синхронизации Φ = {Φ1i1, Φ1i2, …, Φjl}. R – отношение над множествами вершин и дуг, определяющее способ их связи. Дуга управления, соединяющая любые две вершины Ai и Aj, имеет три метки: предикат pij(D) ∈P, приоритет k(Ψij) ∈N и тип дуги T(Ψij) ∈N. Каждая дуга синхронизации Φij помечена сообщением μij∈N. В классической (последовательной) модели вычислительного процесса, используемой в технологии ГСП, отсутствуют дуги синхронизации, которые вводятся в модель параллельного вычислительного процесса для решения задач синхронизации между его различными участками.

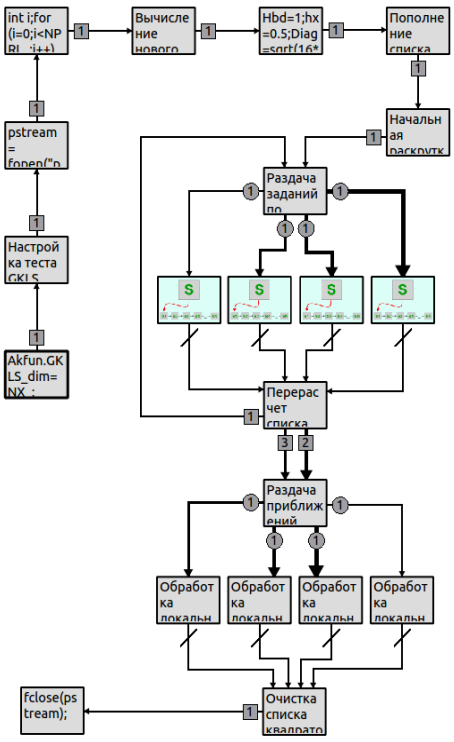


Рис. 1. Граф-модель параллельного алгоритма нахождения глобального экстремума

Под предметной областью программирования (ПО) будем понимать некоторую среду программирования, имеющую общую цель – разработку программного обеспечения автоматизации расчетов в некоторой области практических интересов (авиационные двигатели, бизнес и т.д.), общую область данных и общую область знаний.

Развитие вычислительного процесса, описываемого моделью, ассоциируется с переходами из вершины в вершину по дугам управления. При этом переход по дуге управления возможен лишь в случае истинности предиката, которым она помечена. Если несколько предикатов, помечающих исходящие из вершины дуги, одновременно становятся истинными, переход осуществляется по наиболее приоритетной дуге. Функционирование модели начинается с выполнения оператора f0(D), помечающего начальную вершину A0.

Граф-модель в данном случае заменяет текстовую (вербальную) форму описания алгоритма программы.

Этапы разработки граф-программ в технологии ГСП

1. Создание словаря данных ПО. На данном этапе создаются новые типы и структуры данных, а также происходит накопление словаря данных, где хранится информация обо всех переменных программы.
2. Разработка базовых модулей. Это этап традиционного текстового программирования, на котором программист создает исходные тексты небольших подпрограмм – базовых модулей, учитывая требования стандарта ГСП к оформлению этих текстов.

3. Создание объектов ПО. Этот этап производится автоматически после привязки формальных параметров базовых модулей к фактическим данным предметной области. Объекты, формируемые на основе базовых модулей, называются акторами и формируют множество операторов F модели ГСП.
4. Конструирование агрегатов. На этапе графического программирования пользователь может создать графовый образ новой программы.

Разработанный и отлаженный агрегат, в свою очередь, может быть использован в качестве исходного материала при конструировании следующих агрегатов. Таким образом, в общем случае агрегат имеет иерархическую структуру.

Эффективность программирования в технологии ГСП возрастает по мере развития пользователем своей среды программирования. Доля текстового программирования с традиционной трудоемкой отладкой постепенно снижается, и программирование перерастает в конструирование агрегатов из надежных программных модулей. Отладка при этом заключается только в корректировке структуры графа.

Стандарт хранения и использования данных в технологии ГСП

В технологии ГСП используется внутренний стандарт при организации межмодульного информационного интерфейса [2]. Стандарт обеспечивается выполнением пяти основных правил:

1. Вводится единое для всей предметной области программирования хранилище данных, актуальных для всей области, – словарь данных ПО.
2. В пределах ГСП описание типов данных размещается централизованно в архиве типов данных.
3. В базовых модулях в качестве механизма доступа к данным допускается только передача параметров по адресам данных.
4. Привязка данных объектов ПО реализована в паспортах объектов ПО.
5. В технологии ГСП не рекомендуется использовать иные способы организации межпрограммных связей по данным.

Данный подход к организации межмодульного информационного интерфейса приводит к тому, что синтезируемые программные коды и информационные связи «пространственно» отделены друг от друга. Модификация любого из объектов (актора, предиката или агрегата) не требует изменения кодов других объектов, входящих в ПО.

В технологии ГСП предполагается, что словарь данных располагается в общей памяти и одинаково доступен всем объектам граф-программы. Вместе с тем большинство современных суперкомпьютеров являются кластерами и имеют распределенную память. Чтобы обеспечить возможность разработки программ для таких систем, не обременяя пользователя технологии ГСП изучением механизмов обмена данными в распределенных системах, была реализована модель общей памяти. Описанный выше стандарт организации межмодульного информационного интерфейса был расширен понятием локальных данных.

Способ реализации общей памяти в технологии ГСП

В среде выполнения программы выбирается вычислительный узел, на котором будет запущен процесс, отвечающий за хранение переменных ПО. Учитывая аппаратные особенности и топологию вычислительной системы, это может быть узел с наибольшим объемом оперативной памяти или центральный узел, имеющий минимальное время доступа от любого из остальных узлов кластера. Преимущество данного подхода в том, что значительно экономится ресурс памяти на вычислительных узлах, т.к. на узлах память выделяется только под те переменные, которые используются процессом, исполняющимся на данном узле.

Предлагаемый способ обмена данными требует введения понятия диспетчера данных – подпрограммы, выполняющей функции хранения, чтения и модификации данных предметной области.

Диспетчер данных

Диспетчер данных, называемый также менеджером памяти, исполняется в отдельном процессе параллельной программы. Он порождает объект, описываемый классом TPOData, который хранит значения данных предметной области. В каждом из процессов, содержащих параллельные ветви граф-модели, также порождается объект класса TPOData. Однако функции доступа к членам-данным у объекта диспетчера данных и у объектов параллельных ветвей различаются. Диспетчер данных хранит все данные в локальной памяти и для обращения к ним использует обычные указатели. На остальных процессах используется ленивая инициализация памяти под переменную при первом доступе.

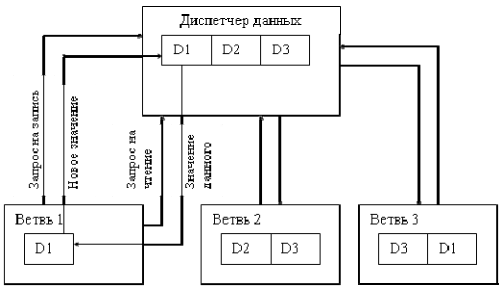


Рис. 2. Модель общих данных в технологии ГСП

В параллельных ветвях граф-модели для чтения или записи некоторого данного осуществляется обращение к диспетчеру памяти с помощью совокупности сообщений. В первом сообщении пересылается запрос на чтение или запись конкретного данного. Каждая переменная из ПО получает уникальный номер, по которому диспетчер памяти может ее идентифицировать. В случае чтения параллельная ветвь переходит к ожиданию ответа от диспетчера данных. При записи во втором сообщении пересылается новое значение данного. Диспетчер данных циклически принимает и обрабатывает запросы параллельных ветвей (рисунок 2).

Обзор класса TPOData

Класс TPOData – это ядро механизма хранения и передачи данных в технологии ГСП. Класс TPOData инкапсулирует все данные ПО и предоставляет доступ к ним через поля-свойства. Полей ровно столько, сколько переменных в ПО, а их названия совпадают с названиями переменных.

Свойство – способ доступа к внутреннему состоянию объекта, имитирующий переменную некоторого типа. Обращение к свойству объекта выглядит так же, как и обращение к полю объекта, но в действительности оно реализовано через вызов функции. При попытке задать значение данного свойства вызывается один метод (*setter*), а при попытке получить значение данного свойства – другой (*getter*).

Реализацией свойств в технологии ГСП являются шаблонные классы __property_rw и __indexed_property_rw. В качестве параметра шаблону передается тип переменной, которая будет доступна через данное свойство, и ссылка на класс, в котором это свойство будет использоваться, в нашем случае это TPOData. После инициализации объекта свойства ему необходимо установить методы доступа. В качестве методов доступа используются соответствующие методы из класса TPOData.

Для простых типов и для массивов используются различные классы свойств. Для простых типов данных необходимо указать по одному методу для получения значения и для его установки. Методы должны иметь следующую сигнатуру:

_T (TPOData:: *)(void); //для getter

_T (TPOData:: *)(_T const &); // для setter, где _T – тип переменной.

Для массивов необходимо указать 2 метода получения значения и 2 метода установки значения. Одна пара методов используется для получения всего массива, другая пара используется для получения одного элемента по индексу.

_T* (TPOData:: *)(void); // для получения всего массива

_T* (TPOData:: *)(_T* const &); // для установки всего массива

_T (TPOData:: *)(_I const &); // для получения одного элемента по индексу

_T (TPOData:: *)(_I const &,_T const &); //для установки одного элемента по индексу

Это один из вариантов реализации свойств в языке C++, спроектированный с учетом особенностей технологий ГСП и MPI.

Класс TPOData и экземпляр класса напрямую недоступен для разработчика граф-программ, и разработчик не обязан знать о его существовании. Когда разработчик обращается к переменной ПО, на самом деле он обращается к полям-свойствам объекта класса TPOData.

Использование локальных переменных в параллельных процессах

Описанные выше механизмы позволяют пользователю технологии ГСП создавать параллельные программы для распределенных систем, оставаясь в рамках концепции общей памяти. При определенных условиях такой подход генерирует эффективные параллельные программы без дополнительных усилий со стороны разработчика. Однако в большинстве случаев для повышения производительности вычислений на распределенной системе (например, на кластере), когда скорость доступа к данным в локальной памяти вычислительного узла существенно отличается от скорости доступа к данным другого узла, нужно искусственно увеличивать долю локальных вычислений. Для этого необходим механизм копирования данных из менеджера памяти в локальную память узла, на котором исполняется параллельная ветвь программы. Таким механизмом может служить механизм передачи сообщений между параллельными ветвями, основным назначением которого является синхронизация параллельных ветвей. Вместо передачи единичного флага, разрешающего запуск конкретной вершины, сообщение может содержать данные предметной области. Дуги синхронизации помечаются данными, пересылаемыми в каждом сообщении. При создании сообщения определяется список переменных для отправки и список соответствующих им переменных для приема. Если переменные векторные, то может быть задан диапазон индексов отправляемых и принимаемых элементов вектора. При этом необходимо обеспечить соответствие размера и типов отправляемых и принимаемых данных.

Множество данных предметной области разбивается на два подмножества: общие данные и локальные данные. Общие данные создаются и инициализируются менеджером памяти. Локальные данные создаются и инициализируются в том процессе, в котором они используются. Принадлежность данных конкретному подмножеству определяется установкой флага «Локальное данное» в свойствах каждого данного. По умолчанию все данные – общие. В передаваемых между вершинами сообщениях в качестве передаваемых и/или принимаемых данных должны участвовать локальные данные.

Заключение

Рассмотренные в статье концепции и механизмы технологии ГСП позволяют скрыть от пользователя технологии детали реализации управления и обмена данными в параллельном вычислительном процессе, способствуя упрощению использования современных суперкомпьютерных систем специалистами в различных предметных областях.

Литература

1. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. Самара: Самар. гос. аэрокосм. ун-т, 1999. – 150 с.

2. Коварцев А.Н., Жидченко В.В. Методы и средства визуального параллельного программирования. Автоматизация программирования: электрон. учеб. пособие. Самара, 2010.

УСКОРЕНИЕ СИМПЛЕКСНЫХ МЕТОДОВ ЛИПШИЦЕВОЙ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

Ю. Жилинскас[1], Д.Е. Квасов[2,3], Р. Паулавичюс[1], Я.Д. Сергеев[2,3]

[1]Вильнюсский университет, Литва
[2]Нижегородский госуниверситет им. Н.И. Лобачевского
[3]Калабрийский университет, Ренде, Италия

Предлагается ряд подходов к ускорению работы алгоритмов липшицевой глобальной оптимизации на основе симплексного разбиения области поиска и множественных оценок константы Липшица. Приводятся результаты численного сравнения новых методов с известными алгоритмами глобальной оптимизации на классах тестовых функций GKLS-генератора.

Введение

В настоящей работе рассматриваются задачи принятия оптимальных решений, которые характеризуются многоэкстремальностью целевых функций (так называемые задачи глобальной оптимизации) и высокой вычислительной стоимостью определения их локальных характеристик. Отмеченные особенности отличают системы, вычисление критериев качества которых связано со сложными и длительными численными расчетами (выполнением компьютерного имитационного моделирования, численным решением сложных систем дифференциальных уравнений и т.п.).

Возможность построения адаптивных схем поиска глобального решения подобных многомерных задач, отличных от переборных схем, связана с наличием неких априорных предположений о свойствах задач. Для многих важных прикладных задач (таких как, например, решение нелинейных уравнений и неравенств, регулирование сложных нелинейных систем и т.д.) типичным является предположение о липшицевости функций (см., например, [1,4,5,7,11,13,19,23,25-27]).

1. Липшицева глобальная оптимизация

Задача поиска глобального минимума многомерной многоэкстремальной функции, удовлетворяющей в допустимой области $D \subset \mathrm{R}^n$ условию Липшица с константой Липшица $0 < L < \infty$, может быть сформулирована следующим образом:

$$\min f(x),\ x \in D, \tag{1}$$

$$|f(x') - f(x'')| \le L\,\|x' - x''\|,\ x', x'' \in D, \tag{2}$$

где $\|\cdot\|$ есть евклидова норма, а допустимая область D задается в виде

$$D = \{x \in \mathrm{R}^n : a_i \le x_i \le b_i,\ i=1,\ldots,n\}. \tag{3}$$

В литературе рассматривается множество различных алгоритмов решения задачи (1)-(3) (см. [1–27]). Они могут быть разделены на четыре группы в зависимости от способа получения оценки константы Липшица L из (2). К первой группе относятся алгоритмы, в которых применяется оценка L, заданная априори (см. [5,6,23]). Вторая группа включает алгоритмы, адаптивно оценивающие в ходе поиска глобальную константу Липшица во всей области D (адаптивная глобальная оценка, см. [1,8,13,20,23,27]). Алгоритмы с использованием адаптивных локальных оценок L_i в различных подобластях D_i поисковой области (см. [10,11,16,25–27]) входят в третью группу. Наконец, к четвертой группе принадлежат алгоритмы, в которых оценка константы Липшица выбирается

из некоторого множества возможных значений (см. [3,9,14,15,21,24]). В силу своей относительной простоты методы последней группы нашли достаточно широкое применение при решении практических задач (см. [9,11,19,25]) и привлекли внимание многих исследователей. Именно этой группе методов уделено основное внимание в данной работе.

Для упрощения выбора новых точек испытаний и получения нижних границ значений функции $f(x)$ в области D во многих многомерных методах применяется техника адаптивного разбиения области поиска D на множество подобластей D_i, а теоретическое исследование алгоритмов разбиения может быть проведено в рамках единых схем (см. [4,11,13,23,27]). Например, применяются разбиения области D на гиперинтервалы с вычислением функции в их центральных точках [6,15,23]. Широко используются также разбиения с проведением испытаний функции в вершинах, соответствующих главной диагонали каждого гиперинтервала (диагональные схемы разбиения, [8,11,12,23,24]). В настоящей работе рассматриваются более сложные стратегии разбиения, основанные на симплексах [2,4,19,21,22,23]. Для ряда прикладных задач симплексные разбиения могут оказаться более эффективными по сравнению с другими техниками [4,19,21].

Необходимо отметить, что использование только глобальной информации о поведении целевой функции в ходе ее минимизации может замедлить сходимость алгоритма к точке глобального минимума. Одним из способов преодоления такой сложности является (см. ссылки в [1,4,11,25]) остановка метода глобального поиска и запуск некоего локального алгоритма, призванного улучшить найденное решение. При этом, однако, возникает непростой вопрос об определении момента завершения глобального алгоритма: его преждевременная остановка может привести к потере глобального решения, в то время как поздняя остановка существенно замедляет поиск.

Именно поэтому столь важным аспектом в методах глобальной оптимизации является учет локальной информации о поведении целевой функции. Например, в работах [10,11,27] была предложена схема локальной настройки, основная идея которой заключается в сопряжении локальной и глобальной информации при адаптивном оценивании локальных констант Липшица в различных подобластях поисковой области. Использование локальной настройки позволяет значительно ускорить работу методов липшицевой глобальной оптимизации. Наличие информации о производных целевой функции также даёт очень важную локальную информацию о характере функции (см. [11,16,17,18,27]). Весьма перспективными представляются также техника уточнения полученного в ходе поиска текущего оптимального решения (см. [17,18,26]) и двухфазная схема поиска глобального минимума (см. [11,24,26]).

2. Двухфазные симплексные методы глобальной оптимизации с множественными оценками константы Липшица

В докладе рассматриваются варианты ускорения симплексных методов DISIMPL-C и DISIMPL-V [21] с множественными оценками константы Липшица в рамках двухфазной схемы поиска [11,24]. В методе DISIMPL (DIviding SIMPLices, по аналогии с названием метода DIRECT – DIviding RECTangles [15]) с суффиксом (-C) целевая функция вычисляется в центрах получаемых в ходе разбиения симплексов, в то время как в методе DISIMPL с суффиксом (-V) испытания функции проводятся в вершинах симплексов, избегая возможного дублирования информации при помощи специально поддерживаемой базы вершин (см. рис. 1). На каждой итерации методов определяются недоминируемые симплексы (см. [11,15,21]), некоторые из которых подвергаются дальнейшему разбиению с последующими новыми испытаниями целевой функции. В работе [21] проведен теоретический и экспериментальный анализ обоих методов.

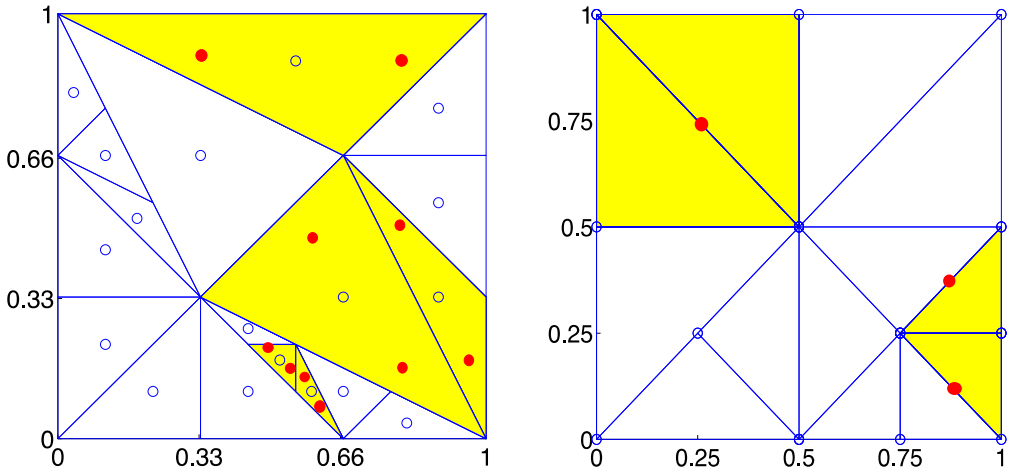


Рис. 1. Разбиения области поиска $[0,1]^2$ при помощи методов DISIMPL-C (слева) и DISIMPL-V (справа): желтым цветом выделены симплексы для нового разбиения и красным – новые точки вычисления целевой функции после проведения разбиений

Использование двухфазной схемы поиска позволяет значительно ускорить работу обоих методов и приводит к построению двух новых алгоритмов, Gb-DISIMPL-C и Gb-DISIMPL-V (префикс Gb- означает «глобально-ориентированный» от англ. Globally-biased). При достижении достаточного уровня разбиения симплексов около точки текущего минимального значения функции методы переключаются на исследование больших подобластей, которые могут содержать лучшее решение. Так как около текущей лучшей точки проводится большое число разбиений, ее окрестность содержит лишь малые симплексы, в то время как большие могут находиться только далеко от нее. Тем самым по сравнению с методами DISIMPL-C и DISIMPL-V из [21] предлагаемые в работе алгоритмы балансируют глобальную и локальную информацию более совершенным образом с целью обеспечения скорейшей сходимости к точкам глобального минимума сложных многоэкстремальных функций.

В табл. 1–3 и на рис. 2 приведены результаты численного сравнения методов Gb-DISIMPL-C и Gb-DISIMPL-V с их первоначальными версиями из [21] и с методами DIRECT и DIRECT*l*, широко используемыми (в силу их простоты) при решении прикладных задач глобальной оптимизации. Сравнение проведено на двухстах функциях из двух дифференцируемых GKLS-классов «простой» и «сложной» структуры (см. [11, гл.5] для детального описания техники проведения экспериментов с тестовыми классами на основе GKLS-генератора). Как видно из табл. 1–3, новые методы при значительно меньшем числе испытаний функций (табл. 1 и 2) обеспечивают весьма хороший уровень исследования области поиска (табл. 3).

Таблица 1. Среднее число испытаний на двумерных GKLS-классах

Класс	DIRECT	DIRECT*l*	DISIMPL-C	Gb-DISIMPL-C	DISIMPL-V	Gb-DISIMPL-V
«Простой»	198.89	292.79	200.80	187.72	192.93	173.09
«Сложный»	1063.78	1267.07	1189.60	701.28	1003.56	535.95

Таблица 2. Максимальное число испытаний на двумерных GKLS-классах

Класс	DIRECT	DIRECT*l*	DISIMPL-C	Gb-DISIMPL-C	DISIMPL-V	Gb-DISIMPL-V
«Простой»	1159	2318	960	938	773	418
«Сложный»	3201	3414	6696	2624	2683	1745

Таблица 3. Максимальное число подобластей на двумерных GKLS-классах

Класс	DIRECT	DIRECT*l*	DISIMPL-C	Gb-DISIMPL-C	DISIMPL-V	Gb-DISIMPL-V
«Простой»	1159	2318	960	938	1419	735
«Сложный»	3201	3414	6696	2624	5085	3281

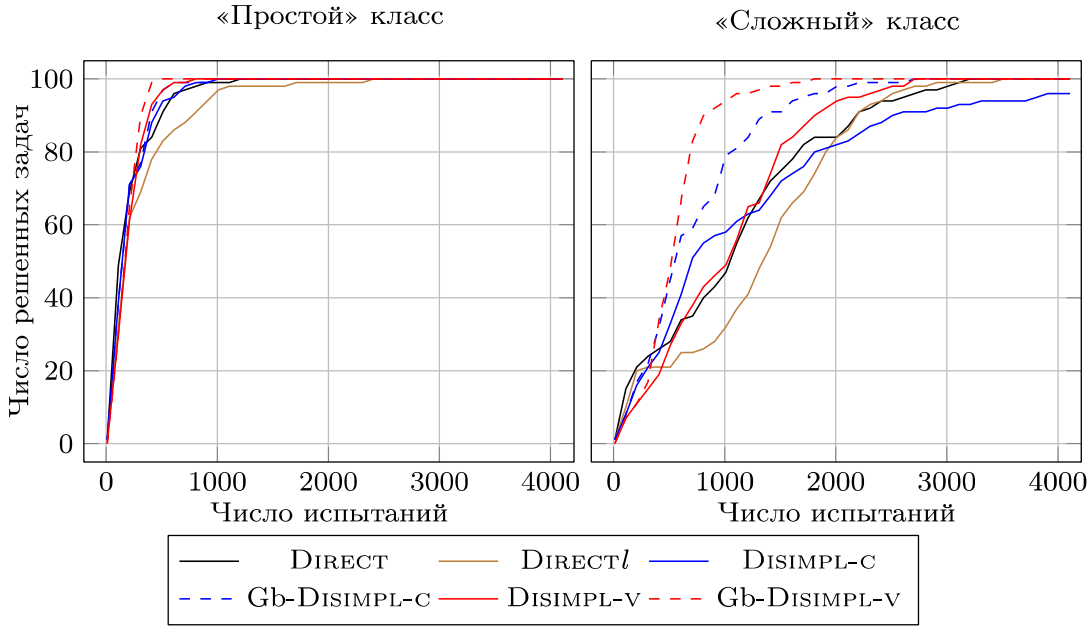


Рис. 2. Операционные характеристики [4,13,27] сравниваемых методов при решении задач «простого» и «сложного» GKLS-классов размерности n=2

По результатам проведенных экспериментов можно заключить, что при решении сложных многомерных многоэкстремальных задач наблюдается значительное преимущество методов с использованием двухфазной стратегии поиска по отношению к глобальным алгоритмам без использования подобной стратегии ускорения. С ростом размерности задач ожидается еще более существенное преимущество данной стратегии поиска глобального минимума. Кроме того, рассмотренная симплексная стратегия разбиения может быть успешно распараллелена (см. [1,22,27]), что дает дополнительные возможности для ускорения поиска.

Работа Д.Е. Квасова и Я.Д. Сергеева выполнена при финансовой поддержке Минобрнауки России, государственное соглашение о предоставлении гранта № 14.B37.21.0878. Постдокторантура Р. Паулавичюса финансируется проектом Структурных фондов Европейского союза «Внедрение Постдокторантуры в Литве».

Литература

1. 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.
2. Городецкий С.Ю. Многоэкстремальная оптимизация на основе триангуляции области // Вестник ННГУ: Математическое моделирование и оптимальное управление. – 1999. – Т. 2, № 21. – С. 249–268.
3. Городецкий С.Ю. Несколько подходов к обобщению метода Direct на задачи с функциональными ограничениями // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2013. – № 6. – В печати.

4. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. – Н. Новгород: Изд-во ННГУ, 2007.
5. Евтушенко Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации. – М.: Наука, 1982.
6. Евтушенко Ю.Г., Посыпкин М.А. Применение метода неравномерных покрытий для глобальной оптимизации частично целочисленных нелинейных задач // Журн. вычисл. мат. и матем. физ. – 2011. – Т. 51, № 8. – С. 1376–1389.
7. Жиглявский А.А., Жилинскас А.Г. Методы поиска глобального экстремума. – М.: Наука, 1991.
8. Квасов Д.Е., Сергеев Я.Д. Многомерный алгоритм глобальной оптимизации на основе адаптивных диагональных кривых // Журн. вычисл. мат. и матем. физ. – 2003. – Т. 43, № 1. – С. 42–59.
9. Квасов Д.Е., Сергеев Я.Д. Методы липшицевой глобальной оптимизации в задачах управления // Автоматика и телемеханика. – 2013. – №9. – С. 3–19.
10. Сергеев Я.Д. Одномерный детерминированный алгоритм глобальной минимизации // Журн. вычисл. мат. и матем. физ. – 1995. – Т. 35, № 5. – С. 705–717.
11. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008.
12. Сергеев Я.Д., Квасов Д.Е. Адаптивные диагональные кривые и их программная реализация // Вестник ННГУ. Математическое моделирование и оптимальное управление. – 2001. – Т. 2, № 24. – С. 300–317.
13. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. Информационно-статистический подход. – М.: Наука, 1978.
14. Gablonsky J.M., Kelley C.T. A locally-biased form of the DIRECT algorithm // J. Global Optim. – 2001. – Vol. 21, № 1. – P. 27–37.
15. Jones D.R., Perttunen C.D., Stuckman B.E. Lipschitzian optimization without the Lipschitz constant //J. Optim. Theory Appl. – 1993. – Vol. 79, № 1. – P. 157–181.
16. Kvasov D.E., Sergeyev Ya.D. Univariate geometric Lipschitz global optimization algorithms // Numer. Algebra Contr. Optim. – 2012. – Vol. 2, № 1. – P. 69–90.
17. Kvasov D.E., Sergeyev Ya.D. Lipschitz gradients for global optimization in a one-point-based partitioning scheme // J. Comput. Appl. Math. – 2012. – 236(16) – P. 4042–4054.
18. Lera D., Sergeyev Ya.D. Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives // SIAM J. Optim. – 2013. – Vol. 23, № 1. – P. 508–529.
19. Paulavičius R., Žilinskas J. Simplicial Global Optimization. – N.Y.: Springer, 2013.
20. Paulavičius R., Žilinskas J. Influence of Lipschitz bounds on the speed of global optimization // Technological Econ. Developm. of Economy. – 2012. – 18(1). – P. 54–66.
21. Paulavičius R., Žilinskas J. Simplicial Lipschitz optimization without the Lipschitz constant // J. Global Optim. – 2013. – In Press.
22. Paulavičius R., Žilinskas J., Grothey A. Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds // Optim. Lett. – 2010. – Vol. 4, № 2. – P. 173–183.
23. Pintér J.D. Global Optimization in Action. – Dordrecht: Kluwer Publishers, 1996.
24. Sergeyev Ya.D., Kvasov D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants // SIAM J. Optim. – 2006. – Vol. 16, № 3. – P. 910–937.
25. Sergeyev Ya.D., Kvasov D.E. Lipschitz global optimization // In Wiley Encyclopedia of Operations Research and Management Science (Ed. by Cochran J.J. et al.). – New York: Wiley, 2011. – Vol. 4. – P. 2812–2828.
26. Sergeyev Ya.D., Strongin R.G., Lera D. Introduction to Global Optimization Exploiting Space-Filling Curves. – N.Y.: Springer, 2013.

27. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms. – Dordrecht: Kluwer Academic Publishers, 2000.

ИСПОЛЬЗОВАНИЕ СУПЕРКОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ ДЛЯ РАСЧЕТА СТРУКТУР МОЛЕКУЛ ОРГАНИЧЕСКОЙ ФОТОНИКИ

А.С. Зайцев, П.В. Белозёрова

Костромской государственный университет им. Н.А.Некрасова

Проведен расчет структуры и параметров молекул фталоцианина и его производных с атомом кислорода O, гидроксилом OH, двумя атомами кислорода OO, атомом кислорода и гидроксилом OOH на 32 ядрах вычислительного кластера в программе NWChem. Сделано сравнение полученных результатов с результатами, рассчитанными последовательной программой Firefly на компьютере AMD Phenom/Linux Firefly version под Linux. Отмечены ускорение вычислений и их более высокая точность.

В настоящее время благодаря уникальным электронным свойствам производных фталоцианина их активно используют в органических полупроводниковых устройствах [1], в том числе для изготовления TFT-матриц и активного слоя CD-R дисков, солнечных элементах, фотодиодах и др. Для вычислений электронной энергии и параметров электронной структуры таких молекул требуются значительные вычислительные ресурсы, поэтому разумно использовать суперкомпьютерные кластеры.

Для таких вычислений создано большое количество коммерческих и свободнораспространяемых программ квантово-химических расчетов. Наибольший интерес представляют программы, относящиеся к категории открытого программного обеспечения (Open Source) в связи с их доступностью и возможностью изучения и модификации исходного кода. В данной работе была использована программа NWChem [2], хорошо зарекомендовавшая себя при вычислениях на многопроцессорных кластерах.

Использовался вычислительный кластер Костромского государственного университета им. Н.А. Некрасова. Параллельные вычисления проводились на 4 узлах по 8 ядер (всего 32 ядра). Оперативная память использовалась полностью.

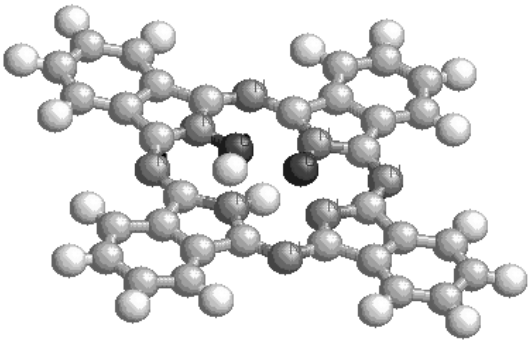


Рис. 1. Рассчитанная структура производной молекулы фталоцианина с атомом кислорода O и гидроксилом OH

Проведен расчет структуры и параметров разных производных молекул фталоцианина: с атомом кислорода O, гидроксилом OH, двумя атомами кислорода OO, атомом кислорода и гидроксилом OOH (рис. 1). Результаты вычислений сравнили с результатами, полученными с помощью последовательной программы Firefly на компьютере AMD Phenom/Linux Firefly version под Linux. По точности результаты на кластере получились лучше, чем на AMD (энергия меньше, что должно быть по теории). Это свя-

зано с тем, что на AMD использовался более легкий базис: не по три d,p-функции на атом, а по одной. Программа при этом считает быстрее и требует меньше ресурсов, но точность ухудшается.

По термодинамическим функциям – энтальпии и энтропии результаты практически совпали. Геометрические и силовые параметры оптимизированной структуры в обоих случаях были близки. По скорости расчетов процессорное время на кластере получилось меньше в 37.7 раза, чем на AMD, а общее уменьшилось в 5.2 раза. Особенностью программы NWChem является требование больших ресурсов памяти. Но это может быть особенностью сборки и конфигурации операционной системы.

Литература

1. Hiroko Yamada, Naoko Kamio, Manami Kawano. Photocurrent generation by benzoporphyrin films prepared by a solution process // J. Porphyrins Phthalocyanines. – 2007. – № 11. – P. 383-390.
2. Valiev M., Bylaska E.J., Govind N., Kowalski K., Straatsma T.P, van Dam H.J.J., Wang D., Nieplocha J., Apra E., Windus T.L., de Jong W.A. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations // Comput. Phys. Commun. – 2010. – V. 181. – P. 1477-1489.

МЕТОД ПАРАЛЛЕЛЬНОГО ВЫЧИСЛЕНИЯ ДЕСКРИПТОРА, ОСНОВАННОГО НА ИЗВЛЕЧЕНИИ ИНФОРМАТИВНОЙ ЧАСТИ ИЗОБРАЖЕНИЯ, ДЛЯ РАСПОЗНАВАНИЯ ОБЪЕКТОВ РАЗНЫХ КЛАССОВ

Р.К. Захаров

Самарский государственный аэрокосмический университет им. С.П. Королёва

Рассматривается классификация и распознавание объектов различных классов на изображениях. Исследуется параллельная реализация алгоритма, основанного на объединении дескрипторов Histogram of Oriented Gradients (HOG) и Local Binary Patterns (LBP) и извлечении из них наиболее информативной части. Описана схема решения задачи для объектов различных классов. Представлены результаты вычислительных экспериментов.

Введение

В настоящее время для решения многих практических задач используются системы компьютерного зрения (системы видеонаблюдения, системы помощи водителю и другие). В работе рассматривается задача распознавания и классификации объектов разных классов, таких как мотоциклы, автомобили, люди и др., на изображениях с использованием классификатора SVM. Представлены результаты вычислительных экспериментов на базе данных изображений PASCAL Visual Object Challenge-2007 (VOC2007, http://pascallin.ecs.soton.ac.uk/challenges/VOC).

Важной частью распознавания и локализации объекта является выбор признакового описания объекта. В области компьютерного зрения разработано множество дескрипторов для описания изображений HOG [1], LBP [2]. В работе представлена схема формирования дескриптора, основанного на комбинации дескрипторов HOG и LBP и извлечении из объединённого дескриптора более информативной части с помощью метода главных компонент PCA [3]. Представлена параллельная схема работы метода.

1. Постановка задачи

Общая постановка задачи распознавания образов следующая. Предполагается, что имеется M изображений каждого из K объектов. Каждое изображение представляется вектором $x = [x_1, x_2, \dots, x_N]^T$ размерности N, где $x_1, x_2, \dots, x_N$ – признаки. Векторы, соответствующие изображениям одного объекта, составляют класс. Совокупность векторов признаков всех классов образует обучающую выборку. Решение задачи распознавания состоит в конструировании решающей функции $f: R^N \to \{0,1, \dots, K\}$, 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 играет также используемая при распознавании мера близости и построенное на ее основе решающее правило.

Для решения задачи классификации объектов на изображении широко используется метод машины опорных векторов, а в качестве признаков наиболее популярными являются гистограммы ориентированных градиентов (HOG) [1], также часто использу-

ют метод Viola-Jones, который показывает хорошие результаты для детектирования человеческих лиц в кадре, и различные комбинированные методы.

В настоящей работе ставится задача провести исследования метода формирования дескриптора, основанного на информации о форме объекта, и дескриптора, основанного на текстурных характеристиках объекта, и выделить информативные части из объединённого дескриптора. Дескрипторы HOG и LBP являются довольно распространенными дескрипторами для локализации и классификации объектов на изображениях.

Новизна работы заключается в формировании метода для получения дескриптора, основанного на дескрипторах HOG и LBP. Для построения дескрипторов используется объединение дескрипторов и извлечение из них более важной информации. Классификация происходит с помощью метода, основанного на машине опорных векторов на сформированных дескрипторах.

2. Описание алгоритма

Предлагаемый метод основан на комбинации двух дескрипторов – дескриптора, основанного на информации о форме объекта, и дескриптора, составленного с использованием локальных бинарных признаков на изображении. На рис. 1 изображена общая схема классификации для предлагаемого метода

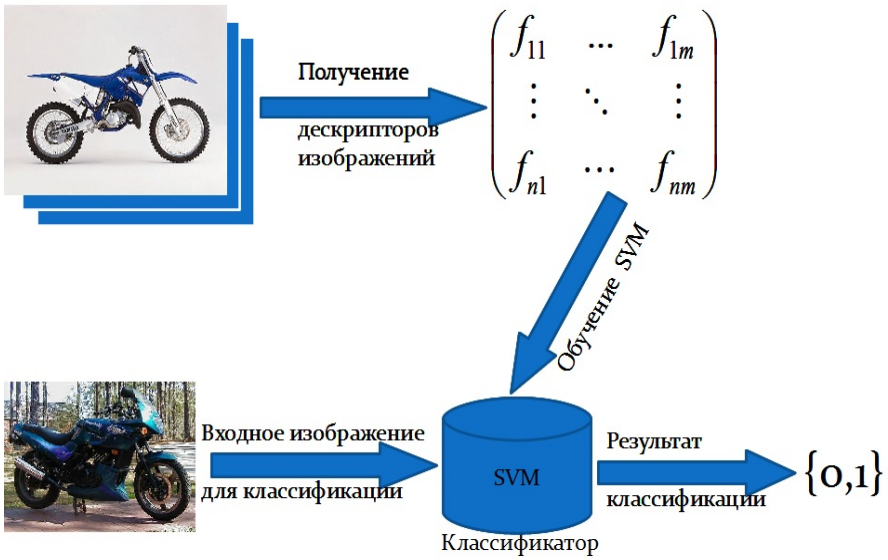


Рис. 1. Общая схема классификации на основе SVM для распознавания и классификации изображений

Далее опишем общие схемы для классификации объектов. В данной работе исследовалось извлечение более существенной информации из объединенных дескрипторов. Далее рассмотрим последовательно каждый из пунктов и опишем их схему работы и кратко рассмотрим формирование самих дескрипторов HOG и LBP.

3. Схема параллельной работы метода сокращения размерности для дескрипторов HOG и LBP

Для сокращения размерности был выбран метод PCA [3]. Метод PCA (Principal Component Analysis) анализирует дескрипторы и выделяет более информативные их части. Ниже на рис. 2 приведём схему алгоритма.

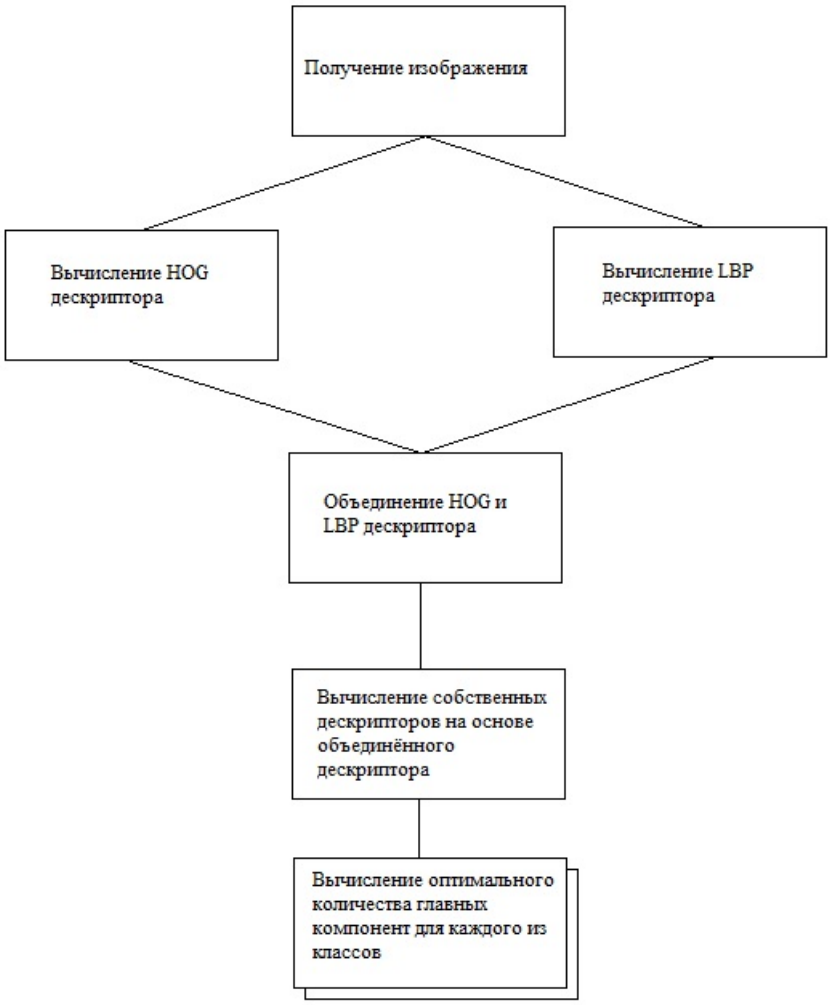


Рис. 2. Схема получения нового дескриптора на основе дескрипторов HOG и LBP с применением метода сокращения размерности PCA и оптимального вычисления количества главных компонент для каждого из тестируемых классов

Ниже представлены основные шаги для выделения главных компонент в дескрипторах.

1. Подготовка данных

На этом шаге обучающая выборка дескрипторов должна быть представлена в виде вектора Γ_i, где i – номер дескриптора.

2. Вычитание среднего из выборки

Вычисляется средний дескриптор из выборки, затем он вычитается из выборки Γ. Для объединённого дескриптора HOG и LBP применяется одна евклидова метрика, с помощью которой находится средний вектор

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i, \tag{1}$$

$$\Phi_i = \Gamma_i - \Psi. \tag{2}$$

3. Вычисление ковариационной матрицы

На этом шаге ковариационная и информационная матрица C и L соответственно вычисляются следующим образом

$$C = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^T, \quad (3)$$

$$L = \frac{1}{M}\sum_{i=1}^{M} \Phi_i^T \Phi_i. \quad (4)$$

4. Выбор главных компонент

Вычисляются собственные числа и векторы *v* для матриц *C* и *L*. Было доказано [3], что ненулевые собственные числа этих матриц совпадают. Размерность матрицы *L* много меньше размерности матрицы *C*, это позволяет вести обработку на матрице меньшей размерности, что повышает эффективность метода и программы в целом.

5. Собственные дескрипторы

Собственные дескрипторы вычисляются следующим преобразованием собственных векторов информационной матрицы

$$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k. \quad (5)$$

Для распознавания можно взять первые M′ собственных дескрипторов, расположенных по убыванию соответствующих им собственных значений. Фактически это размерность базиса, на который мы будем проецировать другие векторы. Значение M′ выбирается экспериментальным путём. В данной работе количество главных компонент выбирается исходя из работы алгоритма, который параллельно запускает вычисления с различным числом главных компонент, наилучший результат выбирается для дальнейшей работы.

6. Применение собственных дескрипторов для дальнейшего распознавания

Нормируем каждый собственный дескриптор и тогда получим, что собственные дескрипторы образуют ортонормированный базис в пространстве дескрипторов. Будем проецировать на этот базис все дескрипторы и получать координаты в новом базисе следующим образом

$$\omega_k = u_k^T(\Gamma_{new} - \Psi), \text{где } k = 1 \dots M', \quad (6)$$

$$\Omega_{new}^T = [\omega_1, \omega_2, \dots, \omega_{M'}]. \quad (7)$$

Далее полученный дескриптор переходит на вход к классификатору.

Заключение

В работе представлен подход для распознавания различных классов изображений на основе модифицированного дескриптора, основанного на дескрипторах LBP и HOG и извлечении из них наиболее информативной части с помощью метода главных компонент PCA. Количество главных компонент выбирается автоматически, решая оптимизационную задачу на тренировочной выборке, параллельно запускается алгоритм с несколькими главными компонентами, наилучший результат принимается для дальнейшей работы метода. В результате эксперимента показано, что метод достаточно эффективен и надёжен.

Достоинством предлагаемого метода является:

1. Улучшение качества классификации объектов;
2. Сокращение размерности дескриптора, что позволяет производить вычисления быстрее чем дескриптор основанный на объединении дескрипторов LBP и HOG;
3. Автоматический выбор оптимального количества главных компонент для объектов разных классов.

Литература

1. Dalal N., Triggs W. Histograms of Oriented Gradients for Human Detection // IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05. – 2005. – Vol. 1(3). – P. 886-893.
2. Pietikäinen M., Hadid A., Zhao G. and Ahonen T. Computer Vision Using Local Binary Patterns, Springer. 2011. – [http://www.springer.com/mathematics/book/978-0-85729-747-1].
3. Miranda A.A., Le Borgne Y.-A., Bontempi G. New Routes from Minimal Approximation Error to Principal Components // Neural Processing Letters. Vol. 27, N. 3. 2008.
4. Lowe D.G. Object recognition from local scale-invariant features // Proceedings of the International Conference on Computer Vision. 1999. 2. P. 1150–1157.
5. Matas J., Chum O., Urban M., Pajdla T. Robust wide baseline stereo from maximally stable extremal regions // Proc. of British Machine Vision Conference. 2002. P. 384-396.

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В ЗАДАЧЕ ПОИСКА ЭДМ ЭЛЕМЕНТАРНЫХ ЧАСТИЦ

А.Н. Иванов

Санкт-Петербургский госуниверситет, Россия
Institute for Nuclear Physics, Forschungszentrum Juelich, Germany

Приводится формулировка проблемы длительного моделирования спин-орбитального взаимодействия заряженных частиц в электромагнитных полях для задачи поиска электрического дипольного момента элементарных частиц. Обосновывается необходимость разработки специальных численных методов интегрирования дифференциальных уравнений. Наряду с анализом существующих подходов приводится описание разработанного параллельного метода решения систем обыкновенных дифференциальных уравнений, основанного на численной реализации нелинейного матричного формализма теории групп Ли. Описанный метод обладает свойством естественного параллелизма, что делает возможным его реализацию на специализированных матричных GPU-ускорителях.

Введение

Наличие электрического дипольного момента (ЭДМ) элементарных частиц свидетельствует о нарушении как пространственной, так и временной четности, что, в рамках CPT-теоремы, свидетельствует о нарушении CP-инвариантности. Ненулевой ЭДМ в этом случае может служить сигналом для развития «новой физики» за пределами Стандартной модели [1].

Одним из подходов в измерении ЭДМ является изучение спиновой динамики заряженных частиц при движении в накопительном кольце [2]. Такой подход основывается на анализе поляризованных пучков в течение длительных интервалов времени. Ввиду маленького ожидаемого значения ЭДМ ($10^{-26} e \cdot cm$), а как следствие – и его влияния на динамику частиц необходимо обеспечить время жизни пучка на уровне 1000 секунд, что соответствует миллиардам оборотов частиц в накопительном кольце. Исследования, посвященные данной тематике, в настоящее время ведутся в научно-исследовательском центре Юлих Германии.

Отметим, что традиционные пошаговые методы интегрирования динамических систем в данном случае неприменимы. Во-первых, они не обеспечивают приемлемое время вычислений, во-вторых, глобальная ошибка таких методов растет с каждым шагом интегрирования. Причем общее число необходимых шагов для одной частицы оценивается величиной 10^{12} и зависит от задаваемой локальной ошибки. Кроме того, на таких длительных временных интервалах интегрирования решающим фактором является сохранение физических свойств рассматриваемой системы. В физике частиц это в основном условие симплектичности ввиду гамильтонова характера динамики. Существующие симплектические методы пошагового интегрирования описываются неявными схемами [3], что в разы увеличивает затраченное время расчетов.

В рамках данного проекта для решения указанных задач используются методы интегрирования обыкновенных дифференциальных уравнений (ОДУ), основанные на построении отображений. Такие подходы позволяют оценивать нелинейное отображение, переводящее множество начальных состояний в конечное, которое соответствует полному обороту частицы в накопительном кольце.

1. Обзор известных решений

Впервые теория групп Ли для решения обыкновенных дифференциальных уравнений была применена Алексом Драгтом [4], профессором университета Мэрилэнда. Им была написана программа MARYLIE, предназначенная для проектирования и моделирования ускорителей заряженных частиц.

В Европейской организации по ядерным исследованиям (CERN) была разработана программа MAD, предназначенная для моделирования динамики частиц в магнитной оптике. Данная программа использует подмодуль TRANSPORT, написанный К. Брауном для построения отображения до второго порядка нелинейности, и собственную реализацию для четвёртого порядка нелинейности. Решение также основано на построении отображения Ли в виде разложения в ряд Тейлора.

В данных программах не реализовано моделирование спиновой динамики, что делает невозможным их использование в исследовании. Единственной на сегодняшний день программой общего пользования для численного расчета спин-орбитального взаимодействия является COSY Infinity [5], разработанная профессором Мартином Берцем в Мичиганском университете. Решение системы ОДУ строится в виде тензорного представления разложения в ряд Тейлора до заданного порядка нелинейности. Программа также предоставляет MPI-интерфейс для запуска ее на кластерных вычислительных системах.

Реализация COSY Infinity основывается на методах дифференциальной алгебры, что позволяет решать систему дифференциальных уравнений до высоких порядков нелинейности. В целом данная программа и методы, заложенные в ее работу, удовлетворяют требованиям, предъявляемым к производительности, однако имеют ряд недостатков. Так, используемые математические модели нигде не описаны, и в логике работы программы применяется ряд спорных с физической точки зрения подходов (например, перестроение опорной кривой). Это делает затруднительным ее использование для задач учета систематических ошибок и реализации нестандартных управляющих полей.

Отметим, что на начальном этапе исследований программа COSY Infinity может быть успешно применена для численных расчетов. В частности, в настоящее время с использованием этой программы моделируется длительная эволюция динамики пучка частиц на вычислительных ресурсах научно-исследовательского центра Юлих в рамках исследований, проводимых в международной коллаборации JEDI.

Указанные ограничения на существующие программы и методы делают очевидной необходимость разработки нового высокопроизводительного метода интегрирования систем ОДУ, дальнейшая реализация которого подразумевает использование современных суперкомпьютерных технологий. В данной работе описана численная реализация матричного формализма. Теоретические основы этой идеологии развиваются профессором С.Н. Андриановым (СПбГУ) и основаны на применении теории групп Ли [6]. Каждый коэффициент отображения в данном случае представляет собой аналитическую формулу и описывает общее решение системы ОДУ. Численная реализация упрощает и унифицирует процесс построения отображения. Достоинством такого подхода является универсальность метода, который может быть построен на основе любого пошагового алгоритма интегрирования. К недостаткам можно отнести значительный рост времени вычисления коэффициентов отображения при повышении порядка нелинейности. Однако следует иметь в виду, что отображение для системы уравнений строится один раз, после чего оно в неизменном виде применяется для вычисления динамики целого множества начальных состояний динамической системы.

Спин-орбитальная динамика заряженных частиц описывается системой ОДУ, состоящей из уравнений Ньютона – Лоренца и Т-БМТ-уравнения [7]. Далее будем рассматривать задачу построения отображения для решения системы ОДУ в общем виде.

2. Постановка задачи

В настоящее время все численные подходы построения отображения, так или иначе, основываются на разложении нелинейных правых частей дифференциального уравнения и искомого решения в ряд Тейлора до заданного порядка нелинейности. Систему нелинейных ОДУ

$$\frac{d\vec{X}}{dt} = \vec{F}(t, \vec{X})$$

и ее решение при определенных предположениях [6] можно представить в виде

$$\frac{d\vec{X}}{dt} = \sum_{k=0}^{n} P_k(t)\vec{X}^{[k]}, \tag{1}$$

$$\vec{X} = \sum_{k=0}^{n} R_k(t)\vec{X}_0^{[k]}, \tag{2}$$

где n – заданный порядок нелинейности; $P_k(t)$, $R_k(t)$, – матрицы коэффициентов разложения правых частей уравнения и решения в ряд Тейлора. Под оператором $(\cdot)^{[k]}$ понимается кронекеровская степень с редуцированием размерности [6].

Формула (2) задает решение задачи Коши системы (1) для произвольного вектора начального состояния X_0 на интервале интегрирования [0; t]. При вычисленных коэффициентах разложения (2) вычисление нового вектора состояния по начальному производится лишь с использованием операций умножения и сложения числовых двумерных матриц и векторов. В отличие от пошаговых методов интегрирования в данном случае отображение строится сразу для всей системы, оно одинаковое для разных начальных состояний. Кроме того, нет необходимости разбиения интервала интегрирования на шаги, а точность метода определяется лишь порядком нелинейности n и точностью вычисления матриц $R_k(t)$.

3. Матричное интегрирование системы ОДУ

Для вычисления матриц $R_k(t)$ будем использовать следующий подход. Дифференцируя уравнение (2) по времени

$$\frac{d\vec{X}}{dt} = \sum_{k=0}^{n} \frac{d}{dt} R_k(t)\vec{X}_0^{[k]}$$

и сравнивая полученное соотношение с (1), имеем

$$\begin{aligned} \frac{dR_0(t)}{dt} &= \sum_{k=0}^{n} P_k(t)\left(R_k(t)\right)^{[k]}, \\ \frac{dR_j(t)}{dt} &= \sum_{k=0}^{n} P_k(t)\frac{\partial\left(\vec{X}(t)\right)^{[k]}}{\partial\left(\vec{X}_0^{[j]}\right)^{T}}, \end{aligned} \tag{3}$$

где j=1,…,n; вектор $X(t)$ задается равенством (2); под $(\cdot)^T$ понимается операция транспонирования.

Система уравнений (3) описывает динамику отображения на заданном интервале интегрирования и может быть записана в виде $dR/dt = \widetilde{F}(t, R)$. Вычисление коэффициентов матриц $R_k(t)$ может вестись любым подходящим пошаговым методом интегрирования. Общий алгоритм решения системы ОДУ может быть записан в виде следующей последовательности действий.

1. Разложение нелинейных правых частей системы уравнений (1) в ряд (2).
2. Построение вспомогательной системы ОДУ (3) для коэффициентов отображения.
3. Решение задачи Коши: вычисление вектора состояния X, соответствующего начальному состоянию системы X_0.

4. Реализация матричных алгоритмов

Матричный подход интегрирования систем ОДУ предоставляет существенный выигрыш по производительности в сравнении с классическими пошаговыми алгоритмами. На рис. 1 представлено сравнение данного подхода со схемой Рунге–Кутта 4-го порядка точности. Матрицы нелинейного отображения строились также до 4-го порядка нелинейности. Здесь следует отметить, что если не учитывать время, затраченное на построение отображения (решение системы уравнений (3)), время, затраченное на вычисление конечного решения при использовании отображений, не зависит от величины интервала интегрирования. В случае использования пошагового интегрирования время растёт пропорционально.

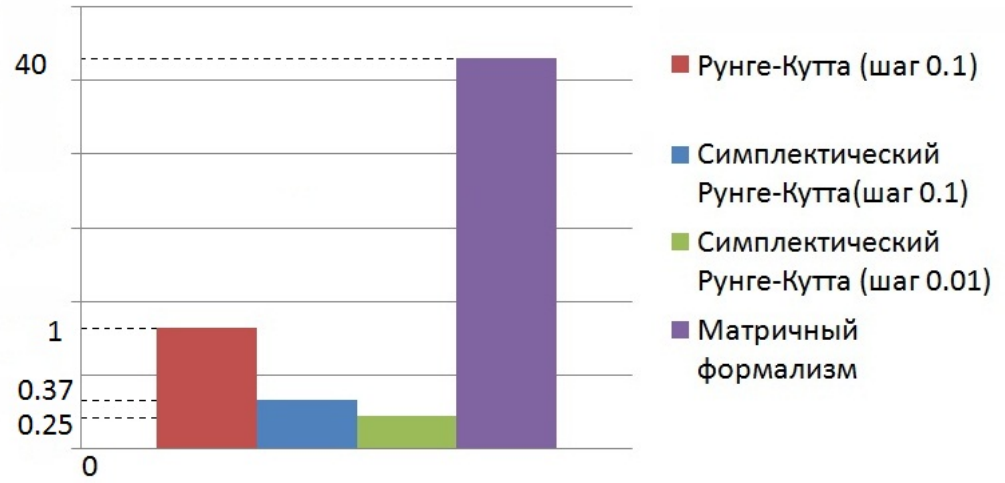


Рис. 1. Относительная производительность численных методов

На втором месте после длительности интервала интегрирования в изучении динамики стоит проблема большого количества частиц в ансамбле пучка. При использовании пошагового интегрирования каждую частицу приходится «просчитывать» отдельно. Использование матричного подхода существенно упрощает задачу, при этом формула (2) для множества частиц преобразуется в

$$\vec{X} = \sum_{k=0}^{n} R_k(t)\left(\vec{X}_0^1 \quad \vec{X}_0^1 \quad \dots \quad \vec{X}_0^N\right)^{[k]},$$

где N – число частиц в пучке. Все операции в этом случае выполняются в виде перемножения числовых матриц. Размерность матриц $R_k(t)$ зависит от вектора состояния и равняется 9 для спин-орбитальной динамики, а также от порядка нелинейности n. Размерность матрицы, описывающей начальное состояние пучка, зависит от количества частиц.

Ввиду того, что предлагаемый метод использует в конечном итоге только линейные операции над числовыми матрицами больших размерностей, наилучшим решением для его реализации является использование архитектуры GPU. Построенный численный метод, подчиняясь парадигме естественного распараллеливания, полностью отображается в конвейерную архитектуру таких вычислительных систем.

Развитие описанного подхода ведется в двух направлениях. Во-первых, совершенствуются алгоритмы, специфические для предметной области [8, 9]. Во-вторых, исследуется и оптимизируется реализация метода на конкретных вычислительных системах.

5. Заключение

В настоящее время численные расчеты по моделированию длительной эволюции заряженных частиц в рамках проекта JEDI ведутся в основном в суперкомпьютерном

центре JUROPA, расположенном в научно-исследовательском центре Юлих. В основном используется MPI-интерфейс программы COSY Infinity. К недостаткам этой программы можно отнести неудачный выбор формализма при реализации вычислительных алгоритмов. В COSY Infinity используются тензорные операции, что делает ее распараллеливание затруднительным даже на традиционных кластерных системах. Матричный формализм, описанный в данной работе, записывается в конечном итоге в виде операторов линейной алгебры и может быть реализован на любой архитектуре высокопроизводительных вычислений.

Для целей тестирования и верификации алгоритма и методов в настоящее время используется гибридный кластер Санкт-Петербургского государственного университета на базе процессоров NVIDIA Tesla. Полномасштабные расчеты планируется проводить в вычислительном центре JuDGE, который также построен на основе гибридной GPU-архитектуры.

А

втор выражает благодарность профессору Ю.В. Сеничеву за постановку задач в области ускорительной физики и обсуждение результатов численного моделирования, Д.В. Зюзину за помощь в проведении сравнительных расчетов на программе COSY Infinity, своему научному руководителю профессору С.Н. Андрианову, а также профессору Мартину Берцу за плодотворные дискуссии в области методов решения дифференциальных уравнений на основе построения отображений.

Литература

1. Fukuyama T. Searching for New Physics beyond the Standard Model in Electric Dipole Moment. – [pdf] (http://arxiv.org/pdf/1201.4252v7.pdf), 2013.
2. Senichev Yu. Storage Ring EDM Simulation: Methods and Results // Proc. of ICAP2012, Rostock, Germany, 2012. – P. 99–103.
3. Oevel W., Sofroniou M. Symplectic Runge-Kutta Schemes II: Classification of Symmetric Methods. – [pdf] (http://citeseerx.ist.psu.edu/viewdoc/download?doi= 10.1.1.46.5060 & rep=rep1&type=pdf).
4. Dragt A., Douglas D. Particle tracking using Lie algebraic methods // Computing in Accelerator Design and Operation, 1984. – P. 122–127.
5. COSY Infinity (http://www.bt.pa.msu.edu/index_cosy.htm).
6. Андрианов С.Н. Динамическое моделирование систем управления пучками частиц. – СПб.: Изд-во С.-Петерб. ун-та, 2002. – 376 с.
7. Balandin V., Golubeva N. Hamiltonian methods for the study of polarized proton beam dynamics in accelerators and storage rings. – [pdf] (arxiv.org/pdf/physics/9903032).
8. Ivanov A., Andrianov S., Kulabukhova N., Maier R., Senichev Yu., Zuyzin D. Testing of Symplectic Integrator of Spin-orbit motion based on Matrix Formalism // Proc. of IPAC2013, Shanghai, China, 2013. – P. 2582–2584.
9. Ivanov A., Andrianov S. Matrix Formalism for Long-term Evolution of Charged Particle and Spin Dynamics in Electrostatic Fields // Proc. of ICAP2012, Rostock, Germany, 2012. – P. 187–189.

ИДЕНТИФИКАЦИЯ ДИНАМИЧЕСКИХ СИСТЕМ НА ОСНОВЕ НЕЛИНЕЙНОГО МАТРИЧНОГО ПРЕОБРАЗОВАНИЯ ЛИ

А.Н. Иванов, П.М. Кузнецов

Санкт-Петербургский госуниверситет

Предлагается метод идентификации нелинейных динамических объектов, основанный на построении матричного отображения. Особое внимание уделяется производительности подхода. Конечный вычислительный алгоритм записывается в виде операций сложения и умножения над числовыми матрицами. Приводится общая классификация подходов к решению систем обыкновенных дифференциальных уравнений, на основе которых строятся алгоритмы идентификации. Также приведен алгоритм матричного интегрирования систем обыкновенных дифференциальных уравнений. Предлагаемый подход заметно выигрывает по производительности у традиционных методов при реализации его на высокопроизводительных вы-числительных системах.

Введение

Под динамической системой будем понимать объект, изменяющийся во времени, который можно описать системой обыкновенных дифференциальных уравнений (ОДУ)

$$\frac{dX}{dt} = F(t, X), \qquad (1)$$

где X – вектор состояния системы (набор параметров, описывающих объект), F – векторная функция, описывающая динамику изменения вектора состояния, удовлетворяющая условиям существования и единственности решения задачи Коши. При заданном начальном состоянии системы $X_0 = X(t_0)$, уравнение (1) полностью определяет динамику изменения вектора X. Причем в новый момент времени t_1 состояние системы определяется однозначно.

Традиционным способом решения уравнения (1) является использование пошаговых методов интегрирования [1]. Например, простейший метод Эйлера выглядит как

$t_0 \qquad X_0,$

$t_1 \qquad X_1 = F(t, X_0) \cdot (t_1 - t_0),$

... ...

$t_N \qquad X_N = F(t, X_{N-1}) \cdot (t_N - t_{N-1}).$

Идея пошаговых методов интегрирования заключается в том, что в каждый новый момент времени вектор состояния системы вычисляется по строго определенной формуле (некоторая аппроксимация истинного решения) в зависимости от предыдущих состояний системы. При изменении начального состояния процесс вычисления повторяется заново. *Под идентификацией в этом случае понимается алгоритм поиска такой функции F, что задаваемая ею система дифференциальных уравнений будет переводить заданное множество начальных состояний в заданное множество конечных состояний через заданный интервал времени (см. рисунок 1).*

$$\begin{Bmatrix} X_0^1 \\ X_0^2 \\ \cdots \\ X_0^K \end{Bmatrix} \rightarrow \begin{Bmatrix} X_N^1 \\ X_N^2 \\ \cdots \\ X_N^K \end{Bmatrix}$$

Рис. 1. Пошаговое интегрирование

1. Построение отображения решения для системы ОДУ

К преимуществам подходов решения систем ОДУ, построенных на основе методов пошагового интегрирования, следует отнести возможность построения высокоточных схем высокого порядка [2]. К недостаткам следует отнести рост глобальной ошибки на длительных интервалах моделирования и низкую вычислительную производительность. Существует иная идеология решения систем ОДУ, основанная на построении отображения. При удовлетворении системой (1) условий теоремы Пикара решение определяется однозначно, и, следовательно, существует некоторое отображение, позволяющее вычислить новое состояние системы в заданный момент времени в зависимости от начального состояния и времени

$$X = R(t, X_0, t_0). \tag{2}$$

Представим, что функция R нам известна или, по крайней мере, известна некоторая ее аппроксимация. Рассмотрим теперь задачу перевода множества начальных состояний системы в начальный момент времени в множество конечных состояний за заданный временной интервал (рис. 2). Формулу (2) в этом случае можно записать в общем виде

$$X_N = R \circ X_0. \tag{3}$$

Отметим, что оператор R переводит теперь любой элемент из множества начальных состояний в элемент из множества конечных состояний за заданный интервал времени. Оператор R однозначно определяется самой системой. Чтобы вычислить новое состояние через заданный интервал времени, нужно применить оператор к начальному состоянию системы. При использовании пошаговых методов пришлось бы вычислять промежуточные состояния в моменты времени X_1, X_2,…, $X_{N\text{-}1}$. При использовании отображения новое состояние вычисляется за одну итерацию.

$$\left\{\begin{matrix} X_0^1 \\ X_0^2 \\ \cdots \\ X_0^K \end{matrix}\right\} \xrightarrow{M} \left\{\begin{matrix} X_N^1 \\ X_N^2 \\ \cdots \\ X_N^K \end{matrix}\right\}$$

Рис. 2. Построение отображения

Под идентификацией динамической системы в этом случае понимается алгоритм поиска такого отображения R, которое будет переводить заданное множество начальных состояний в заданное множество конечных состояний через заданный интервал времени (см. рис. 2).

Известной сложностью является поиск самого отображения. Софус Ли, основатель теории непрерывных групп преобразований [3], показал, что в общем случае алгоритм построения отображения по сложности соответствует пошаговому интегрированию. Однако существует ряд подходов, применяемых для оценки этого отображения. К ним следует отнести построение отображения на основе непрерывных преобразований групп Ли [4], использование дифференциальной алгебры [5] для вычисления производных и применение матричного формализма [6].

В данной работе приведен алгоритм численной реализации матричного интегрирования, позволяющего оценивать эволюцию (динамику) отображения R в заданном интервале времени. Преимуществом матричного подхода является линейность всех операций в конечном виде алгоритма. Нелинейная динамика системы описывается при помощи линейных операций сложения и перемножения числовых матриц.

Понятие идентификации системы, данное во введении, можно соотнести с параметрической идентификацией, используемой в классической теории управления. Понятие идентификации на основе отображения возникает, например, в случаях оценки локального преобразования, которому подвергается некоторый объект и которое переводит его из одного состояния в другое.

В случае численного матричного интегрирования систем ОДУ под идентификацией следует понимать комбинацию двух методов. При этом целью подбора параметров дифференциального уравнения (1) является соответствие пошаговой динамике всего отображения (3).

1. Построение матричного отображения

Предположим, что систему дифференциальных уравнений (1) и ее общее решение (2) можно разложить в ряд Тейлора

$$\frac{dX}{dt} = \sum_{k=0}^{n} P_k(t) X^{[k]}, \tag{4}$$

$$X = \sum_{k=0}^{n} R_k(t) X_0^{[k]}, \tag{5}$$

где n – заданный порядок нелинейности; $P_k(t)$, $R_k(t)$, – матрицы коэффициентов разложения правых частей уравнения и решения в ряд Тейлора. Под оператором $(\cdot)^{[k]}$ понимается кронекеровская степень с редуцированием размерности [6].

Уравнение (4) описывает динамическую систему. Матрицы P_k задаются коэффициентами, варьирование которых изменяет динамику системы. Матрицы R_k задают отображение (3) в матричном виде и зависят от времени t.

Пример

Пусть имеется динамическая система, заданная в соответствии с формулой (4)

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} x^2 \\ xy \\ y^2 \end{pmatrix},$$

тогда ее решением в виде матричного отображения будет являться соотношение

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\sqrt{2}\cdot t) & \sin(\sqrt{2}\cdot t)/\sqrt{2} \\ -\sqrt{2}\sin(\sqrt{2}\cdot t) & \cos(\sqrt{2}\cdot t) \end{pmatrix}\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix}\begin{pmatrix} {x_0}^2 \\ x_0 y_0 \\ {y_0}^2 \end{pmatrix},$$

где $(x_0; y_0)$ – вектор состояния в начальный момент времени t_0=0.

Здесь матричное отображение приведено в аналитической форме. Коэффициенты второго порядка нелинейности опущены ввиду громоздкости. Данное решение может быть проверено непосредственным дифференцированием и сравнением результатов с изначальной системой ОДУ. Для произвольной системы (4) процесс построения такого отображения может быть затруднительным, поэтому будем искать отображения (5) численно, используя идеологию пошагового интегрирования, но примененную к отображению. Дифференцируя уравнение (5) по времени

$$\frac{d\vec{X}}{dt} = \sum_{k=0}^{n} \frac{d}{dt} R_k(t) \vec{X}_0^{[k]}$$

и сравнивая полученное соотношение с (4) имеем

$$\begin{aligned} \frac{dR_0(t)}{dt} &= \sum_{k=0}^{n} P_k(t)\left(R_k(t)\right)^{[k]}, \\ \frac{dR_j(t)}{dt} &= \sum_{k=0}^{n} P_k(t)\frac{\partial\left(\vec{X}(t)\right)^{[k]}}{\partial\left(\vec{X}_0^{[j]}\right)^T}, \end{aligned} \tag{6}$$

где j=1,…,n; вектор $X(t)$ задается равенством (2); под $(.)^T$ понимается операция транспонирования. Система (6) является системой обыкновенных дифференциальных уравнений относительно коэффициентов отображения R:

$$\frac{dR}{dt} = \widetilde{F}(t, P). \qquad (7)$$

Заметим, что правые части уравнения (7), во-первых, вычисляются в матричном виде, а во-вторых, не зависят от вектора начального состояния X_0. Уравнение (7) полностью задается системой (4) и может быть разрешено некоторым пошаговым методом интегрирования:

$$\begin{aligned} t_0 &\Rightarrow R(t_0), \\ t_1 &\Rightarrow R(t_1), \\ \ldots &\quad \ldots \\ t_N &\Rightarrow R(t_N). \end{aligned}$$

Алгоритм построения решения матричной динамической системы через заданные интервалы времени запишется в виде следующей последовательности шагов.

1. Задать параметры динамической системы – коэффициенты матриц P_k.
2. Пошагово вычислить эволюцию матричного отображения R:
   a. В начальный момент времени отображение тождественное (переводит начальное множество состояний само в себя).
   b. В каждый момент времени t_i отображение $R(t_i)$ вычисляется по заданной формуле на основе значений матриц P_k и $R(t_{i-1})$.

Немаловажным является тот факт, что можно применять оператор отображения (3) сразу ко множеству начальных состояний системы

$$\left\{X_N^1; X_N^2; \ldots; X_N^K\right\} = R \circ \left\{X_0^1; X_0^2; \ldots; X_0^K\right\},$$

причем в случае матричного отображения это соотношение запишется в виде

$$\mathbf{X}_\mathrm{N} = \mathrm{R} \circ \mathbf{X}_0,$$

где каждый столбец матриц $\mathbf{X}_\mathrm{N}$ и $\mathbf{X}_0$ соответствует вектору состояния. Алгоритм идентификации может быть представлен в виде блок-схемы, изображенной на рис. 3.

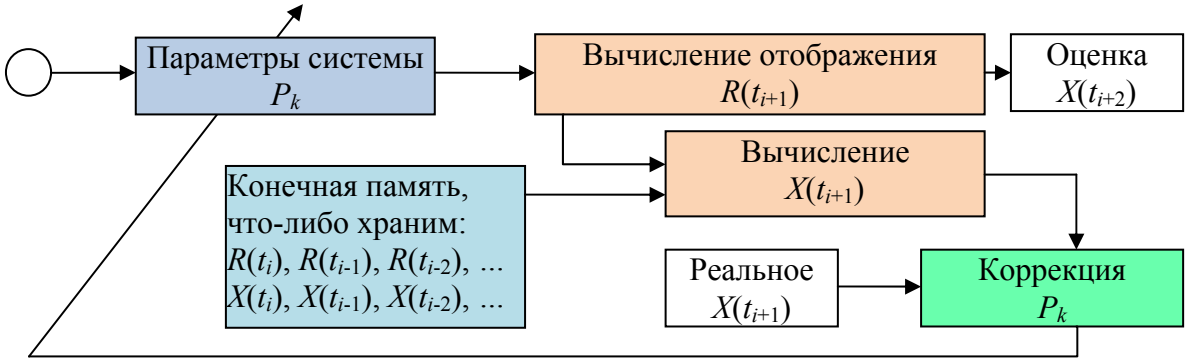


Рис. 3. Алгоритм идентификации

В данном случае отслеживается динамика состояний. На основе поступающих данных (новых состояний) корректируется динамическая система (коэффициенты матриц P_k), которая порождает новую последовательность отображений. Уравнение (4) в силу своей нелинейности может описывать нелинейную динамику (например, включать в себя движение системы координат). Производительность метода при этом зависит целиком от порядка нелинейности и размерности задачи. При этом все вычисления представлены в виде операций над числовыми матрицами.

Заключение

В работе представлен метод построения систем идентификации динамических объектов на основе построения матричного нелинейного отображения. Рассмотренный подход обладает рядом преимуществ по сравнению с пошаговыми алгоритмами интегрирования, а именно позволяет строить решение системы ОДУ сразу в виде отображения. Матричный подход успешно применяется при моделировании сложных систем управления пучками частиц [6] на длительных временных интервалах. Дальнейшее развитие метода авторами видится в расширении сферы его применения. Так, например, особый интерес представляет применение подхода в области компьютерного зрения, где массовый параллелизм задач может удачным образом отразиться на матричной записи алгоритма идентификации. Отметим, что теория групп Ли уже используется для алгоритмов оценки движения на изображениях (см., напр., [7]). Однако в настоящее время разработанные подходы не оценивают динамику системы и основываются на способах оценки локального отображения.

Литература

1. Hairer E., Lubich C., Wanner G. Geometric numerical Integration. – Netherlands: Springer-Verlag, 2006. – 644 p.
2. Oevel W., Sofroniou M. Symplectic Runge-Kutta schemes II: classi_cation of symmetric methods. – [pdf] (http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.46.5060).
3. Полищук Е.М. Софус Ли. – Л.: Наука, 1983. – 212 с.
4. Dragt A.J., Douglas D.R. Particle tracking using Lie algebraic methods // Computing in Accelerator Design and Operation. 1984. – V. 215. P. 122–127.
5. Hubbard J., Lundell B. A first look at differential algebra. – [pdf] (http://www.math.washington.edu/~bel05/Research/Hubbard_Lundell-A_First_Look_at_Differential_Algebra.pdf).
6. Андрианов С.Н. Динамическое моделирование систем управления пучками частиц. – СПб.: Изд-во С.-Петерб. ун-та, 2004. – 376 с.
7. Xu Q., Ma D. Applications of Lie Groups and Lie Algebra to Computer Vision: A Brief Survey // International Conference on Systems and Informatics, 2012. – P. 2024–2029.

МАССИВНО-ПАРАЛЛЕЛЬНЫЕ ПРЕДОБУСЛОВЛЕННЫЕ ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ С РАЗРЕЖЕННЫМИ МАТРИЦАМИ БОЛЬШОГО РАЗМЕРА

И.Е. Капорин[1], О.Ю. Милюкова[2], Ю.Г. Бартенев[3]

[1]Вычислительный центр им. А.А.Дородницына РАН, Москва
[2]Институт прикладной математики им. М.В.Келдыша РАН, Москва
[3]Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Для больших разреженных систем линейных алгебраических уравнений (порядка до десятков миллионов), возникающих, в частности при дискретизации краевых задач для эллиптических уравнений в частных производных, описаны массивно-параллельные алгоритмы приближенного решения. Продемонстрирована возможность их эффективной параллельной реализации на многопроцессорных вычислительных системах с распределенной памятью при числе процессов MPI до 1500.

Введение

Одной из наиболее часто встречающихся трудоемких стандартных вычислительных задач является решение невырожденных систем линейных алгебраических уравнений (СЛАУ) $Ax = b$ с разреженной $n \times n$-матрицей A большого размера n. При этом возможность экономии памяти дает итерационным методам серьезное преимущество по сравнению с методами точной разреженной треугольной факторизации, если требуется решать задачи большого размера. Кроме того, итерационные методы обладают более гибкой структурой вычислений, что облегчает их параллельную реализацию. Однако для многих важных классов задач известные массивно-параллельные итерационные методы (например, методы проектирования на крыловские подпространства с диагональным или блочно-диагональным предобусловливанием) обладают недостаточной надежностью и эффективностью.

Таким образом, решение разреженных СЛАУ на высокопроизводительных многопроцессорных системах все еще является открытой проблемой вычислительной математики и требует дальнейших исследований. Вместе с тем, достаточно широко распространилось мнение, что в настоящее время следует разрабатывать только такие методы решения прикладных задач большого размера, которые вообще не приводят к необходимости решения разреженных СЛАУ (т.н. «matrix-free methods»). Представляется, однако, что дальнейшая разработка массивно-параллельных решателей СЛАУ достаточно перспективна ввиду наличия больших теоретических и практических заделов (как в области построения решателей, так и в способах их эффективного использования), а также ясных перспектив их развития, связанных, например, с оптимизацией используемых предобусловливателей. Ниже описано достаточно общее, эффективное и надежное предобусловливание [1-3, 8], пригодное для ускорения метода сопряженных градиентов и других итерационных методов, использующих проекцию на крыловские подпространства (BiCGStab, GMRes). Для задач, возникающих при сеточной дискретизации пространственных объектов или их поверхностей, используемый предобусловливатель можно рассматривать как особый вариант метода разделения на подобласти с налеганием. Также приводятся результаты тестирования этих решателей на реальных задачах.

1. Предобусловливание посредством оптимального согласования приближенных треугольных разложений перекрывающихся подматриц

Алгоритмической основой реализованных решателей является предобусловливание, использующее специальный метод согласования приближенных треугольных факторизаций перекрывающихся главных подматриц исходной матрицы коэффициентов [3]. Количество используемых главных подматриц принимается равным доступному числу процессов MPI, что обеспечивает хорошую параллельную масштабируемость.

Применяются итерационные методы проекций на крыловские подпространства

$$x_i - x_0 \in K_i(Hr_0; HA) = \mathrm{span}(Hr_0, HAHr_0, ..., (HA)^{i-1} Hr_0),$$

где H – матрица-предобусловливатель. Таким образом, невязка i-го приближения к решению имеет вид $r_i = b - Ax_i = \pi_i(AH)r_0$, где $\pi_i(t)$ – подходящий многочлен, удовлетворяющий $\pi_i(0) = 1$ (так, $\pi_i(\cdot)$ в методе GMRes выбирается из условия минимизации евклидовой нормы невязки). Ключевым моментом является то, что если предобусловленная матрица AH достаточно близка к единичной матрице I, хотя бы «в среднем», то степень многочлена i (а тогда и число итераций метода), обеспечивающая достаточную малость нормы невязки, оказывается не слишком большой (см., напр., [3]). Следовательно, необходима разработка технологии построения матриц-предобусловливателей H, обеспечивающих достаточно хорошее приближение $I \approx AH$, и в то же время представимых в виде, допускающем построение эффективного массивно-параллельного алгоритма, реализующего операцию $z = Hy$, где y - произвольный n-вектор. Поставленная задача весьма нетривиальна ввиду противоречивости указанных требований к H.

Опишем теперь конструкцию предобусловливателя, предназначенного для применения к матрицам с достаточно «сильной» диагональю (например, к положительно определенным матрицам), который лишь несущественно сложнее «блочного метода Якоби», однако способен обеспечивать существенно более быструю сходимость итераций.

Пусть матрица A переупорядочена и разбита на блоки, причем на блочной диагонали расположены p квадратных блоков размера n_s, $1 \le s \le p$. Обозначим $k_s = n_1 + ... + n_s$, и пусть $m_s \ge n_s$ – размеры расширенных блоков. Определим прямоугольные матрицы

$$V_s = [e_{j_s(1)} | ... | e_{j_s(m_s - n_s)} | e_{k_{s-1}+1} | ... | e_{k_s}],$$

столбцы которых являются единичными n-векторами, где $k_{s-1}+1, ..., k_s$ представляют собой индексы s-го блока, а $j_s(1), ..., j_s(m_s - n_s)$ являются индексами перекрытия, причем $j_s(.) \le k_{s-1}$. Для построения индексов перекрытия s-го блока используется множество столбцовых позиций соответствующего блока строк матрицы A^q, где $q>0$ (заметим, что при $q=0$ перекрытие пусто, и получается т.наз. блочный метод Якоби). Применим теперь к каждой расширенной $m_s \times m_s$-подматрице $V_s^T A V_s$ приближенное треугольное разложение 2-го порядка «по значению» ILU2(τ) (the 2nd order LU-factorization), см. [4]:

$$V_s^T A V_s = L_s U_s + L_s R_s + K_s U_s - S_s, \ \|R_s\| = O(\tau), \ \|K_s\| = O(\tau), \ \|S_s\| = O(\tau^2),$$

где R_s и K_s – основные матрицы погрешностей (строго треугольные, верхняя и нижняя соответственно), S_s – остаточная матрица погрешностей, $0 < \tau \le 1$ – порог отсечения, $s = 1, ..., p$. Предлагаемый предобусловливатель, получивший обозначение BIILU(p;q)-ILU2(τ) (от Block Incomplete Inverse LU), запишем в следующем виде [1, 2]:

$$H = \sum_{s=1}^{p} V_s U_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} L_s^{-1} V_s^T .$$

Строгое обоснование оптимальности такого метода предобусловливания (при заданной заранее структуре перекрытий) имеется в случае произвольной симметричной положительно определенной матрицы A (когда $L_s = U_s^T$, а для отыскания U_s применяется IC2(τ)-разложение), см. [1, 3]. Отметим, что для решения СЛАУ с несимметричной матрицей, не обладающей «сильной» диагональю, можно использовать другое представление для H , см. [1, 3], а к подматрицам типа $V_s^T A V_s$ применять более общее разложение ML-ILU(τ) [6, 7].

Реализация на FORTRAN/MPI соответствующих матрично-векторных операций аналогична описаннной в [2, 5]. В параллельных алгоритмах, реализующих предобусловленные методы BiCGStab и GMRes, для умножения матрицы коэффициентов на вектор используется ее распределенное представление, отвечающее специальному аддитивному разложению

$$A = \sum_{s=1}^{p} \widetilde{V}_s [A_s \mid C_s] \widetilde{W}_s^T , \ \widetilde{V}_s = [e_{k_{s-1}+1} | \ldots | e_{k_s}] , \ \widetilde{W}_s = [e_{k_{s-1}+1} | \ldots | e_{k_s} | e_{j'_s(1)} | \ldots | e_{j'_s(m'_s - n_s)}] ,$$

где $n_s \times n_s$-матрица A_s является s-м диагональным блоком матрицы A, а C_s – $n_s \times (m'_s - n_s)$-матрица, содержащая ненулевые столбцы подматрицы, расположенные слева и справа от A_s, индексы $j'(\cdot)$ представляют собой номера соответствующих столбцовых позиций в A. Используются p процессов MPI (по числу слагаемых аддитивных представлений A и H), причем s-й процесс выполняет вычисления, отвечающие умножению на вектор s-го слагаемого матрицы A или предобусловливателя H. Для реализации межпроцессорных обменов разработаны надежные и эффективные алгоритмы на базе операций MPI_Send и MPI_Recv.

2. Программная реализация решателей СЛАУ и их тестирование

На основе описанного выше подхода был разработан комплекс программных модулей (библиотека VC_RAN_SLAU), предназначенный для приближенного решения больших разреженных систем линейных алгебраических уравнений с симметричными положительно-определёнными и несимметричными матрицами. Созданный программный код, написанный на языке FORTRAN-90 с использованием интерфейса MPI, удовлетворяет достаточно высоким требованиям надежности и эффективности функционирования, отличаясь в то же время небольшим объемом (менее 200 Kbytes текста) и полной замкнутостью. Библиотека создавалась для пакетов программ инженерного анализа, разработанных в РФЯЦ-ВНИИЭФ.

Расчеты задач и замеры времени проводились на вычислительной системе ЭВМ ВЦКП ВНИИЭФ. Вычислительные узлы связаны между собой быстрой коммуникационной сетью. В каждом узле многопроцессорной системы (возможно, за исключением последнего узла) во всех случаях использовались все вычислительные ядра.

Результаты тестирования на несимметричных СЛАУ, возникающих при моделировании задач аэродинамики пакетом ЛОГОС Аэрогидромеханика [9], а также на СЛАУ с симметричными положительно определенными матрицами, возникающих при расчете напряженно-деформированного состояния конструкций пакетом ЛОГОС Прочность [10] и при расчете задач подземной фильтрации пакетом НИМФА [11] (см. табл.1), приведены на рис.1, 2.

Таблица 1. Структурные характеристики тестовых матриц

Матрица *n* *nz*(*A*) тип Источник
C20_A_0420_CFL_50 3 655 200 126 831 200 5x5 несимм. ЛОГОС-Аэро
ONERA_M6_A_0386_CFL_100 9 494 800 329 975 600 5x5 несимм. ЛОГОС-Аэро
Duck_A_0486_CFL_100 24 473 160 851 654 700 5x5 несимм. ЛОГОС-Аэро
49_750 2 615 169 190 101 409 симм.пол.опр. ЛОГОС-Проч
p4_6 4 216 212 228 406 070 симм.пол.опр. ЛОГОС-Проч
18mln 18 063 492 1 335 851 712 симм.пол.опр. ЛОГОС-Проч
Balt12 12 140 928 314 737 280 симм.пол.опр. НИМФА
32mln 32 022 528 815 096 620 симм.пол.опр. НИМФА

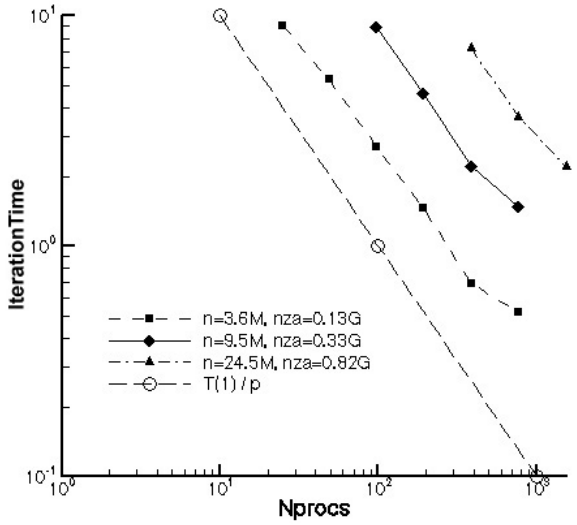


Рис. 1. Зависимости времени счета от числа процессов MPI для задач с несимметричными матрицами

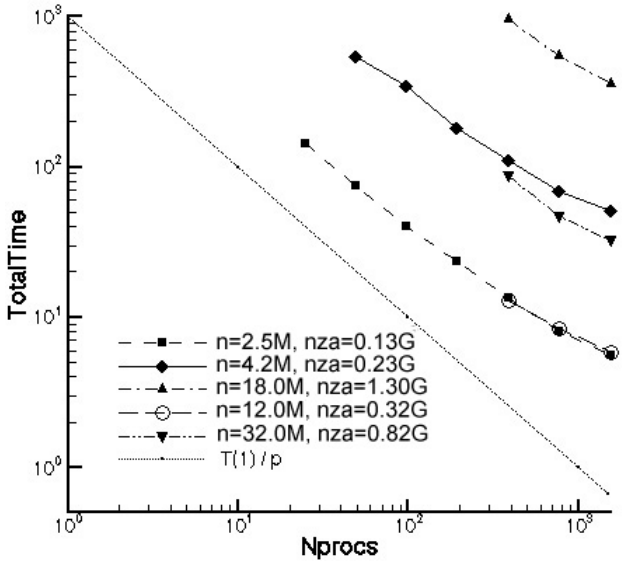


Рис. 2. Зависимость времени счета от числа процессов MPI для задач с симметричными положительно определенными матрицами

3. Выводы и направления развития

Использование предлагаемого массивно-параллельного предобусловливания для ускорения итераций методов сопряженных градиентов или BiCGStab оказалось эффективным для решения СЛАУ размера до $n = 3.2 \cdot 10^7$ и числом ненулевых элементов до

$nz = 1.3 \cdot 10^9$, возникающих в расчетах по пакетам ЛОГОС и НИМФА. При этом наблюдалась достаточно хорошая параллельная масштабируемость при использовании до 1500 процессов MPI. Производительность построенных алгоритмов предобусловливания может быть улучшена за счет более сбалансированного разбиения матрицы на перекрывающиеся блоки, а также на пути разработки способов предварительной подготовки матрицы СЛАУ, например, посредством двустороннего мелкоблочно-диагонального масштабирования. Дальнейшее повышение эффективности решателей (особенно при использовании итераций GMRes) может быть достигнуто за счет применения адаптивного полиномиального предобусловливания.

Авторы выражают благодарность В.А. Ерзунову, А.Н. Стаканову и Е.Б. Щаниковой за помощь в разработке, тестировании параллельных алгоритмов решения СЛАУ большого размера и за интеграцию VC_RAN_SLAU в комплекс библиотек параллельных решателей СЛАУ LParSol [12] ВНИИЭФ для «промышленного» применения.

Работа выполнялась по договору между ВЦ РАН и РФЯЦ-ВНИИЭФ, а также поддерживалась Целевыми программами П-15 и П-18 президиума РАН, грантом РФФИ 11-01-00786, грантом НШ-5264.2012.1 и программой ОМН №3 РАН.

Литература

1. Kaporin I.E., Konshin I.N. A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems // Numer. Linear Algebra Appls. 2002. V. 9, P. 141-162.
2. Капорин И.Е., Милюкова О.Ю. Массивно-параллельный алгоритм предобусловленного метода сопряженных градиентов для численного решения систем линейных алгебраических уравнений // Сб. трудов отдела проблем прикладной оптимизации ВЦ РАН (под ред. В.Г. Жадана). М.: Изд-во ВЦ РАН, 2011. С. 32-49.
3. Kaporin I.E. New convergence results and preconditioning strategies for the conju-gate gradient method // Numer. Linear Algebra Appls. 1994. V. 1, N. 2. P. 179-210.
4. Kaporin I.E. High quality preconditioning of a general symmetric positive matrix based on its $U^T U + U^T R + R^T U$ -decomposition // Numer. Linear Algebra Appls. 1998. V. 5. P. 484-509.
5. Милюкова О.Ю. Некоторые параллельные итерационные методы с факторизованными матрицами предобусловливания для решения эллиптических уравне-ний на треугольных сетках // Ж. вычисл. матем. и матем. физики. 2006. Т.46. № 6. С. 1096-1112.
6. Saad Y. Multilevel ILU with reorderings for diagonal dominance // SIAM J. Sci. Comput. 2005. V. 27. P. 1032-1057.
7. Kaporin I. Multilevel ILU preconditionings for general unsymmetric matrices // Proc. Int. Conf. NUMGRID/VORONOI-2008, Moscow, 10-13 June 2008. P.150-157.
8. Капорин И.Е., Милюкова О.Ю. Предобусловливание итерационных методов для эффективного массивно-параллельного решения систем линейных алгебраических уравнений // Труды XIII Международного семинара «Супервычисления и математическое моделирование», 3-7 октября 2011. Саров, 2012. С. 243-252.
9. Козелков А.С., Дерюгин Ю.Н., Зеленский Д.К. и др. Многофункциональный пакет программ ЛОГОС для расчета задач гидродинамики и тепломассапереноса на суперЭВМ // Тезисы докладов на XIV Международном семинаре «Супервычисления и математическое моделирование», 1-5 октября 2012. Саров, 2012. С. 108.

10. Циберев К.В., Авдеев П.А., Артамонов М.В. и др. Пакет программ ЛОГОС. Об-зор текущих возможностей решения задач прочности // Тезисы докладов на XIV Международном семинаре «Супервычисления и математическое моделирование», 1-5 октября 2012. Саров, 2012. – С.159-161.
11. Алейников А.Ю., Бардина М.Н., Горев В.В. и др. Параллельная версия комплек-са программ НИМФА // Тезисы докладов на XIII Международном семинаре «Супервычисления и математическое моделирование», 3-7 октября 2011. Саров, 2011. – С. 26-27.
12. Бартенев Ю.Г., Бондаренко Ю.А., Ерзунов В.А. и др. Комплекс LParSol для решения СЛАУ // Тезисы докладов на XIII Международном семинаре «Супервычисления и математическое моделирование», 3-7 октября 2011. Саров, 2011. С. 34-36.

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ ДЛЯ ТЕСТИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММНЫХ МОДУЛЕЙ

А.Н. Коварцев, Е.Е. Серповская

Самарский государственный аэрокосмический университет им. С.П. Королёва

Рассматривается метод глобального поиска разрывов второго рода функции для поиска ошибок, имеющих «арифметический» характер. Представлена математическая модель поиска бесконечных разрывов функции. Приводится оценка качества предлагаемого алгоритма.

Введение

Основной целью тестирования программных модулей (ПМ) является обнаружение и устранение программных ошибок или (и), если это возможно, доказательство отсутствия последних в программе.

Техника тестирования программных модулей во многом определяется их специализацией. Рассмотрим ПМ, построенные на основе численного анализа и вычислительной математики. При построении таких программ в качестве «неделимых» программных единиц обычно выступают модули, реализующие функциональные преобразования между *n* и *m*-мерными пространствами действительных (комплексных) чисел. К предметным областям с такой схемой построения исходных модулей относятся САПР технических объектов, АСНИ и т.д.

Формально вычислительный модуль можно рассматривать как отображение

$$f_k : X_k^{in} \to X_k^{out}, \qquad (1)$$

действующее из множества входных параметров $X_k^{in} \in \Omega_{f_k}$ модуля на множество вычисляемых данных X_k^{out}.

Ошибочные ситуации в программных модулях возникают по разным причинам [1, 2]. Наиболее трудно обнаруживаемыми следует считать фатальные ошибки, связанные с прерыванием вычислительного процесса, поскольку любая такая ошибка, возникшая даже во второстепенном модуле, может прервать сложный вычислительный эксперимент, привести к зависанию программы или даже потере полученных результатов.

Природа таких ошибок, как правило, имеет «арифметический» характер, среди которых наиболее сложно идентифицируются ошибки типа деление на ноль (ошибки переполнения или потери порядка). Для вычислительных модулей (1) подобные ошибки возникают в точках разрыва второго рода функции и, следовательно, для их поиска можно применять методы глобального оптимизации [4, 5]:

$f(X) \to Min\,(Max), \quad X \in \Omega_f$.

1. Постановка задачи

В работе предлагается метод глобального поиска точек разрывов второго рода функции, идеи которого во многом заимствованы из метода многоэкстремальной оптимизации Р.Г. Стронгина [3].

Пусть областью поиска ошибок в вычислительном модуле является единичный квадрат $\Pi = [0,\ 1] \times [0,\ 1]$, который в процессе деления разбивается на четыре квадрата меньших размеров (см. рис. 1).

В узлах сформированной регулярной сетки вычисляются значения тестируемой функции z_{ij} . В качестве модели тестируемой функции рассмотрим многочлен четвертого порядка (второго порядка по каждой из переменных) вида

$$P_{2,2}(x, y) = a_0 x^2 y^2 + a_1 x^2 y + a_2 x y^2 + a_3 x^2 + a_4 y^2 + a_5 x y + a_6 x + a_7 y + a_8 . \qquad (2)$$

Коэффициенты многочлена (2) подбираются по результатам вычислений функции в узлах интерполяции z_{ij} .

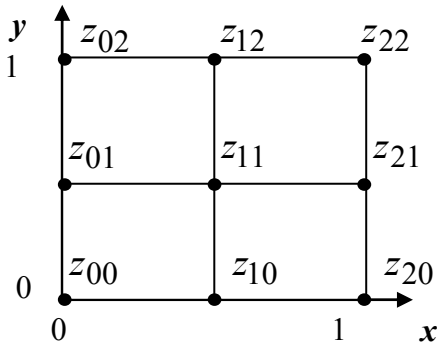


Рис. 1. Единичный квадрат

В соответствии с логикой метода характеризации [3] один из вновь образованных квадратов процедуры деления исходного квадрата подвергается очередному делению на квадраты меньшего размера и т.д. В общем случае происходит неравномерное разбиение области поиска.

2. Математическая модель поиска бесконечных разрывов функции

С практической точки зрения вместо многочлена (2) на регулярных сетках удобнее использовать двумерную первую интерполяционную формулу Ньютона.

Введем двойные разности высших порядков по следующей схеме. Определим частные конечные разности первого порядка

$$\begin{aligned} \Delta^1_x z_{ij} &= z_{i+1\,j} - z_{ij}, \\ \Delta^1_y z_{ij} &= z_{ij+1} - z_{ij}. \end{aligned} \qquad (3)$$

Повторно применяя эти операции, можно получить двойные разности высших порядков

$$\Delta^{n+m} z_{ij} = \Delta^{n+m}_{x^n y^m} z_{ij} = \Delta^n_{x^n} (\Delta^m_{y^m} z_{ij}) . \qquad (4)$$

Значения тестируемой функции в узлах интерполяционной сетки (см. рис. 1) представим матрицей

$$Z^T = \begin{Vmatrix} z_{00} & z_{10} & z_{20} \\ z_{01} & z_{11} & z_{21} \\ z_{02} & z_{12} & z_{22} \end{Vmatrix} . \qquad (5)$$

Для удобства вычислений введем новые переменные

$$q = \frac{x - x_0}{h_x},\ p = \frac{y - y_0}{h_y}. \qquad (6)$$

Полином $P_{2,2}(q, p)$ будем искать в виде

$$P_{2,2}(q, p) = A_0(p) + A_1(p) q + A_2(p) q(q-1) . \qquad (7)$$

При фиксированных значениях второй переменной ($p = 0, 1, 2$) построим одномерные полиномы в форме Ньютона.

Например, при p=0 многочлен может быть представлен следующим образом:

$$P^{(0)}(q) = z_{00} + \Delta_q z_{00} q + \frac{\Delta^2_{qq} z_{00}}{2} q(q-1). \tag{8}$$

Аналогично выводятся формулы для $p = 1$ и $p = 2$.

В формуле (7) коэффициент $A_0(p)$ описывает изменение первого члена многочлена (8) в зависимости от переменной p и может быть представлен в свою очередь многочленом второго порядка. Для его построения построим таблицу конечных разностей и получим:

$$A_0(p) = z_{00} + \Delta_p z_{00} p + \frac{\Delta^2_{pp} z_{00}}{2} p(p-1). \tag{9}$$

Аналогичным образом построим таблицу конечных разностей для коэффициентов $A_1(p)$ и $A_2(p)$. Коэффициенты $A_1(p)$ и $A_2(p)$ могут быть вычислены по формулам

$$A_1(p) = \Delta_q z_{00} + \Delta^{1+1}_{qp} z_{00} p + \frac{\Delta^{1+2}_{qpp} z_{00}}{2} p(p-1), \tag{10}$$

$$A_2(p) = \frac{1}{2}\Delta^2_{qq} z_{00} + \frac{1}{2}\Delta^{2+1}_{qqp} z_{00} p + \frac{\Delta^{2+2}_{qqpp} z_{00}}{4} p(p-1). \tag{11}$$

Введем матричную форму полного двухмерного полинома. Для этого определим матрицу конечных разностей

$$A = \begin{Vmatrix} z_{00} & \Delta_p z_{00} & \frac{1}{2}\Delta^2_p z_{00} \\ \Delta_q z_{00} & \Delta^{1+1}_{qp} z_{00} & \frac{1}{2}\Delta^{1+2}_{qpp} z_{00} \\ \frac{1}{2}\Delta^2_{qq} z_{00} & \frac{1}{2}\Delta^{2+1}_{qqp} z_{00} & \frac{1}{4}\Delta^{2+2}_{qqpp} z_{00} \end{Vmatrix}. \tag{12}$$

Если ввести вектора

$$Q = \begin{Vmatrix} 1 \\ q \\ q(q-1) \end{Vmatrix}, P = \begin{Vmatrix} 1 \\ p \\ p(p-1) \end{Vmatrix}, \tag{13}$$

тогда многочлен (7) с коэффициентами, вычисляемыми по формулам (9), (10) в векторной форме может быть представлен выражением

$$P_{2,2}(q, p) = Q^T AP = (AP, Q). \tag{14}$$

С помощью матричных форм легко подсчитать вторые частные производные для полиномиальной функции (14):

$$P''_{2,2\,qq}(q, p) = \frac{\partial^2 P_{2,2}}{\partial q^2} = \left(Q^T\right)''_{qq} AP = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}^T AP, \tag{15}$$

$$P''_{2,2\,pp}(q, p) = \frac{\partial^2 P_{2,2}}{\partial p^2} = Q^T AP''_{pp} = Q^T A \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}, \tag{16}$$

$$P''_{2,2\,qp}(q,p)=\frac{\partial^2 P_{2,2}}{\partial q\partial p}=\left(Q^T\right)'_q AP''_p=\begin{pmatrix}0\\1\\2q-1\end{pmatrix}^T A\begin{pmatrix}0\\1\\2p-1\end{pmatrix}. \quad (17)$$

Вторые частные производные от полиномиальной функции используются для вычисления характеристической функции, в которой одна из компонентов пропорциональна дифференциалу второго порядка.

С геометрической точки зрения второй дифференциал пропорционален главной части гауссовой кривизны поверхности. В частности в качестве приращений независимых переменных в первом приближении можно взять шаг частичной интерполяционной сетки разбиения очередного квадрата:

$dx=h_x$, $dy=h_y$, тогда

$$d^2P_{2,2}(x,y)=\frac{\partial^2 P}{\partial x^2}h_x^2+2\frac{\partial^2 P}{\partial x\partial y}h_xh_y+\frac{\partial^2 P}{\partial y^2}h_y^2. \quad (18)$$

Откуда

$$R(i)=\left(\left|\frac{\partial^2 P}{\partial x^2}\right|+2\left|\frac{\partial^2 P}{\partial x\partial y}\right|+\left|\frac{\partial^2 P}{\partial y^2}\right|\right)\left(h_xh_y\right)^r, \quad (19)$$

где r – параметр метода тестирования функции.

Если учесть, что

$$\frac{\partial^2 P(x,y)}{\partial x^2}=\frac{1}{h_x^2}\frac{\partial^2 P(q,p)}{\partial q^2}=\frac{1}{h^2}\frac{\partial^2 P(q,p)}{\partial q^2},$$

$$\frac{\partial^2 P(x,y)}{\partial y^2}=\frac{1}{h_y^2}\frac{\partial^2 P(q,p)}{\partial p^2}=\frac{1}{h^2}\frac{\partial^2 P(q,p)}{\partial p^2},$$

$$\frac{\partial^2 P(x,y)}{\partial x\partial y}=\frac{1}{h_xh_y}\frac{\partial^2 P(q,p)}{\partial q\partial p}=\frac{1}{h^2}\frac{\partial^2 P(q,p)}{\partial q\partial p},$$

то (18) приобретает вид

$$d^2P_{2,2}(x,y)=\frac{h^{2r}}{h^2}\left(\frac{\partial^2 P}{\partial x^2}+2\frac{\partial^2 P}{\partial x\partial y}+\frac{\partial^2 P}{\partial y^2}\right).$$

В результате

$$R(i)\approx\left|d^2P_{2,2}(x,y)\right|h^{2r-2}=\left(\left|\frac{\partial^2 P}{\partial x^2}\right|+2\left|\frac{\partial^2 P}{\partial x\partial y}\right|+\left|\frac{\partial^2 P}{\partial y^2}\right|\right)h^{2r-2}. \quad (20)$$

Смысл характеристики $R(i)$ заключается в том, что в окрестностях точек или линий разрыва второго рода параметр $R(i)\to\infty$, а в окрестностях локальных экстремумов он принимает наибольшие значения. Следовательно, $R(i)$ позволяет в процессе поиска вычислительных ошибок локализовать те места функции, где формируются ее бесконечные разрывы.

В процессе тестирования программного модуля формируется множество квадратов $D=\{D_1,D_2,...,D_n\}$, причем каждому квадрату ставится в соответствие векторный критерий $K(i)=(K_1(i),K_2(i))$.

Множество квадратов D_k упорядочивается лексикографически и из них выбирается квадрат, удовлетворяющий условию D_t : $(\max\limits_i K_1(i),\max\limits_i K_2(i))$. Выделенный квадрат D_t подвергается последующему делению. Процесс тестирования продолжается

до тех пор, пока не выполнится условие остановки $\|D_t\| < \varepsilon$, или не будет найдена точка разрыва.

3. Исследование производительности алгоритма поиска точек разрывов

Эффективность (производительность) предложенного алгоритма поиска точек разрывов второго рода будем оценивать по числу обращений к тестируемой функции. Объемы вычислений в десятки тысяч обращений к разрывной функции можно считать незначительными поскольку, как показали эксперименты, стандартные методы глобальной оптимизации либо вообще не находят разрывов, либо требуют существенно больших затрат машинного времени. Исследование производительности предлагаемого алгоритма проводилась с использованием нескольких модифицированных тестовых функций. Результаты испытаний представлены в табл. 1.

Таблица 1. Результаты испытаний

№	Тестируемая функция	Среднее число обращений к функции
1	**De Jong 2**. Овражная функция, один глобальный экстремум. Одна точка разрыва (X=(0.21;1.51) $F(x,y)=\frac{100}{100\ (x^2-y)+(1-x)^2+1}+\frac{1}{1-e^{-\frac{(x-0.21)^2+(y-1.51)^2}{0.01}}}.$ $-1.28 \le x, y \le 1.28.$	584
2	**Griewank.** Один глобальный и множество локальных максимумов, одна точка разрыва $F(x,y)=\frac{1}{\frac{x^2+y^2}{200}-\cos(x)\cos(y/\sqrt{2})+1},$ $-20 \le x, y \le 20.$	676
3	**Модифицированная Растригина**. Множество линий разрывов $F(x,y)=\frac{1}{(10\cos(2\pi x)-x^2)-(10\cos(2\pi y)-y^2)},$ $-5.12 \le x, y \le 5.12.$	181

Из таблицы видно, что в среднем точки разрыва функции второго рода идентифицируются предложенным алгоритмом за сравнительно небольшое количество обращений к функции. Сложнее решаются задачи, когда функция содержит «трубчатые» разрывы или имеет многочисленные локальные оптимумы. Значительно проще реализуется поиск линий разрывов тестируемой функции.

Заключение

Вычислительные эксперименты с тестовыми функциями показали, что методы глобальной оптимизации существенно сокращают трудоемкости алгоритмов тестирования вычислительных программных модулей.

Литература

1. Липаев В.В. Отладка сложных программ. Методы, средства, технология. М.: Энергатомиздат, 1993.
2. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. Самара: Самар. гос. аэрокосм. ун-т. 1999.
3. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. М.: Наука, 1978.
4. Коварцев А.Н., Попова-Коварцева Д.А. Многомерный параллельный алгоритм глобальной оптимизации модифицированным методом половинных делений // В мире научных открытий. 2012. № 8.1(32).
5. Gergel V.P., Strongin R.G. Parallel computing for globally optimal decision making on cluster systems // Future Generation Computer Systems. 2005. V. 21. № 5.

СИСТЕМА ТРЕХМЕРНОГО ОТОБРАЖЕНИЯ КОНФИГУРАЦИИ МЕТАЛЛИЧЕСКИХ КЛАСТЕРОВ

А.Н. Коварцев, В.С. Смирнов

Самарский государственный аэрокосмический университет им. С.П. Королёва

Описана система, позволяющая строить трехмерное изображение и проводить визуальный анализ конфигурации металлических кластеров. Платформой разработки послужила среда Matlab 2012, функционирующая в операционной системе Windows 8.

Введение

Одной из практических и теоретических задач нанотехнологий является оптимизация структуры атомных и молекулярных кластеров, цель которой - нахождение лучших в некотором смысле объемных конфигураций наноразмерных частиц. Оптимальные конфигурации кластеров могут строится с использованием разных моделей межатомного взаимодействия, каждая из которых по-своему описывает поведение атомов в кластерах, учитывая межатомные силы. Одной из наиболее используемых моделей является так называемый «кластер Морса». Эта модель позволяет описывать различные конфигурации металлических кластеров и находить среди них оптимальные, имеющие минимальную потенциальную энергию меж-атомных связей (на рис. 1 приведены примеры подобных конфигураций с сайта Кембриджского университета [1]).

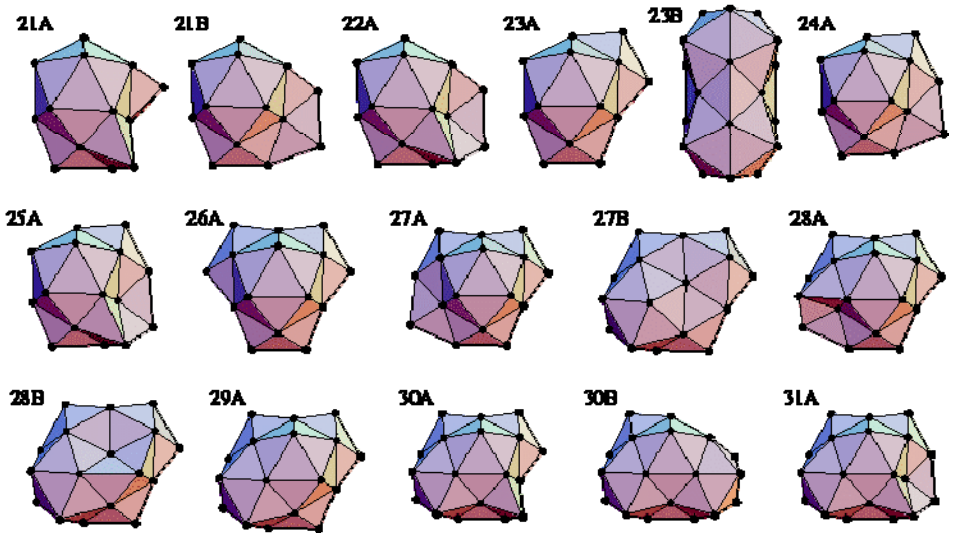


Рис. 1. Примеры оптимальных конфигураций металлических кластеров

В предлагаемом сообщении обсуждаются вопросы разработки системы конструирования и трехмерной визуализации кластеров Морса. Создание такого графического интерфейса с набором средств визуального анализа поможет исследователю находить оптимальные конфигурации многоатомных кластеров. В частности, при развитых средствах визуального анализа для поиска оптимальной конфигурации можно опереться на интуицию пользователя созданной системы, применять различные эвристики для реконфигурирования кластера.

1. Платформа разработки

Всякий комплекс программ можно оценить двумя критериями: точность и удобство. Точность означает, что при поступлении на вход комплекса допустимых входных значений, на его выходе будут получены верные результаты. Нередко лишь эта сторона является главной заботой разработчиков программного обеспечения, тогда как удобству работы с вычислительной системой уделяется гораздо меньше внимания. Напротив, в задании на разработку рассматриваемой системы задачи обеспечения наглядности и удобства интерфейса были едва ли не более высоки, чем требования к точности результатов вычислений потенциалов энергетических полей.

Фундаментальные вопросы организации интерфейса «человек-компьютер» рассмотрены в монографиях [2-4]. В рассматриваемой задаче основу интерфейсной компоненты программной системы составляет подсистема изображения и визуального анализа трехмерных структур. Построение 3D-образов сложных тел (у нас кластеров) тесно связана с технологией триангуляции трехмерных поверхностей.

Анализ показал, что среди доступных сред программирования надлежащим комплексом возможностей обладает платформа – «математическая лаборатория» MATLAB [5]. В ее стандартном инструментарии присутствуют функции, позволяющие построить триангуляцию Делоне для множества точек на плоскости или в трехмерном пространстве (рис. 2).

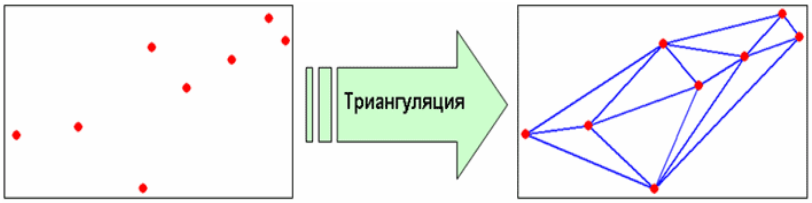


Рис. 2. Триангуляция Делоне, используемая при построении трехмерных выпуклых оболочек множества точек

В вычислительных экспериментах было установлено, что возможности и мощность выбранной платформы достаточны для реализации разрабатывавшейся системы. Однако при использовании стандартных инструментов MATLAB для триангуляции Делоне вокруг интересующих нас кластеров можно было обрисовывать лишь трехмерную выпуклую оболочку, когда впадины поверхности сглаживаются. Эта картина не соответствует действительной природе металлических кластеров.

2. Базовый алгоритм трехмерной визуализации

Укрупненная блок-схема алгоритма, строящего трехмерные изображения металлических кластеров с учетом того, что поверхность кластера может быть невыпуклой, содержать впадины, а также участки, где несколько внешних точек кластера лежат в одной плоскости, приведена на рис. 3.

Предложенное решение основывается на использовании физических соображений о положении атомов кластера, построения специальной базовой системы координат для точек кластера, применение векторной оптимизации для классификации точек кластера и др.

Визуальный анализ построенных кластеров показал адекватность воспроизведения поверхности кластеров, включая их выпуклости и впадины.

3. Апробация системы

Для оценки качества созданной программной системы проведено сравнение полученных трехмерных моделей металлических кластеров с вариантами оптимальных

конфигураций металлических кластеров, полученными в Кембриджском университете [1]. Изображения кластеров типа 34B и 34C, полученные с помощью построенной программной системы приведены на рис. 4.

Рис. 3. Алгоритм построения трехмерной оболочки кластера

а)

б)

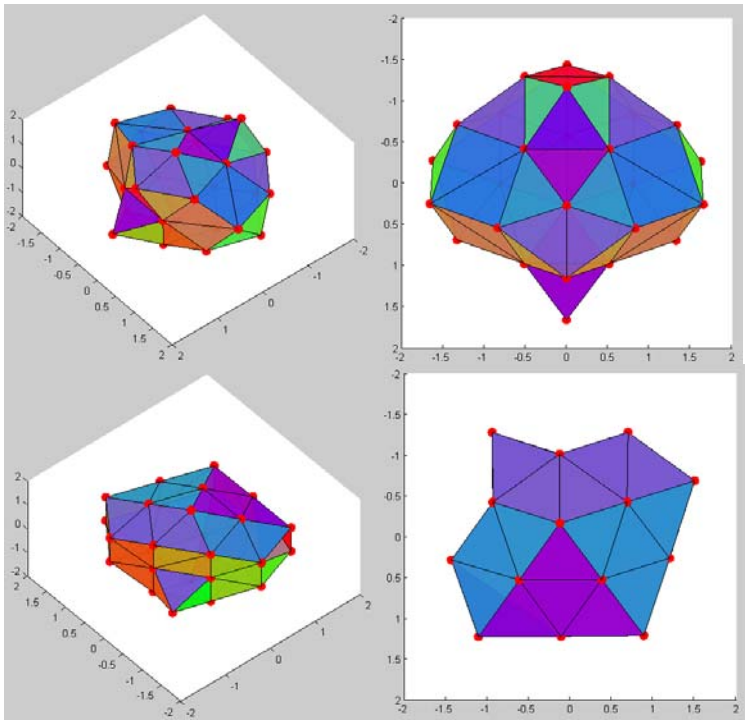


Рис. 4. Полученные изображения металлических кластеров типа 34B (а) и 34C (б)

Визуальный анализ построенных кластеров показал совпадение результатов работы независимо от созданных программных комплексов, включая все характерные особенности (выпуклости и впадины) трехмерных моделей кластеров.

Заключение

Разработанная система трехмерного отображения и визуального анализа многоатомной молекулы – металлического кластера является необходимой частью программного обеспечения для поиска оптимальной конфигурации молекулярно-атомных кластеров с минимальной энергией. Предполагается (и эксперименты это подтверждают), что удобный многофункциональный интерфейс системы окажется пригодным по крайней мере для эвристического поиска возможностей улучшения свойств конфигураций кластеров.

Литература

1. http://doye.chem.ox.ac.uk/jon/structures/Morse/pictures.html.
2. Коутс Р., Влейминк И. Интерфейс «человек-компьютер»: Пер. с англ. – М.: Мир, 1990. – 501 с.
3 Зенкин А. Основы когнитивной компьютерной графики. - М.: Наука, 1991. - 344 с.
4. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем: Пер. с англ. – СПб: Символ-Плюс, 2004. – 272 с.
3. http://www.mathworks.com/products/matlab.

АВТОМАТИЧЕСКОЕ РАСПАРАЛЛЕЛИВАНИЕ ПРОГРАММ В ТЕХНОЛОГИИ ГРАФОСИМВОЛИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Д.П. Крамаренко, В.В. Жидченко

Самарский государственный аэрокосмический университет им. С.П. Королёва

Визуальное программирование - способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста.

Технология графосимволического программирования (ГСП), созданная на кафедре программных систем СГАУ, является одним из способов наглядного представления алгоритмов программы в виде графа управления. Данная технология предоставляет пользователям широкие возможности анализа параллельной структуры алгоритма, корректности применяемых механизмов синхронизации вычислений, поиска тупиковых ситуаций, оценки эффективности алгоритма и т.д.

Данная технология программирования исповедует два основополагающих принципа:

- визуальную, графическую форму представления алгоритмов программ и других компонент их спецификации;
- принцип структурированного процедурного программирования.

В качестве методологической основы для представления алгоритмов в работе используется модель объекта с дискретными состояниями. Основу такой модели составляет предположение, что для любого объекта программирования тем или иным способом можно выделить конечное число состояний, в которых он может пребывать в каждый момент времени. Тогда развитие вычислительного процесса можно ассоциировать с переходами объекта из одного состояния в другое.

Существует множество графических моделей описания алгоритмов параллельных вычислений. В области описания вычислительных алгоритмов наиболее удобной представляется форма, близкая описанию блок-схемам, которая реализована в параллельной версии технологии ГСП.

На данный момент имеется большой фонд последовательных граф-программ. Так как в среде ГСП реализована возможность параллельного исполнения кода, то преобразование последовательных программ в параллельные представляет большой интерес. Однако выполнение подобного преобразования граф-программ вручную является трудоемкой задачей. Поэтому в данной работе ставится задача разработать алгоритм для автоматического распараллеливания готовых последовательных граф-программ, что позволит существенно облегчить этот процесс.

Для решения данной задачи производится поиск информационной зависимости между вершинами в граф-программе, организация необходимых структур данных, получение карты всех маршрутов граф-программы и на основе проведенного анализа происходит изменение структуры графа. На данный момент в тестовую версию среды ГСП добавлен программный модуль, исполняющий алгоритм автоматического преобразования последовательной граф-программы в параллельную. Так же в разработке находится алгоритм, позволяющий на основе анализа зависимостей и структуры графа распараллелить выполнение циклов.

ОПТИМИЗАЦИЯ ПЕРЕСЫЛОК ДАННЫХ ПО MPI ПРИ РЕШЕНИИ СИСТЕМЫ УРАВНЕНИЙ НАВЬЕ-СТОКСА НА GPU-КЛАСТЕРЕ

М.А. Кривов[1,3], М.Н. Притула[1], А.А. Гаврилов[2], А.А. Дектерёв[2]

[1]ООО «ТТГ Лабс»

[2]Институт теплофизики им. С.С. Кутателадзе СО РАН

[3]Московский госуниверситет им. М.В. Ломоносова

В работе описан опыт оптимизации пересылок данных между узлами GPU-кластера при решении системы уравнений Навье-Стокса. Приведены три схемы обмена граничными элементами между смежными доменами и представлены результаты их тестирования на GPU-кластере из 12 узлов.

Введение

При использовании графических ускорителей (GPU) для решения промышленных задач аэрогидродинамики крайне важным свойством выбранного решателя является поддержка вычислительных систем с множеством GPU. Причём причина кроется отнюдь не в необходимости многократного сокращения времени расчётов (что иногда тоже является важным), а в ограниченности размера памяти отдельного ускорителя, размер которой обычно не превышает нескольких гигабайт. Как следствие, для проведения расчётов на сетке с требуемым уровнем детализации необходим вычислительный кластер как минимум с несколькими, а то и несколькими десятками графических ускорителей, суммарный размер памяти которых окажется достаточным для хранения сетки и всех сопутствующих массивов.

Для обеспечения данного свойства от разработчика может потребоваться осуществление дополнительной оптимизации решателя. Действительно, если в случае классических гомогенных кластеров при выполнения одной MPI-операции происходит лишь копирование данных в рамках одного класса памяти из некоего массива в MPI-буфер, то при использовании GPU-системы этому будет предшествовать дополнительное копирование данных по шине PCI Express, которое является достаточно медленной операцией. Более того, так как при фиксированном размере домена один GPU оказывается в разы производительнее центрального процессора, то и вклад MPI-операций в суммарное время работы GPU-версии пакета становится намного больше (в процентном соотношении). Другими словами, при переходе на GPU-кластер время обмена между узлами возрастает, и с большой вероятностью подобные операции становятся узким местом.

В данной статье описывается опыт осуществления подобных оптимизаций в GPU-версии пакета SigmaFlow [1], разрабатываемого институтом теплофизики им. С.С. Кутателадзе СО РАН.

Схема обмена граничными элементами сетки

В рассматриваемом пакете озвученная оптимизация проводилась для решения системы уравнений Навье-Стокса, в котором при обмене данными больше всего времени тратилось на пересылку значений граничных элементов сетки. В данной реализации исходная сетка с помощью программы METIS [2] разбивалась на домены, для каждой

пары которых строились два массива индексов вершин, являющихся граничными элементами, что схематично показано на следующей иллюстрации:

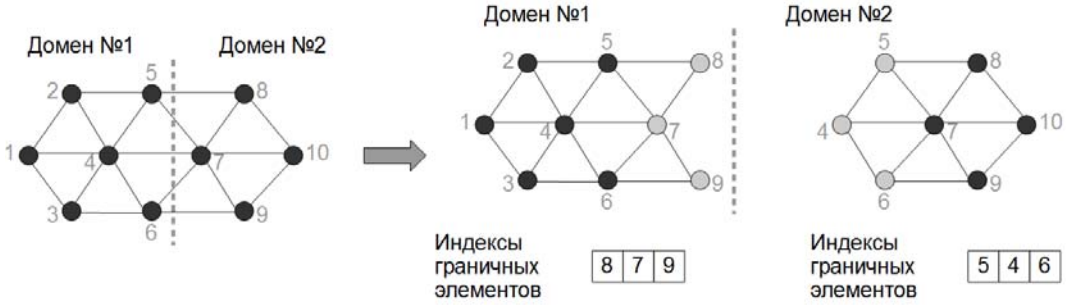


Рис. 1. Схема разбиения сетки на домены

В этом примере узел, обрабатывающий домен №1, на каждой итерации будет пересылать узлу, обрабатывающему домен №2, массив со значениями искомой величины для вершин {5, 4, 6}, в то время как обратно получать значения для вершин {8, 7, 9}. В случае, если количество доменов больше двух, то подобные пересылки осуществляются в рамках каждой пары.

В оптимизируемом пакете все расчёты проводятся на неструктурированных сетках, поэтому для хранения данных в большинстве случаев использовался формат, в котором сетка задаётся двумя массивами, описывающими рёбра соответствующего графа. Так, в первом массиве хранятся индексы первой вершины, соединяемой соответствующим ребром, во втором – второй. Значения же всех искомых величин (например, трёх компонент скорости, плотности и давления) задаются в виде дополнительных массивов, индексы элементов которых совпадают с индексами вершин, как показано на следующей иллюстрации:

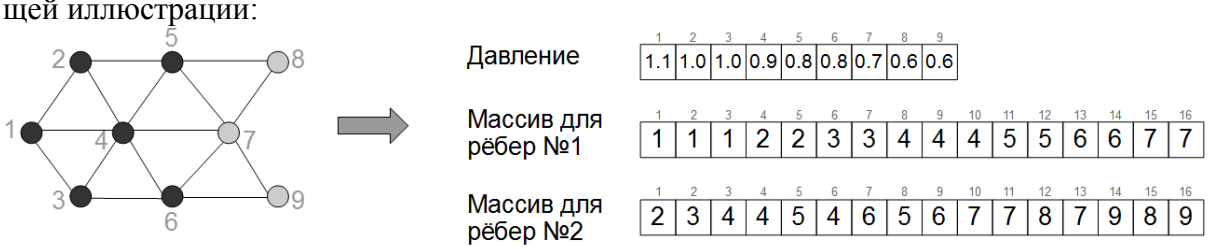


Рис. 2. Формат хранения сеток и заданных на них величин

В таком случае схема обмена граничными элементами сводится к выполнению трёх этапов: 1) копирование всех граничных элементов в рамках каждой пары доменов в общий MPI-буфер, 2) обмен нужными значениями по схеме «каждый узел с каждым», 3) копирование обновлённых граничных элементов в исходные массивы. Стоит отметить, что данное описание является достаточно схематичным и не совсем точно соответствует реализованной в пакете более сложной схеме асинхронного обмена, но для дальнейшего изложения решаемых проблем его вполне достаточно.

При переходе к вычислениям на GPU основной проблемой становится необходимость осуществления операций копирования данных по шине PCI Express, так как они являются крайне медленными. В рамках рассматриваемой схемы это означает, что массив с элементами искомой величины (в примере выше – давлением), расположен только в памяти GPU, и некоторые из его элементов необходимо копировать в MPI-буфер, созданный в памяти центрального процессора. В данной работе было реализовано три варианта формирования данного MPI-буфера, которые описаны в следующем разделе. Также стоит сказать, что на момент написания статьи отдельно не были рассмотрены аппаратные решения типа технологии NVidia GPUDirect [3], применение которых является дальнейшим развитием данной работы.

Варианты формирования массива граничных элементов

Первым и самым простым (и в тоже время самым неэффективным) вариантом является копирование массива с искомой величиной целиком в память центрального процессора, осуществление обмена граничными элементами, после чего копирование обновлённого массива обратно в память GPU. Чтобы потеря производительности не была крайне большой, копирование массива осуществлялось в pinned-память, полученную с помощью функции cudaHostRegister() и обладающую лучшими характеристиками за счёт отказа от страничной адресации. Данный вариант схематично представлен на следующей иллюстрации:

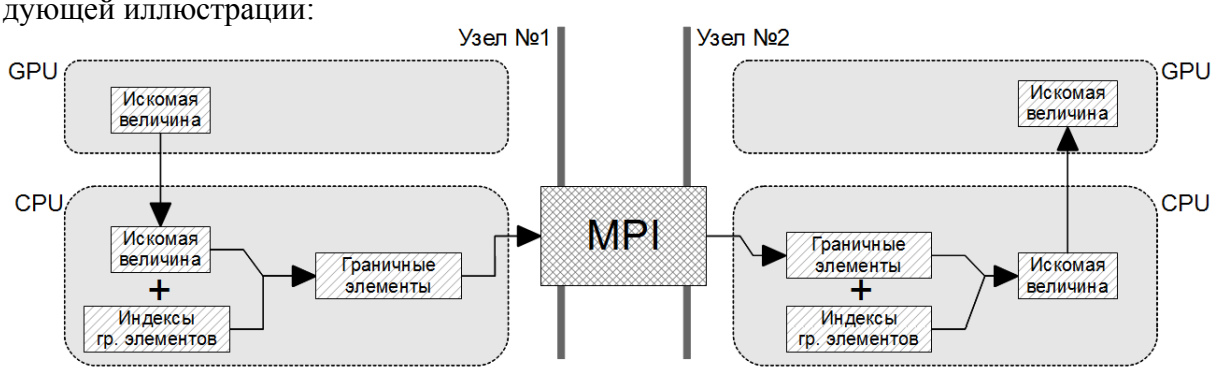


Рис. 3. Первая (исходная) схема пересылки значений граничных элементов

Количество подобных операций равно (N-1)·N·M, где N – количество MPI-процессов, а M – число синхронизируемых массивов, которое обычно равно либо одному (скалярная величина), либо трём (векторная величина), либо девяти (градиенты компонент векторной величины). При этом за счёт укрупнения запросов реально одним узлом производится (N-1) ·2 асинхронных MPI-операций, и всего M·2 операций копирования по шине PCI Express.

В следующей реализации формирование массива граничных элементов производилось силами GPU, после чего в память центрального процессора копировался лишь требуемый массив граничных элементов намного меньшего размера, в результате чего многократно уменьшался объём пересылаемых данных, однако и увеличивалось число операций копирования – с M·2 до M·(N – 1)·2.

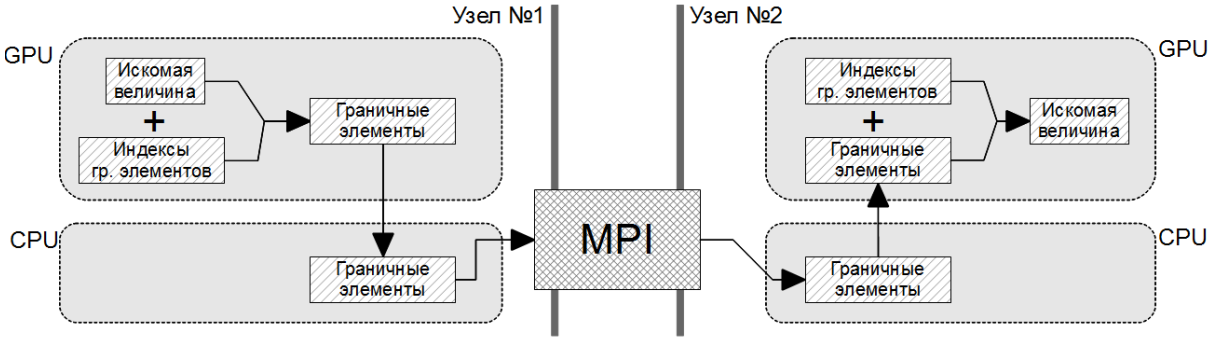


Рис. 4. Вторая схема пересылки граничных элементов

Недостатком данной реализации является как раз излишне большое количество вызовов CUDA-функций, что при работе с сетками небольшого размера может заметно замедлить проведение расчётов из-за задержки шины, которая на практике может доходить до долей миллисекунды. Поэтому также была разработана третья реализация, в которой формирование MPI-буфера для обмена сразу между несколькими узлами проводится исключительно силами GPU, после чего полученный буфер лишь один раз копируется по шине PCI Express, как это показано на следующей иллюстрации:

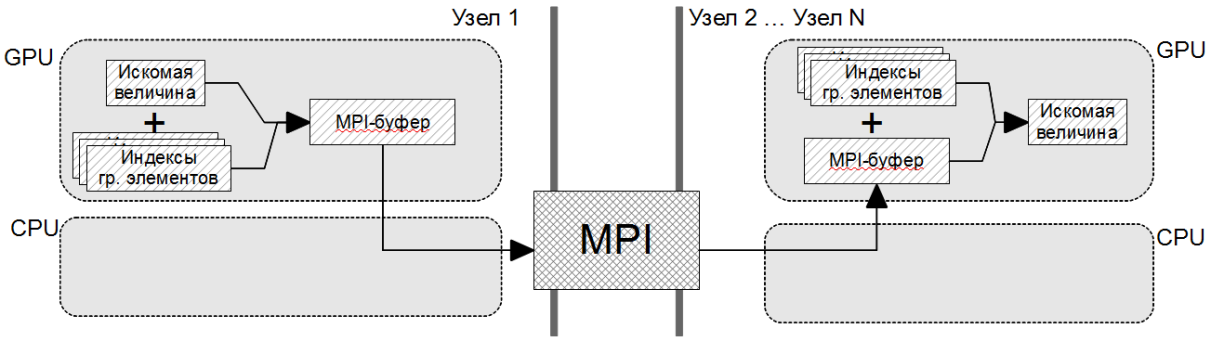


Рис. 5. Третья схема пересылки граничных элементов

Когда количество узлов равно двум, вторая и третья реализация оказываются практически идентичными. При этом с ростом количества узлов во втором варианте также растёт количество CUDA-операций, в то время как в третьем варианте оно остаётся инвариантным и равным двум операциям на узел. Минусом же последнего варианта является синхронность – в нём не удаётся инициировать обмен до тех пор, пока весь MPI-буфер не будет готов.

Для упрощения, в дальнейшем первая реализация пересылок будет обозначаться как «original», так как именно она была реализована в пакете изначально, вторая - «bandwidth-optimized», так как по оценкам она должна была оказаться предпочтительней для больших сеток, третья - «latency-optimized», так как в ней делается акцент на минимизацию задержек. Вышеперечисленные отличия всех трёх версий приведены в следующей таблице 1.

Таблица 1. Сравнение трех предложенных реализаций пересылок данных

Реализация	**Объём пересылаемых данных по шине PCI-E**	**Количество операций копирования по шине PCI-E**	**Асинхронность обмена по MPI**
Original	Большой (M*2*S_1)	M * 2	Да
Bandwidth-optimized	Небольшой (порядка M*2*S_2)	M * (N-1) * 2	Да
Latency-optimized	Небольшой (порядка M*2*S_2)	2	Нет

Здесь S_1 – размер массива с синхронизируемой величиной, S_2 – размер массива со всеми граничными элементами (для всех смежных доменов), N – количество доменов (количество GPU), M – число синхронизируемых скалярных величин.

Результаты тестирования

Тестирование проводилось на кластере из 12 узлов, каждый из которых оснащён двумя шестиядерными процессорами и тремя ускорителями NVidia Tesla C2050. При этом каждый домен сетки в зависимости от теста обрабатывался либо одним ядром центрального процессора, либо одним ускорителем, а на каждый узел создавался ровно один MPI-процесс, которому поручалась обработка соответствующего домена. Таким образом стоит признать, что кластер использовался лишь частично, однако выбранная схема позволяет более качественно оценить именно время пересылок по межсетевому соединению InfiniBand.

Перед тем, как переходить к тестированию всех трёх реализаций, стоит сказать несколько слов об объёме пересылаемых данных по шине PCI Express. Объем напрямую зависит как от типа сетки, так и числа доменов, на которое она разбивается. Ниже приведён график, показывающий, насколько сократился объём пересылаемых данных при синхронизации одного скалярного поля, заданного на сетке из 500 тысяч узлов.

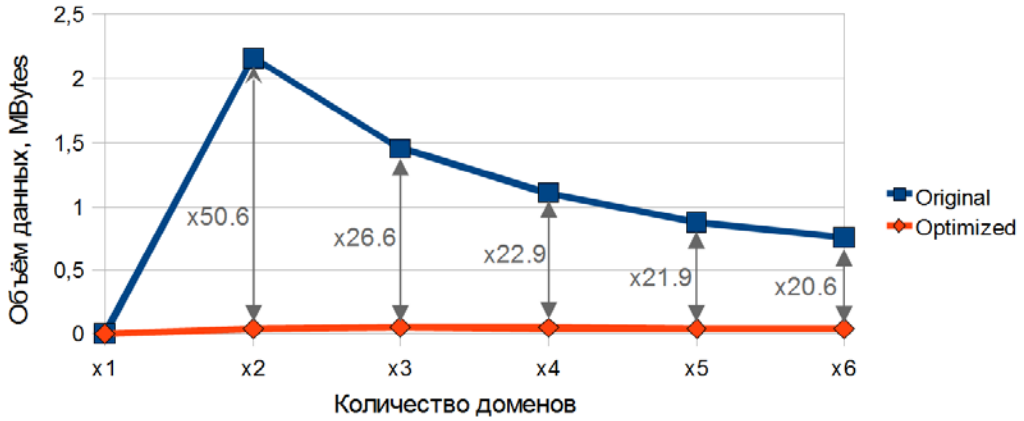


Рис. 6. Объём пересылаемых данных по шине PCI-E в зависимости от реализации

Как видно, проведённые оптимизации позволили сократить объём пересылаемых данных в 20-50 раз и, как показано на рис. 7, также существенно ускорить сами расчёты.

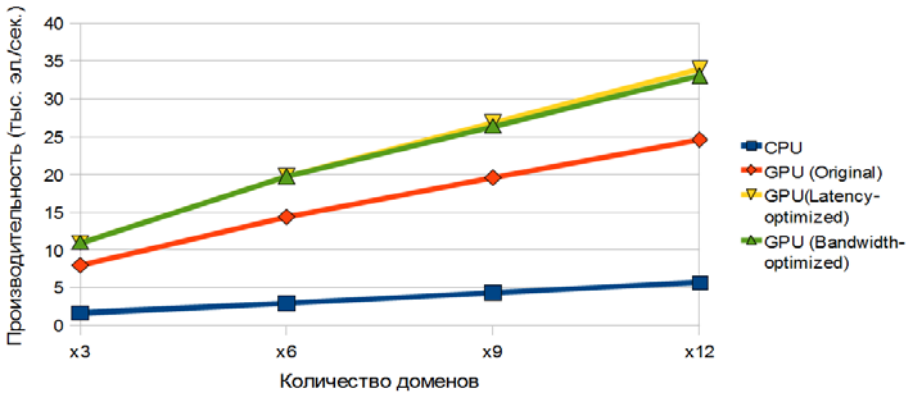


Рис. 7. Скорость проведения расчётов на сетке из 2 млн узлов

Поскольку на рис. 7 отличия двух реализаций друг от друга практически незаметны, то отдельный интерес представляет следующий график, где приведено только время, затрачиваемое на пересылку данных.

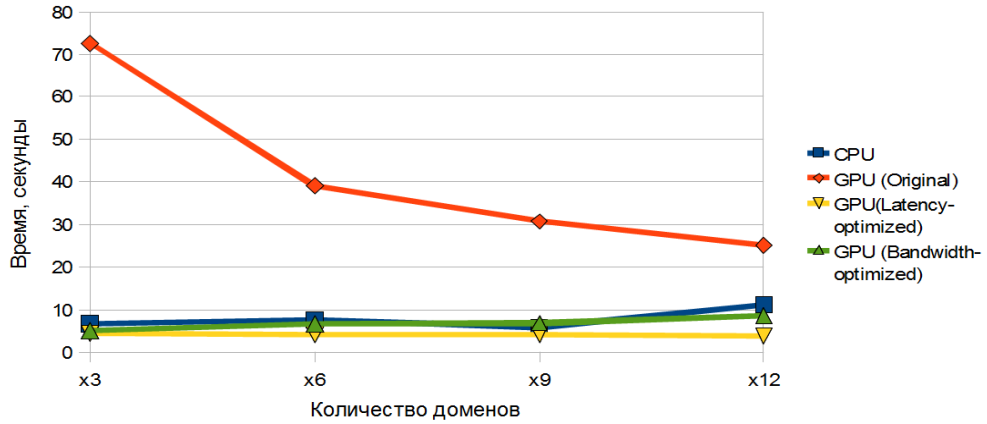


Рис. 8. Время, затрачиваемое на пересылки во всех реализациях, 2 млн узлов

Таким образом оптимальной реализацией оказалась версия, в которой MPI-буфер целиком формируется силами GPU. Случай, когда граничные элементы по частям копируются в память центрального процессора, оказался почти в два раза медленнее, однако по сравнению с общим временем работы выигрыш практически незаметен. Наконец, изначальная версия, в которой весь массив с искомой величиной целиком копиру-

ется в память центрального процессора, как и ожидалось, оказалась крайне медленной. Также стоит отметить, что в результате проведённых оптимизаций время на MPI-пересылки в GPU-версии оказалось даже меньше, чем в исходной реализации для центрального процессора. Это объясняется тем, что формирование массива с граничными элементами силами GPU происходит намного быстрее, что полностью компенсирует затраты на копирование данных по шине PCI Express.

Стоит отдельно сравнить вклад операций по пересылке граничных элементов в общее время расчётов, что проиллюстрировано на рис. 9.

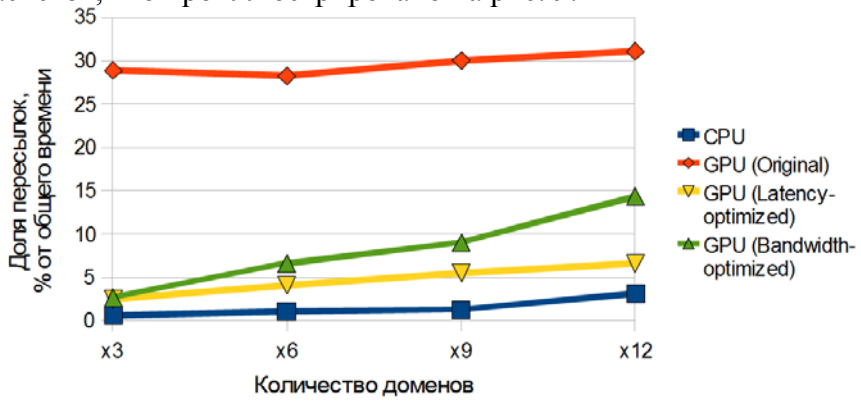


Рис. 9. Доля операций по пересылке данных в общем времени расчётов, 2 млн узлов

Как видно, благодаря схеме с формированием MPI-буфера на GPU удалось сократить время пересылок в процентном соотношении ровно в 5 раз - с 30% до 6%, в результате чего рассматриваемые операции перестали быть узким местом.

Заключение

Три описанные в работе схемы пересылок граничных элементов между смежными доменами являются достаточно очевидными, и в тоже время применимыми для решения разнообразных уравнений математической физики. Однако к вопросу выбора подходящей схемы стоит подходить основательно, так как скорость расчётов с использованием той или иной реализации может существенно изменяться как в зависимости от типа решаемой задачи, так и выбранной аппаратной платформы.

Проведённое в рамках работы тестирование показало, что при решении векторного уравнения на GPU-кластере имеет смысл формировать MPI-буфер исключительно силами GPU. В частности, данная оптимизация позволила сократить время выполнения операций обмена граничными элементами по сравнению с исходной реализацией в 7.5 раз (с 30 до 4 секунд на сетке из 2 млн ячеек и кластере из 12 узлов), или в 5 раз в процентном отношении к итоговому времени расчётов (с 30% до 6%), благодаря чему данные операции перестали быть узким местом выбранного модуля пакета.

Литература

1. Гаврилов А.А., Кривов М.А., Гризан С.А., Дектерёв А.А. GPU-версия CFD пакета SigmaFlow: портирование и оптимизация с использованием инструментария TTG Apptimizer // Параллельные вычислительные технологии (ПаВТ'2013). Челябинск: Издательский центр ЮУрГУ, 2013. С. 106-115.
2. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM Journal on Scientific Computing. 1999. Vol. 20, No. 1. P. 359 – 392.
3. Электронный ресурс https://developer.nvidia.com/gpudirect.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ВЗАИМОДЕЙСТВИЯ МОЛЕКУЛ КРЕМНИЯ НА СИСТЕМАХ С РАСПРЕДЕЛЕННОЙ ПАМЯТЬЮ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ

И.Б. Крылов, А.В. Линев, А.М. Сатанин

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлен алгоритм моделирования взаимодействия молекул в ансамбле, состоящем из трех типов частиц: кремния, кислорода и золота. Для моделирования используются методы молекулярной динамики, в частности метод численного интегрирования Верле. Взаимодействие между частицами описывается двумя типами потенциалов, а именно потенциалом Терсофа и потенциалом Морзе. Алгоритм реализован для гибридных систем с распределенной памятью. Дана оценка эффективности и масштабируемости параллельной версии алгоритма для систем с общей и распределенной памятью.

Введение

Метод молекулярной динамики позволяет выполнять расчеты характеристик физических систем, состоящих из больших массивов атомов. В отличие от расчетов, основанных на первопринципных подходах, когда расчет ведется на языке квантовой теории (требуется вычислять электронные волновые функции, обменную и корреляционную энергию и т.д.), метод молекулярной динамики позволяет работать со значительно большим числом атомов благодаря использованию интерполяционного потенциала взаимодействия частиц, который должен правильно передавать характер их взаимодействия на атомных масштабах и зависит от параметров моделируемой системы. В представляемой работе проводимое моделирование отражает методику получения нанокристаллов, разрабатываемую в НИФТИ (под руководством к.ф.м.н. А.В. Ершова), согласно которой нанокристаллы кремния образуются путем отжига сверхтонких слоев аморфного кремния (толщиной в несколько десятков нанометров), и необходимо рассчитывать взаимодействие молекул в ансамбле, состоящем из трех типов частиц: кремния, кислорода и золота.

В расчетах используются следующие эмпирические потенциалы: взаимодействие в сочетаниях атомов кремния и кислорода описывается модифицированным потенциалом Терсоффа, параметры которого подобраны так, чтобы получающаяся согласно численным расчетам интерфейсная граница Si/SiO2 согласовывалась с экспериментальными данными; взаимодействие атомов золота между собой, а также с атомами кислорода и кремния описывается потенциалом Морзе. Для решения системы дифференциальных уравнений второго порядка для всех частиц используется метод Верле.

Алгоритм реализован для систем с распределенной памятью, использующих графические ускорители. Дана оценка эффективности и масштабируемости параллельной версии алгоритма для систем с общей и распределенной памятью. Проведенные эксперименты показывают эффективность применения вычислительных возможностей графических ускорителей в данной задаче, а также практически линейный рост производительности в зависимости от количества используемых узлов при пропорциональном увеличении размеров задачи.

1. Моделирование молекулярного взаимодействия методом Верле

Для описания взаимодействия между атомами кремния воспользуемся эмпирическим анизотропным потенциалом Терсоффа (J.Tersoff) [1].

Запишем выражение для потенциальной энергии *i*-й частицы в поле, создаваемом всеми остальными частицами системы:

$$V_i = \sum_{i \neq j} V_{ij} . \tag{1}$$

Потенциал Терсоффа парного взаимодействия V_{ij} можно представить как модифицированный изотропный потенциал Морзе:

$$V_{ij} = Ae^{-2\lambda r_{ij}} - Be^{-\lambda r_{ij}} . \tag{2}$$

Здесь $Ae^{-2\lambda r_{ij}}$ – слагаемое, отвечающее за отталкивание, $Be^{-\lambda r_{ij}}$ – слагаемое, отвечающее за притяжение. Переход от изотропного потенциала Морзе (2) к анизотропному потенциалу Терсоффа можно осуществить, приняв, что коэффициент B зависит от положения всех соседних частиц:

$$B = B_{ij} = B_0 \exp\left(-\frac{z_{ij}}{b}\right) ,$$

где

$$z_{ij} = \sum_{k \neq i,j} \left(\frac{e^{-r_{ij}}}{e^{-r_{ik}}}\right)^n \left(c + e^{-d\cos ijk}\right) \tag{3}$$

Таким способом учитывается ослабление межатомной связи, обусловленное перетягиванием электронной плотности соседними частицами.

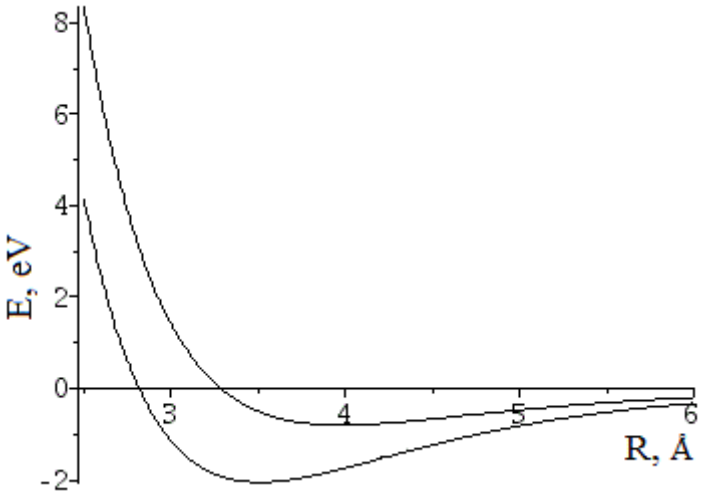


Рис. 1. Вид потенциала Терсоффа при разных конфигурациях соседних частиц

Как видно из рис. 1, межатомный потенциал является экспоненциально убывающей функцией расстояния. Это значит, что потенциал можно считать короткодействующим. С учётом близкодействия суммирование по *j* в (1) и по *k* в (3) можно проводить только по частицам, находящимся на расстоянии меньше заданного расстояния r_c, что существенно уменьшает вычислительную сложность алгоритма. Параметры потенциала Терсоффа подбираются так, чтобы удовлетворить ряду экспериментальных фактов, в частности, требуя соответствие параметров численно «выращенной» алмазной решетки тем, которые измеряются методами рентгеновской дифракции. Необходимо

реализовать алгоритм, моделирующий динамику атомов кремния, взаимодействие между которыми описывается потенциалом Терсоффа.

Согласно [2] взаимодействие атомов кремния и кислорода характеризуется модифицированным потенциалом Терсоффа, параметры которого подобраны так, чтобы получающаяся согласно численным расчетам интерфейсная граница Si/SiO2 согласовывалась с экспериментальными данными.

Взаимодействие атомов золота между собой, а также с атомами кислорода и кремния можно описать парным эмпирическим потенциалом Морзе. Параметры потенциалов для указанных пар атомов приведены в работах [3-8].

Закон движения атома кремния массы m_0 в поле других частиц определяется вторым законом Ньютона. Запишем его в виде

$$m_0 \ddot{\vec{r}}_i = \nabla_{\vec{r}_i} V_i. \quad (4)$$

В выражении (4) градиент берётся по координате *i*-ой частицы.

Решение системы дифференциальных уравнений второго порядка для всех частиц будем проводить с помощью позиционного метода Верле [9]:

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) + \ddot{\vec{r}}_i(t)\Delta t^2 + o(\Delta t^2). \quad (5)$$

Данный метод имеет второй порядок точности по времени, однако он не является самостартующимся. Для получения $\vec{r}_i(\Delta t)$ выполняется несколько шагов интегрирования методом Эйлера [10] с шагом $\Delta t_{Eul} << \Delta t$. Будем считать, что по оси z система ограничена жёсткими стенками, а по осям *x* и *y* она бесконечна. Для моделирования бесконечного пространства воспользуемся периодическими граничными условиями.

Начальное расположение атомов соответствует аморфному состоянию вещества. Для создания подобной структуры атомы помещались точки, смещённые на случайный вектор от узлов правильной решётки.

2. Параллельный алгоритм для систем с общей памятью

Для реализации задачи моделирования на системах с общей памятью было проведено исследование алгоритма на предмет масштабируемости при использовании многоядерных CPU и GPU. Исследование показало, что наиболее трудоемкой частью программы является этап вычисления сил, действующих на частицу, на шаге интегрирования по Верле. Так же в ходе исследования алгоритма выяснилось, что данная часть программы может эффективно выполняться на многоядерных центральных процессорах, графические ускорители, в свою очередь, показывают еще более лучшие результаты.

На рис. 2 и 3 приведено соответственно время работы параллельной CPU- и GPU-версий программы моделирования искусственных экситон-плазмонных наноструктур методами молекулярной динамики, и ускорение GPU версии относительно CPU.

Проведенные эксперименты показывают, что использование графического процессора в данной задаче дает существенное ускорение по сравнению с обычным многоядерным процессором.

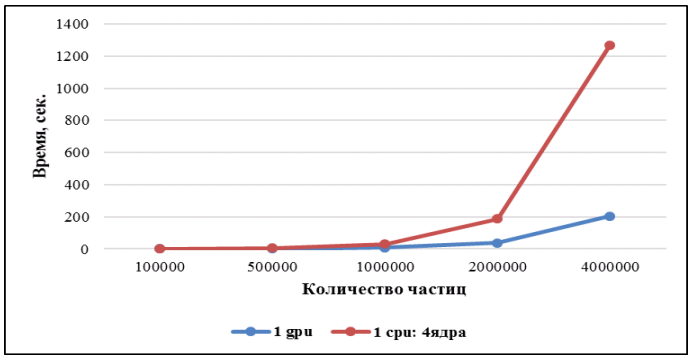


Рис. 2. Время работы параллельной CPU- и GPU-версий в зависимости от количества частиц

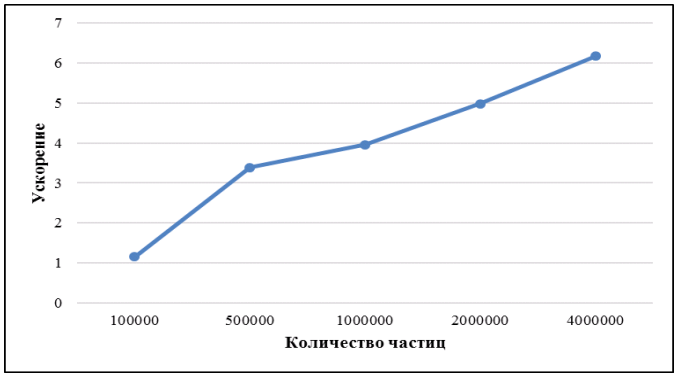


Рис. 3. Ускорение GPU-версии относительно параллельной CPU-версии

3. Параллельный алгоритм для систем с распределенной памятью

Также были проведены тесты для сравнения MPI-версии программы, работающей как с использованием, так и без использования графических процессоров.

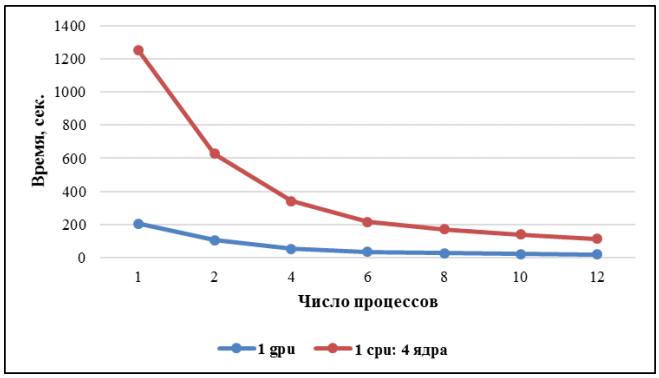


Рис. 4. Время работы параллельной CPU- и GPU-версий для систем с распределенной памятью в зависимости от числа процессов

Как видно из результатов тестов (см. рис. 4), использование графических процессоров, установленных на узлах гетерогенного кластера, также повышает производительность решения.

На рис. 5 приведены результаты запуска на суперкомпьютере «Ломоносов» параллельного GPU-алгоритма на конфигурации из 4 млн атомов.

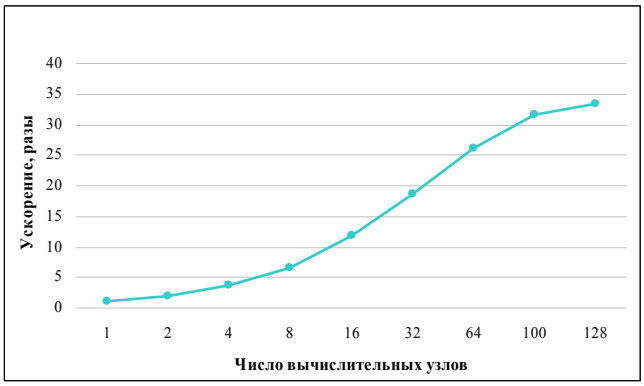


Рис. 5. Ускорение параллельной GPU версии для систем с распределенной памятью относительно запуска на одном узле

Таким образом, разработанный программный компонент моделирования процессов формирования искусственных экситон-плазмонных наноструктур показывает хорошую масштабируемость при запусках с использованием до 128 узлов (57344 GPU-ядер), обеспечивая на 128 узлах ускорение в 33 раза по сравнению с запуском на 1 узле. Уменьшение эффективности распараллеливания падает с увеличением числа используемых узлов, что объясняется повышением доли передач данных при сохранении общего объема вычислений.

Заключение

В данной работе приведены результаты производительности параллельных алгоритмов для моделирования взаимодействия молекул. Проведенные эксперименты показывают эффективность применения вычислительных возможностей графических ускорителей в данной задаче, а также практически линейный рост производительности в зависимости от количества использованных узлов кластера с установленными на них GPU.

Работа выполнена при организационной поддержке Лаборатории информационных технологий ННГУ.

Литература

1. Tersoff J. New empirical approach for the structure and energy of covalent systems. Physical Review B, 1988. 37(12), 6991. APS.
2. Yasukawa A. Using an Extended Tersoff Interatomic Potential to Analyze the Static-Fatigue Strength of SiO2 under Atmospheric Influence // JSME International Journal, Series A. 1996. Vol. 39, No. 3. P. 313-320.
3. Umeno Y., Kitamura T., Date K., Hayashi M., Iwasaki T. Optimization of interatomic potential for Si/SiO2 system based on force matching // Computational Materials Science. 2002. Vol. 25. P. 447–450.
4. Nordlund K., Samela J. Nuclear Instruments and Methods // Physics Research B. 2009. Vol. 267. P. 1420–1423.
5. Dongare A.M., Zhigilei L.V., Rajendran A.M., LaMattina B. Interatomic potentials for atomic scale modeling of metal.matrix ceramic particle reinforced nanocomposites // Composites: Part B. 2009. Vol. 40. P. 461–467/
6. Ryu S., Cai W. A gold–silicon potential fitted to the binary phase diagram // Phys. Condens. Matter. 2010. Vol. 22. P. 055401.

7. Watanabe T., Fujiwara H., Noguchi H., Hoshino T., Ohdomari I. Novel Interatomic Potential Energy Function for Si, O Mixed Systems // Jpn. J. Appl. Phys. 1999. Vol. 38. L366.
8. Ohta H., Hamaguchi S. Classical interatomic potentials for Si–O–F and Si–O–Cl systems // J. Chem. Phys. 2001. Vol. 115. P. 6679.
9. Гулд Х., Тобочник Я.. Компьютерное моделирование в физике. М.: Мир, 1990. ч.2.
10. Герасименко Н.Н., Пархоменко Ю.Н. Кремний – материал наноэлектроники. Москва: Техносфера, 2007. 352 с.
11. Anderson A.J., Lorenz C.D., Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units // J. Comput. Phys. 2008. 227. P. 5348-5359.
12. Hou C., Ge W. GPU-accelerated molecular dynamics simulation of solid covalent crystals // Molecular Simulation 1-8. 2011.
13. Liu W., Schmidt B., Voss G., Muller-Wittig W. Molecular Dynamics Simulations on Commodity GPUs with CUDA. HiPC 2007, LNCS 4873, 185–196 (2007).
14. Chen F., Ge W., Guo Li., He X., Li B., Li J., Li X., Wang X., Yuan X. Multi-scale HPC system for multi-scale discrete simulation – Development and application of a supercomputer with 1 Petaflops peak performance in single precision. Particuology 7. 2009. P. 332–335.

РАСПРЕДЕЛЕННАЯ СИСТЕМА ПОДГОТОВКИ ТЕСТОВЫХ ДАННЫХ ДЛЯ ОЦЕНКИ КАЧЕСТВА ВИДЕОДЕТЕКТИРОВАНИЯ ТРАНСПОРТНЫХ СРЕДСТВ

Е.В. Кутилов, В.Д. Кустикова

Нижегородский госуниверситет им. Н.И. Лобачевского
E-mail: kutilov.egor@gmail.com, valentina.kustikova@gmail.com

Введение

Внедрение систем компьютерного зрения требует предварительной апробации и оценивания результатов качества, т.е. сравнения результатов, полученных в процессе работы системы, с некоторыми эталонными. Апробация систем видеодетектирования предполагает подготовку набора тестовых видео с разметкой интересующих объектов. Подготовка тестовых данных решается посредством поиска видео или съемкой реальной сцены. Выполнение разметки требует разработки программных средств автоматизации, т.к. сложность процедуры разметки растет с увеличением продолжительности видео и ростом количества размечаемых объектов.

Обзор существующих решений

Типичными примерами программных сервисов и систем, позволяющих автоматизировать процесс разметки, являются Piotr's Matlab Toolbox [1] и InteractLabeler [2].

Сервис Piotr's Matlab Toolbox представляет собой набор инструментов, облегчающих работу с изображениями в процессе разметки положений объектов. Он позволяет выделять объекты посредством построения окаймляющего прямоугольника. Также позволяет присваивать имена выделенным областям. Для использования данного сервиса необходимо установить математический пакет MATLAB [3]. Наряду с этим требуются определенные навыки работы в среде MATLAB для ее применения.

InteractLabeler является приложением для Windows. Оно позволяет в автоматическом режиме производить сегментацию изображений с использованием специальных алгоритмов. Каждая область при этом окрашивается в свой цвет, впоследствии любой такой области может быть присвоено свое имя. Также пользователь может с помощью кисти сам доопределить области. Результат сохраняется в виде изображения с закрашенными областями.

Постановка задачи

Целью настоящей работы является разработка распределенного сервиса для разметки видео с дорожным движением (выделение транспортных средств разных классов), позволяющего удаленно через Интернет производить разметку объектов на кадрах видео. Сервис должен обеспечивать обработку запросов большого числа одновременно подключенных к нему клиентов и автоматический сбор полученной от клиента разметки. Создание сервиса позволит упростить задачу подготовки тестовых данных для последующей апробации алгоритмов компьютерного зрения для поиска транспортных средств разных классов.

Архитектура программной системы

Система построена на базе клиент-серверной архитектуры.

1. Подсистема клиентской части. Служит для отображения кадров видео и редактирования разметки.
2. Серверная подсистема. Данная подсистема выполняет распределение кадров видеофайлов между клиентами, формирует предварительную разметку и передает клиентам ее вместе с кадрами, сохраняет полученную от клиента разметку в определенном формате.

Архитектура разработана согласно принципам Domain Driven Design [4] и разделена на несколько слоев:

1. *Domain*. Слой, описывающий предметную область. Содержит сущности для представления разметки (кадр, объект, разметка кадра и т.д.) и реализации планировщика кадров набора видео между клиентами системы. Также определяет интерфейсы взаимодействия с этим слоем и интерфейсы доступа к внешним данным.
2. *Infrastructure*. Слой, реализующий доступ к внешним данным для чтения/записи разметки. Классы данного слоя, как правило, реализуют интерфейсы доступа, определенные в Domain.
3. *Presentation*. Слой, отвечающий за представление данных клиенту. Содержит интерфейс, определяющий методы для отображения данных. Также содержит методы для обработки действий пользователя и представления данных из Domain на пользовательский интерфейс.
4. *Application*. Слой, инициализирующий и создающий остальные слои. Производит связывание слоев с помощью принципа инъекции зависимостей.

Рассмотрим основной сценарий работы системы (рис.1). При запросе пользователя на получение нового кадра, планировщик выдает кадр видео. Для этого кадра с помощью реализованного класса инфраструктурного слоя проводится поиск предварительной разметки. Если разметка обнаружена, то она вместе с кадром передается клиенту, иначе отправляется кадр без предварительной разметки. Далее клиент редактирует разметку: наносит новые окаймляющие прямоугольники объектов, сдвигает границы уже существующих, удаляет некорректно нанесенные положения. При сохранении отредактированной разметки кадра планировщик по идентификатору клиента находит обрабатываемый им номер кадра и сохраняет разметку. Затем клиенту отправляется следующий неразмеченный кадр. Параллельно в системе через определенный интервал времени происходит сохранение состояния с целью последующего восстановления в случае некорректного завершения работы системы.

Рис. 1. Диаграмма использования

Планирование кадров видео

Идея работы планировщика состоит в том, что видео разделяется на фрагменты – наборы последовательных кадров. Планировщик формирует очереди свободных и обрабатываемых фрагментов. При очередном запросе клиента выполняется проверка на существование фрагмента, закрепленного за данным клиентом, и отправка очередного кадра согласно следующей схеме:

1. Если такого фрагмента нет, то за клиентом закрепляется один из фрагментов из очереди свободных фрагментов. Выбранный фрагмент добавляется в список обрабатываемых фрагментов. В случае отсутствия свободных фрагментов, выбирается новое видео для разметки, формируются новые фрагменты и добавляются в очередь свободных фрагментов.
2. Если фрагмент найден, то получаем фрагмент, закрепленный за клиентом.
3. Отправка следующего кадра клиенту из закрепленного за ним фрагмента.

Клиенту каждый раз предоставляются последовательные кадры из одного фрагмента. В случае длительного отсутствия клиента, к которому привязан фрагмент, данный фрагмент без размеченных кадров перемещается в список свободных фрагментов.

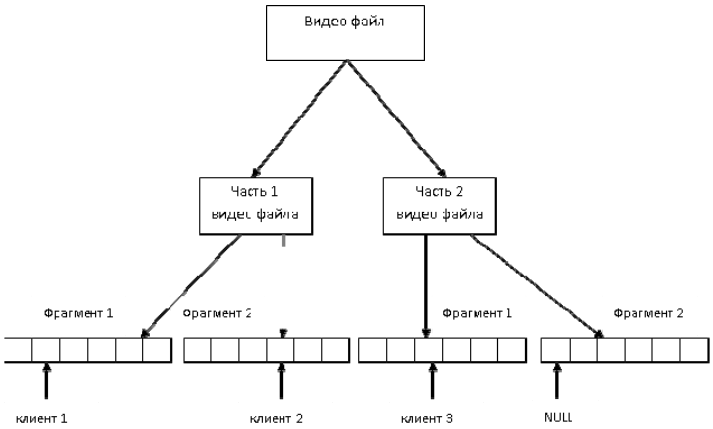


Рис. 2. Схема разбиения видео файла на фрагменты

Программная реализация

Для реализации программного комплекса использовался язык C#. Серверная подсистема разработана в виде сервиса Windows Communication Foundation [5]. Для организации работы с видео и изображениями была выбрана библиотека AForge.NET[6], как наиболее удобная в использовании. Сохранение разметки кадров и вспомогательной информации выполняется в разработанном формате на языке разметки XML. Клиентская часть реализована в виде приложения Windows Forms.

Заключение

Разработана система подготовки тестовых данных для апробации алгоритмов видеодетектирования транспортных средств. Предусмотрены механизмы сохранения состояния системы для его восстановления в случае аппаратного или программного сбоя. Разработанная система позволяет в дальнейшем использовать другой способ хранения данных, например, на базы данных. Использование интерфейсов для представления пользовательского интерфейса обеспечивает простое создание пользовательских интерфейсов с применением разных технологий.

В дальнейшем планируется создание компоненты для формирования «на лету» предварительной разметки, исходя из уже размеченных клиентом кадров. Для этого

предполагается применять алгоритмы машинного обучения и компьютерного зрения, распознающие объекты в реальном времени.

Работа выполнена в лаборатории «Информационные технологии» факультета ВМК ННГУ им. Н.И. Лобачевского.

Литература

1. Piotr's Matlab Toolbox [http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html].
2. InteractLabeler [http://mi.eng.cam.ac.uk/projects/cvgroup/software/index.html].
3. MATLAB [http://www.mathworks.com/products/matlab/].
4. Evans E. Domain Driven Design Quickly. – InfoQ, 2006.
5. MSDN Windows Communication Foundation [http://msdn.microsoft.com/ru-ru/library/dd456779(v=vs.100).aspx].
6. AForge.NET [http://www.aforgenet.com/].

КОГЕРЕНТНЫЕ СВОЙСТВА ЦИКЛИЧЕСКОГО ХАОСА

Т.А. Леванова[1], Г.В. Осипов[1], А.С. Пиковский[2]

[1]Нижегородский госуниверситет им. Н.И. Лобачевского
[2]Университет Потсдама, Германия

В настоящее время внимание многих исследователей привлекают гетероклинические циклы, которые представляют собой устойчивые режимы переключений между метастабильными состояниями. Эти состояния наблюдаются в течение все более продолжительного времени, так что период цикла растет неограниченно. В фазовом пространстве систем, которые воспроизводят описанный эффект, фазовая точка последовательно посещает окрестности траекторий седлового типа [1, 2], что приводит к последовательной активации элементов системы [3]. Системы с гетероклиническим контуром между седловыми состояниями равновесия [4, 5] и седловыми циклами [6-8] в настоящий момент хорошо изучены.

Интересным вариантом гетероклинического цикла является циклический хаос, впервые описанный в работе [9] и в дальнейшем изучавшийся в работах [2, 10, 11]. В этом случае состояния равновесия в системе являются хаотическими аттракторами. В простейшем случае могут наблюдаться последовательные периоды хаотической активности элементов системы, перемежающиеся с периодами покоя. Важно отметить следующую особенность: в то время как сами элементы системы являются сильно нерегулярными, переключения активности между ними имеют высокую степень регулярности.

В данной работе изучаются когерентные свойства циклического хаоса. Мы характеризуем нерегулярность переключений постоянной фазовой диффузии и исследуем её зависимость от параметров. На примере трех моделей циклического хаоса (ансамбли элементов Лоренца, Рёсслера и элементов, заданных логистическим отображением) изучаются статистические свойства циклического хаоса.

Литература

1. Arshwin P., Chossat P. Attractors for robust heteroclinic cycles with continua of connections // J. Nonlin. Sci. 1998. V. 8. P. 105.
2. Ashwin P., Field M. Heteroclinic networks in coupled cell systems // Arch. Rational Mech. Anal. 1999. V.148. P. 107.
3. Afraimovich V.S., Zhigulin V.P., Rabinovich M.I. On the origin of reproducible sequential activity in neural circuits // Chaos, 2004. V. 14(4). P. 1123.
4. Field M. Lectures on Bifurcations, Dynamics and Symmetry // Chapman and Hall/CRC, 1996. 176 p.
5. Krupa M. Robust heteroclinic cycles (review article) // J. Nonlinear Sci. 1997. V. 7. P. 129.
6. Komarov M.A., Osipov G.V., Suykens J.A.K. Sequentially activated groups in neural networks // EPL 2009. V. 86. P. 60006.
7. Komarov M.A., Osipov G.V., Zhou C.S. Heteroclinic contours in oscillatory ensembles // Phys Rev E 2013. V. 87(2). P. 022909.
8. Mikhaylov A.O., Komarov M.A., Levanova T.A., Osipov G.V. Sequential switching activity in ensembles of inhibitory coupled oscillators // EPL 2013. V. 101. P. 20009.
9. Dellnitz M., Field M., Golubitsky M., Hohmann M., Ma J. Cycling chaos // IEEE Trans. Circuits Systems 1995. V. 42(10). P. 3821.
10. Palacios A. Cycling chaos in one-dimensional coupled iterated maps // Int. J. Bif. Chaos 2002. V. 12 (8). P. 1859.
11. Ashwin P. Cycles homoclinic to chaotic sets; robustness and resonance // Chaos, 1997. V. 7. P. 207.

ОПРЕДЕЛЕНИЕ ЧАСТОТЫ ПЕРИОДИЧЕСКИХ РЕШЕНИЙ МОДЕЛИ ФИТЦХЬЮ-НАГУМО С ИСПОЛЬЗОВАНИЕМ БАЗИСОВ ГРЕБНЕРА

С.А. Малашенко

Нижегородский госуниверситет им. Н.И. Лобачевского
E-mail: malashenko_sergei@yahoo.com

Модель Фитцхью-Нагумо является одной из классических моделей в биофизике и часто используется при описании распространения волн в активных биологических средах. В статье ставится задача определения частоты периодических решений исходной модели в зависимости от одного из параметров системы.

Введение

Модель Фитцхью-Нагумо является одной из классических моделей в биофизике и часто используется при описании распространения волн в активных биологических средах, таких как сердечная мышца, или мозговая ткань. Уравнения модели содержат две переменные v и w и имеют вид

$$\begin{aligned} \dot{v} &= \tau\left(v - \frac{v^3}{3} - w\right) + I_{ext}, \\ \tau\dot{w} &= v + a - bw. \end{aligned} \tag{1}$$

где τ, a, b, I_{ext} – параметры модели, которые при некоторых значениях допускают существование периодических решений [1]. В работе для таких решений строится зависимость частоты от некоторых параметров модели.

Метод решения

Общим методом поиска периодических решений нелинейных систем дифференциальных уравнений является метод гармонического баланса [2]. Основная идея метода состоит в представлении решения уравнения в виде усеченного разложения в ряд Фурье $v(t) = \sum_{n=-N}^{N} c_k e^{-ik\omega t}$. Подстановку предполагаемого решения в исходное дифференциальное уравнение и последующее решения результирующей системы уравнений $p_0(c_1, \dots, c_N, \omega)$, $\dots, p_N(c_1, \dots, c_N, \omega)$ относительно коэффициентов разложения $c_1, \dots, c_N$ и ω.

Исходную модель (1) преобразуем к дифференциальному уравнению относительно v получим

$$\ddot{v} + \tau\dot{v}(v^2 - 1) + \frac{b}{\tau}\dot{v} + v - b\left(v - \frac{v^3}{3} + I_{ext}\right) + a = 0. \tag{2}$$

Для упрощения рассмотрим случай, когда $a = 0$, $b = 0$. Решение представляем в виде $v(t) = \sum_{n=-3}^{3} c_k e^{-ik\omega t}$, учитывая, что $\bar{c_k} = c_{-k}$. Получаем следующую систему уравнений относительно $c_1, c_{3i}, c_{3r}, \tau$ искомой частоты ω:

$$\begin{aligned} f_1 &= -c_1\omega^2 - \tau c_1^2 c_{3i}\omega + c_1, \\ f_2 &= 2\tau c_1 c_{3r}^2\omega + \tau c_1^2 c_{3r}\omega + 2\tau c_1 c_{3i}^2\omega + \tau c_1^3\omega - \tau c_1\omega, \\ f_3 &= -9c_{3r}\omega^2 - 3\tau c_{3i}c_{3r}^2\omega - 3\tau c_{3i}^3\omega - 6\tau c_1^2 c_{3i}\omega + 3\tau c_{3i}\omega + c_{3r}, \\ f_4 &= 3\tau c_{3r}^3\omega - 9c_{3i}\omega^2 + 3\tau c_{3i}^2 c_{3r}\omega + 6\tau c_1^2 c_{3r}\omega - 3\tau c_{3r}\omega + \tau c_1^3\omega + c_{3i}. \end{aligned} \tag{3}$$

Для системы полиномиальных уравнений (3) построим базис Гребнера с lex-упорядочиванием $\omega < \tau < c_1 < c_{3i} < c_{3r}$, такое упорядочивание позволит выделить зависимость частоты ω от $\tau, c_1, c_{3i}, c_{3r}$. Базис строится в параллельном режиме с использованием библиотеки parallelGBC [3]. В результате один из полиномов базиса Гребнера имеет следующий вид

$$3249\omega^{10} + (549\tau^2 - 6213)\omega^8 + (3754 - 567\tau^2)\omega^6 + (159\tau^2 - 842)\omega^4 + \\ +(53 - 13\tau^2)\omega^2 - 1 = 0. \quad (4)$$

Отметим что при любом $\tau > 0$ уравнение (4) относительно ω^2 имеет один действительный корень. Представим ω^2 в виде разложения по степеням τ^2, коэффициенты вычислим, используя уравнение (4), тогда получим следующее разложение:

$$\omega^2 = 1 - \frac{1}{8}a^2 + \frac{3}{256}a^4 + \frac{1}{512}a^6 + O(a^8). \quad (5)$$

Результаты расчетов

Построение базиса Гребнера для системы (3) занимает несколько секунд в однопроцессорном режиме, поэтому эффективность распараллеливания исследуется на системе алгебраических уравнений, полученной при $a \neq 0$ и $b \neq 0$. Для такой, существенно большей, системы алгебраических уравнений ищется базис Гребнера в параллельном режиме с использованием библиотеки parallelGBC [3], которая реализует алгоритм [4] с использованием технологий OpenMP и MPI. В таблице приводятся времена расчетов базиса, а также демонстрируется эффективность распараллеливания.

Таблица 1. Результаты масштабируемости MPI-версии

Количество процессоров	Время расчета, сек	Эффективность, %
1	91.88	100
2	106.2	85
4	132.1	68
8	189.7	48

Заключение

В работе получена зависимость частоты периодического решения модели Фитцхью-Нагумо от параметра τ. Одним из этапов получения результата является вычисление базиса Гребнера, которое выполняется посредством библиотеки paralellGBC [3]. В среднем ускорение составляет 4.3 раза на 8 процессах, время работы уменьшено в среднем до 21 секунды для указанного числа процессов. Полученное ускорение неплохой результат при условии, что распараллелены не все этапы алгоритма, а только этап редуцирования системы уравнений.

Литература

1. Елькин Ю.Е. Автоволновые процессы // Математическая биология и биоинформатика. 2006. Т.1, № 1. С. 27-40.
2. Anderson C., Geer J. Power series expansions for the frequency and period of the limit cycle of the van der Pol equation // SIAM J. Appl. Math.42. 1982. P. 678-693.
3. Neumann S. A modified parallel F4 algorithm for shared and distributed memory architectures, SCSS, 2013. 15. P. 70-80.
4. Faugere J.-Ch. A new efficient algorithm for computing Grobner bases (F4) // Journal of Pure and Applied Algebra. 1999. 139(1-3): P. 61-88.

ОБЗОР МЕТОДОВ ВЫПОЛНЕНИЯ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ НАД ЧИСЛАМИ БОЛЬШОЙ РАЗРЯДНОСТИ НА ПРИМЕРЕ ОПЕРАЦИИ ДЕЛЕНИЯ

А.С. Назаров

Северо-Кавказский федеральный университет

Приведен обзор методов работы с многоразрядными данными при использовании устройств с ограниченой разрядной сеткой, таких как FPGA. Показано, как можно построить параллельную арифметику с использованием системы остаточных классов. На примере операции деления показаны основные преимущества предлагаемого подхода.

Введение

На практике часто возникают задачи, требующие высокой скорости работы и надежности вычислительных средств. Микропроцессоры общего назначения способны исполнить практически любой прикладной алгоритм, соответствующим образом запрограммированный с использованием системы команд этого процессора, однако по скорости их нельзя сравнить с быстродействующими заказными интегральными схемами, предназначенными для конкретных приложений (application-specific integrated circuit, ASIC), реализующих те и только те функции, которые необходимы для решения вполне конкретной задачи. При настройке ASIC на данную проблему можно получить микросхему, которая будет значительно меньше, дешевле и быстрее, чем универсальный программируемый микропроцессор. Однако проектирование данной схемы является крайне сложной и затратной задачей.

Например, классические методики синтеза цифровых фильтров (ЦФ) характеризуются тем, что на функциональном уровне представления передаточная функция рассчитывается без учета ограниченной точности представления коэффициентов [1]. Возникающие при этом проблемы точности реализации характеристик ЦФ обычно решаются на более низком уровне, в частности на структурном. Решение этой задачи затрудняется огромным количеством известных структурных решений в условиях отсутствия их систематизации и методик их перебора.

Наиболее перспективным является использование альтернативного типа вычислительных устройств – программируемых логических интегральных схем (ПЛИС) архитектуры FPGA. FPGA представляет собой большую интегральную схему, которая может быть сконфигурирована после изготовления и в любой момент при использовании. Каждый чип FPGA состоит из конечного числа предопределенных ресурсов с программируемыми соединениями для реализации реконфигурируемого цифрового контура и блоков ввода/вывода. При этом ПЛИС данного вида обеспечивают аппаратно-тактируемую скорость и надежность вычислений.

Важной особенностью FPGA является естественный параллелизм каналов обработки данных, что делает их незаменимыми при решении множества прикладных задач. В качестве примеров можно привести криптографические преобразования в реальном времени и цифровую обработку сигналов. Особенностью данных задач является необходимость работы с числами большой разрядности, налагая при этом жесткие рамки на скорость преобразований. Однако в случае работы с FPGA разработчики сталкива-ются

с проблемой ограниченной длины обрабатываемых чисел – от 16 до 128 бит в зависимости от архитектуры устройства. В реальных задачах при этом можно встретить числа диапазоном до 4096 бит. В связи с этим важной задачей является разработка эффективных алгоритмов выполнения операций с числами, разрядность которых превышает имеющиеся возможности для данного аппаратного средства.

1. Методы представления многоразрядных чисел

Представление многоразрядных чисел в виде кортежа чисел стандартной размерности налагает определенные ограничения. Связано это, в первую очередь, с необходимостью отслеживания переполнения типа данных. Например, для хранения результата произведения необходимо вдвое больше бит, чем для хранения каждого из множителей. Таким образом, если каждый из множителей – 16-битное беззнаковое целое число, то для корректного выполнения операции умножения необходимо выделить для результата уже 32 бита (тип unsigned integer). При этом для продолжения выполнения арифметических действий с результатом операции, необходимо «вернуть» его в 16-битный вид, осуществив перенос избытка в старшие разряды программно. Другой вариант представления многоразрядных чисел заключается в использовании битового массива большой длины. Однако на всех устройствах доступ к элементам этого массива является дорогостоящей операцией. При этом свой отпечаток налагает необходимость действий по каждому элементу отдельно с учетом позиционности представления.

Описанные обстоятельства демонстрируют ограниченность позиционных систем счисления: обработка происходит последовательно от младших разрядов к старшим с переносом значений между разрядами.

В свою очередь, система остаточных классов (СОК) допускает параллельную обработку по каждому основанию ввиду отсутствия межразрядных связей [2]. Система остаточных классов (Residue number system) является непозиционной системой представления чисел.

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 $a_i = A(\mathrm{mod}\ p_i)$ 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. Здесь как и в позиционных системах необходимо ограничивать разрядность операндов для элементарных объектов и типов данных половиной доступной длины. Однако при выполнении операций в СОК избыточная часть результата отбрасывается, т.к. операции проводятся в кольцах классов вычетов. Результат вычисляется верно за счет того, что информация об исходных числах распределена между всеми остатками. Данное обстоятельство позволяет производить вычисления в параллельных потоках, не отслеживая взаимосвязи между ними.

2. Операции над числами большой разрядности на примере деления

Хорошим примером, позволяющим оценить возможности распараллеливания алгоритмов с использованием СОК, является алгоритм деления. Как известно, основу целочисленного деления составляет общепринятый способ деления с помощью операций вычитания или сложения и сдвига [3]. Задача сводится к вычислению частного $Q \in \mathrm{Z} \ge 0$ и остатка $S \in \mathrm{Z} \ge 0$ от деления $Y \in Z \ge 0$ на $D \in Z > 0$ таких, что $Q = \lfloor Y / D \rfloor$ и $S = Y - QD$ при $S < D$. Деление выражается как последовательность вычитаний делителя сначала из делимого, а затем из образующихся в процессе деления частичных остатков.

При реализации на уровне логических схем делимое $Y(y_{2n-1}\ y_{2n-2}\ ...\ y_1\ y_0)$ обычно представляется двойным словом ($2n$ разрядов, $n \in Z > 0$), делитель $D(d_{n-1}\ d_{n-2}\ ...\ d_1\ d_0)$, частное $Q(q_{n-1}\ q_{n-2}\ ...\ q_1\ q_0)$ и остаток $S(s_{n-1}\ s_{n-2}\ ...\ s_1\ s_0)$ имеют разрядность n. В данной работе рассматриваются только положительные (беззнаковые) числа, что не является значительным ограничением применения результатов работы.

Операция деления выполняется за n итераций и описывается следующим образом:

$$S^{(i)} = 2S^{(i-1)} - q_{n-i}(2^n D),$$

при $S^{(0)} = Y$, $S^{(n)} = 2^n S$ и $q_{n-i} = \begin{cases} 1, \textit{если } (2\mathrm{S}^{(\mathrm{i-1})} - 2^{\mathrm{n}}\mathrm{D}) \geq 0 \\ 0, \textit{если } (2\mathrm{S}^{(\mathrm{i-1})} - 2^{\mathrm{n}}\mathrm{D}) < 0 \end{cases}$.

После n итераций, на основании приведенных формул получаем

$$S^{(n)} = 2^n S^{(0)} - Q(2^n D) = 2^n [Y - (Q \times D)] = 2^n S.$$

Частное от деления $2n$-разрядного числа на n-разрядное может содержать более чем n разрядов. В этом случае возникает переполнение, из-за чего перед выполнением деления необходима проверка условия $Y < 2^n D$.

Данный алгоритм переносится на описанный выше способ представления чисел в виде набора простых целых чисел одинаковой разрядности. При этом необходимо контролировать переносы бит и избытков между разрядами. Учитывая итеративность процесса, данный алгоритм деления представляет достаточно высокую вычислительную сложность.

С другой стороны, деление в системе остаточных классов может быть абсолютно параллельным [4]. При этом в отличие от традиционных методов деления в СОК, данный алгоритм не требует данных о позиционных характеристиках числа, необходимых для сравнения чисел по величине. Алгоритм основывается на переходе от одной системы оснований СОК к другой и соответствует следующей схеме вычислений:

1. Пусть $q_1, q_2, \ldots, q_m$ есть набор модулей, $(a_1, a_2, \ldots, a_m)$ – делимое и $(b_1, b_2, \ldots, b_m)$ – делитель.
2. Определим вспомогательную систему оснований $(q'_1, q'_2, \ldots, q'_m)$, где НОД$(q_i, q'_j) = 1$ для $i = 1, 2, \ldots, m$ и $j = 1, 2, \ldots, m$.
3. Вычислим $(p_1, p_2, \ldots, p_m)$, где $p_i = q'_1 \times q'_2 \times \ldots \times q'_m \bmod q_i$.
4. Переведем $(a_1, a_2, \ldots, a_m)$ по основаниям $q_1, q_2, \ldots, q_m$ в $(a'_1, a'_2, \ldots, a'_m)$ по основаниям $q'_1, q'_2, \ldots, q'_m$.
5. Переведем $(b_1, b_2, \ldots, b_m)$ по основаниям $q_1, q_2, \ldots, q_m$ в $(b'_1, b'_2, \ldots, b'_m)$ по основаниям $q'_1, q'_2, \ldots, q'_m$.
6. Вычисляем $(b_1^{-1}, b_2^{-1}, \ldots, b_m^{-1})$ по основаниям $q_1, q_2, \ldots, q_m$, $(b_1'^{-1}, b_2'^{-1}, \ldots, b_m'^{-1})$ по основаниям $q'_1, q'_2, \ldots, q'_m$ и $(p_1^{-1}, p_2^{-1}, \ldots, p_m^{-1})$ по основаниям $q_1, q_2, \ldots, q_m$.
7. Вычисляем $(c'_1, c'_2, \ldots, c'_m) = (a'_1, a'_2, \ldots, a'_m) \times (b_1'^{-1}, b_2'^{-1}, \ldots, b_m'^{-1})$.
8. Переведем $(b_1'^{-1}, b_2'^{-1}, \ldots, b_m'^{-1})$ по основаниям $q'_1, q'_2, \ldots, q'_m$ в $(b_1''^{-1}, b_2''^{-1}, \ldots, b_m''^{-1})$ по основаниям $q_1, q_2, \ldots, q_m$ и $(c'_1, c'_2, \ldots, c'_m)$ по основаниям $q'_1, q'_2, \ldots, q'_m$ в $(c_1, c_2, \ldots, c_m)$ по основаниям $q_1, q_2, \ldots, q_m$.

9. Вычисляем $(a_1, a_2, \ldots, a_m) = [(a_1, a_2, \ldots, a_m) \times (b_1''^{-1}, b_2''^{-1}, \ldots, b_m''^{-1}) - (c_1, c_2, \ldots, c_m)] \times$ $\times (p_1^{-1}, p_2^{-1}, \ldots, p_m^{-1})$.
10. Вычисляем $(b_1, b_2, \ldots, b_m) = [(b_1, b_2, \ldots, b_m) \times (b_1''^{-1}, b_2''^{-1}, \ldots, b_m''^{-1}) - (1,1,\ldots,1)] \times$ $\times (p_1^{-1}, p_2^{-1}, \ldots, p_m^{-1})$.
11. Если $(b_1, b_2, \ldots, b_m) = (1,1,\ldots,1)$, то $(a_1, a_2, \ldots, a_m)$ есть частное; иначе – перейти к шагу 4.

Стоит отметить, что вычисление обратных мультипликативных величин можно организовать посредством LUT таблиц – специального способа организации вычислений за счет хранения результатов в памяти устройства. Использование СОК, таким образом, позволяет сократить количество последовательных операций за счет параллельного выполнения операций сложения и умножения, что придает данному способу ускорение более чем в 1,5 раза относительно первого описанного алгоритма в зависимости от разрядности исходного числа.

Вывод

Работа с многоразрядными числами на устройствах с ограниченой разрядной сеткой налагает ряд требований на алгоритмы выполнения операций и способы представления данных чисел. Стандартный позиционный подход ограничен за счет невозможности распараллелить процесс вычислений. С другой стороны, система остаточных классов, являясь непозиционной системой счисления, дает дополнительные преимущества при работе с числами большой длины.

Литература

1. Лесников В.А., Наумович Т.В. Теоретико-числовые и алгебро-топологические аспекты структурного синтеза цифровых фильтров. – Сб. трудов X-ой международной научно-технической конференции «Радиолокация, навигация и связь». Т. 1, Воронеж, 2004. – С. 209-217.
2. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Ряднов С.А. Модулярные параллельные вычислительные структуры нейропроцессоных систем. – М.: Физматлит, 2003. – 288 с.
3. Сорокин Н.Ю., Вохмин Е.В. Высокоскоростные модули целочисленного деления // Проблемы разработки перспективных микроэлектронных систем – 2005. Сборник научных трудов / Под общ. ред. А.Л. Стемпковского. М.: ИППМ РАН, 2005. – С. 434-439.
4. Chin-Chen Chang, Yeu-Pong Lai: A division algorithm for residue numbers. // Applied Mathematics and Computation, 172(1), 2006 – P. 368-378.

ПРОВЕРКА ЦЕЛОСТНОСТИ ДАННЫХ, ПЕРЕСЫЛАЕМЫХ МЕЖДУ MPI-ПРОЦЕССАМИ В MCCT 1.0

А.В. Огородников, М.И. Старов

ООО «Центр компетенций и обучения», Саров

e-mail: mistarov@compcenter.org

Разработка современных параллельных программных комплексов MPI – непростая задача, при которой возрастает вероятность некорректного применения MPI вызовов. Представляемый инструмент MCCT призван значительно ускорить поиск подобных ошибок в реализации параллельного кода MPI приложений. Важной особенностью в версии MCCT 1.0 является новая проверка целостности данных, пересылаемых между MPI-процессами.

Введение

Анализ и отладка современных параллельных программных комплексов MPI [1] – непростая задача, усложняющаяся целым рядом факторов. Среди них и непрерывный рост количества вычислительных ядер, и применение асинхронных обменов, и использование сложных типов данных, и, в целом, наличие большого количества специализированных ресурсов MPI. Неоднозначность стандарта MPI также привносит свой вклад в усложнение процесса разработки – под этим подразумевается как двоякое использование одних и тех же типов данных (например, MPI_Comm) и функций (например, использующих входной аргумент MPI_IN_PLACE), так и разнообразие реализаций библиотеки MPI. Вдобавок к вышесказанному, как правило, ни компиляторы, ни библиотеки MPI не отслеживают ошибки использования MPI.

В результате вероятность некорректного применения MPI-вызовов возрастает. Наличие данных ошибок в свою очередь может привести к искажению передаваемых данных, зависанию программы и к другим сбоям.

Как правило, при поиске проблем в программном коде разработчики используют программы-отладчики, позволяющие пошагово исследовать выполнение программы. Но подобное исследование всего параллельного программного кода, в котором идет постоянный межпроцессный обмен, является весьма длительным и изменяет поведение исследуемой программы.

Соответственно, является логичным переложить все проверки параллельного кода на вычислительную машину, которая будет выполнять их без задержки и по точно закодированным спецификациям. Это поможет программисту точно определить расположение ошибки в коде и сразу заняться её исправлением.

Эту задачу решает инструмент MCCT 1.0 (MPI Correctness Checking Tool) [2], входящий в состав программного комплекса S-MPI [3] и базирующийся на программном обеспечении с открытым кодом MUST 1.1 [4], призванный значительно ускорить поиск подобных ошибок в реализации параллельного кода MPI-приложений.

Одной из важных особенностей MCCT 1.0 является способность обнаружить нарушение целостности передаваемых данных.

Механизм работы и возможности MCCT

Механизм работы MCCT основан на оберточных функциях (wrappers) стандарта MPI-2, в которых производятся все необходимые проверки корректности использования MPI, и из которых вызывается соответствующая PMPI-функция. Инициализация MCCT происходит при вызове функции MPI_Init или MPI_Init_thread.

При каждом входе в функцию MPI MCCT выполняет проверку ряда условий, таких как: проверка правильности аргументов, соответствие получаемых данных их типу, сопоставление коллективных операций и т.д. Для проведения проверок, которые требуют данные от двух и более процессов, так называемых глобальных проверок, MCCT пересылает проверяемые данные служебному MPI процессу; MCCT создает этот процесс автоматически при использовании скрипта запуска. Кроме того, служебный MPI процесс позволяет разгрузить основные счетные процессы MPI задачи. При невыполнении проверяемых условий MCCT сообщает об обнаруженных проблемах в поток вывода STDOUT.

Инструмент MCCT выполняет проверки следующих типов:

- тестирование параметров функций MPI; при этом учитывается контекст вызовов для данной функции MPI, а также, какие ограничения на нее могут быть наложены с учетом языковой принадлежности используемой реализации MPI (C, Fortran);
- проверка переносимости исходного кода на другие реализации MPI;
- проверка целостности используемых данных в вызовах MPI;
- проверка памяти, используемой пользовательскими ресурсами MPI, на своевременное освобождение;
- проверка возможного наложения памяти, в т.ч. внутри сложных типов данных MPI;
- проверка совпадения типов пересылаемых данных между процессами;
- сопоставление коллективных операций, вызванных из вовлеченных в данный коммуникатор процессов MPI, по ряду признаков;
- отслеживание возможных потерь двухточечных сообщений;
- проверка на условия возникновения взаимоблокировок (deadlocks) между MPI-процессами (как реальные, так и потенциальные).

Проверка целостности пересылаемых данных

Во время передачи данных между MPI-процессами в параллельном приложении возможны различные сбои, приводящие к порче или потере пересылаемых данных. Причинами повреждения данных во время пересылки могут являться:

- ошибка разработчика, некорректно использующего асинхронные двухточечные операции в своём приложении;
- возникновение сбоев внутри используемой библиотеки MPI;
- возникновение сбоев внутри коммуникационной среды.

Основных проверок MCCT, диагностирующих некорректное использование вызовов MPI в приложении и наследуемых из базового продукта MUST, недостаточно для обнаружения повреждения пересылаемых данных.

В связи с этим в версии MCCT 1.0 было решено реализовать новый механизм проверок, основанный на сравнении контрольных сумм переданных данных.

Вычисление контрольных сумм данных происходит в обёрточных MPI-функциях непосредственно перед вызовом PMPI-функции для функций передачи и после вызова PMPI-функции для функций приема данных. Затем контрольные суммы передаются сервисному процессу, который удаленно занимается хранением, сопоставлением и сравнением контрольных сумм, соотвествующих обменным операциям MPI. При обна-

ружении различия значений контрольных сумм для одной пары приема/передачи MCCT выводит соответствующее диагностическое сообщение.

При таком подходе достигается минимальное воздействие на исследуемую программу, т.к. проверяемые данные и алгоритм работы программы не подвергаются изменению.

Для примера диагностики повреждения пересылаемых данных разработан следующий тест:

```
$ cat datacorruption.c
#include <string.h>
#include <mpi.h>
int main (int argc, char **argv)
{
 int rank;
 int buf[10];
 MPI_Request req;
 MPI_Status status;
 MPI_Init( &argc, &argv );
 MPI_Comm_rank( MPI_COMM_WORLD, &rank );
 if( rank == 0 ) {
 memset( buf, 1, 10 );
 MPI_Issend( &buf, 10, MPI_INT, 1, 100, MPI_COMM_WORLD, &req );
 buf[0] = 0;
 MPI_Barrier(MPI_COMM_WORLD);
 }
 if( rank == 1 ) {
 MPI_Barrier(MPI_COMM_WORLD);
 MPI_Recv( &buf, 10, MPI_INT, 0, 100, MPI_COMM_WORLD, &status );
 }
 MPI_Finalize( );
 return 0;
}
```

В данном случае применяется двухточечный обмен Issend/Recv. После инициализации запроса на отправку буфер с данными изменяется, а фактическая пересылка данных задерживается барьером. Таким образом, второй процесс принимает уже испорченные данные.

Диагностическое сообщение для такого примера приведено ниже:

```
$ checkrun -n 2 ./datacorruption

MPI CCT: ERROR: MPI_Recv [rank 1]:
MPI CCT: ERROR: Data corruption during transferring from rank 0 to rank 1
MPI CCT: ERROR: MPI_Recv [rank 1] called from:
MPI CCT: ERROR: #0 main@datacorruption.c:19
MPI CCT: ERROR: #1 __libc_start_main@/lib64/libc-2.12.so:(0x32a2e1ecdd)
MPI CCT: ERROR: #2 _start@/work/examples/datacorruption:(0x400809)

MPI CCT: INFO: Detected 1 errors and 0 warnings. More details is available in
the checker output file.
```

Как видно из примера, диагностическое сообщение содержит информацию о стеке вызовов, что позволяет точно указать на обнаруженное проблемное место MPI-приложения. Данная особенность MCCT использует API библиотеки Stackwalker из LGPL программного продукта Dyninst [5].

В итоге, представленная новая проверка значительно расширяет возможности утилиты MCCT 1.0, предоставляющей полноценный и гибкий комплекс автоматизированного поиска проблемных мест в программном коде сложных параллельных комплексов, что позволяет сократить время на их отладку и оптимизацию.

Можно также отметить, что утилита MCCT оптимизирована для работы в связке с MPI-библиотекой, входящей в состав программного комплекса S-MPI [2, 3].

Литература

1. Message Passing Interface Forum "MPI: A Message Passing Interface". – In Proceedings of Supercomputing ’93. IEEE Computer Society Press, November 1993. P. 878–883.
2. Огородников А.В., Побуринная Н.А., Старов М.И. Компонент проверки корректности использования MPI в программном комплексе S-MPI. – Вопросы атомной науки и техники, сер. «Математическое моделирование физических процессов», 2013, № 2, С. 86-94.
3. Воронов Г.И., Трущин В.Д., Шумилин В.В., Ежов Д.В. Программный комплекс S-MPI для обеспечения разработки, оптимизации и выполнения высокопараллельных приложений на суперкомпьютерных кластерных системах. – Вопросы атомной науки и техники, сер. «Математическое моделирование физических процессов», 2013, № 3, С. 55-60.
4. Hilbrich T., Schulz M., de Supinski B.R., Muller M. MUST: A Scalable Approach to Runtime Error Detection in MPI Programs. – In Proceedings of the 3rd Parallel Tools Workshop, Dresden, Germany, September 2009. P. 53-66.
5. Библиотека для работы со стеком вызовов Stackwalker. – [Электронный ресурс]: http://www.dyninst.org/stackwalker.

ФОРМАЛЬНАЯ СПЕЦИФИКАЦИЯ АСИНХРОННЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НАД ОБЩЕЙ ПАМЯТЬЮ

С.В. Панков

Южный федеральный университет, Ростов-на-Дону

Исследуется корректность асинхронных параллельных алгоритмов (всех его возможных вычислений над общей памятью) относительно заданных пред- и постусловий. Такая корректность гарантирует правильность результата для всех успешно завершающихся вычислений, началом которых является один из допустимых наборов входных данных. Демонстрируется подход к спецификации и обоснованию правильности асинхронных параллельных алгоритмов на основе формализма L-программ. Этот формализм включает в себя логико-математическую модель параллельных вычислений – L-программы (впервые описанные в [1]) и методы верификации L-программ [2].

Введение

Проблема обоснования функциональной корректности параллельных вычислительных алгоритмов является актуальной, особенно для систем, использующих асинхронные (неблокирующие) методы синхронизации параллельно исполняемых вычислительных процессов. В том числе это относится к перспективным системам с общей транзакционной памятью [3], в которых обеспечивается непротиворечивость результатов работы параллельных процессов. Семантика L-программ позволяет достаточно адекватно специфицировать вычисления в системах такого рода. Главной особенностью подхода на основе L-программ является его способность специфицировать и анализировать вычисления независимо от размера решаемой задачи [4]. Необходимо отметить, что здесь не затрагиваются такие свойства корректности, как отсутствие тупиков и завершаемость.

1. Основные понятия

Пусть $S=(Q,T)$ – система переходов, где Q – множество состояний, а $T\subseteq Q\times Q$ – отношение перехода на Q. Через $post[S](P)$ обозначим множество состояний непосредственно достижимых из $P\subseteq Q$, т.е. $post[S](P)=\{q\in Q \mid \exists q'\in Q((q',q)\in T \text{ и } q'\in P)\}$.

Множеством состояний, *достижимых из* $P\subseteq Q$ в системе переходов S за конечное число шагов, называется множество $post^*[S](P)= \cup_{i\geq 0}\ post^i[S](P)$, где $post^0[S](P)=P$, $post^{i+1}[S](P)=post[S](post^i[S](P))$.

Множество состояний $P\subseteq Q$ называется *инвариантом* системы переходов S, если для него справедливо $post[S](P)\subseteq P$ (войдя в инвариант, система S уже не сможет его покинуть). В соответствии с определением множества $post[S](P)$ справедливо следующее *условие инвариантности*: $\forall q',q\in Q\ ((q',q)\in T\ \&\ q'\in P)\Rightarrow q\in P)$.

Через $fin[S]$ обозначим множество *заключительных состояний* системы переходов S, из которых нет выхода, т.е. $fin[S]=\{q\in Q \mid \forall q'\in Q((q,q')\notin T)\}$. Пусть отношение $\{I\}S\{E\}$ выражает *частичную корректность* системы переходов S относительно предусловия $I\subseteq Q$ и постусловия $E\subseteq Q$. Это отношение определяется, как: $\{I\}S\{E\} \Leftrightarrow post^*[S](I)\cap fin[S]\subseteq E$ (т.е. все заключительные состояния, достижимые системой S из предусловия I, принадлежат постусловию E).

2. *L*-программы, как частный случай систем переходов

Множество *L*-программ определяется заданием многосортного языка *L* логики предикатов первого порядка (который также является языком некоторой предметной области). Через *Q* обозначим класс всех частичных (т.е. с частичными функциями) *L*-структур. *L*-программа действует на *Q* и работает по шагам, преобразуя одну *L*-структуру в другую. В отличие от обычных логических программ, *L*-программы могут произвольным образом переопределять значения функций и предикатов. Их работа аналогична прямому выводу в логических программах и допускает одновременное исполнение разных правил и различных конкретизаций одного и того же правила.

Синтаксис *L*-программ. *L-программа* задаётся конечной совокупностью *правил* вида $cond(\bar{x}) \Rightarrow act(\bar{x})$, где *cond* – условие, *act* – действие, $\bar{x}=<x_1,..,x_n>$ – кортеж переменных. В этом правиле $cond(\bar{x})$ – произвольная *L*-формула, $act(\bar{x})$ – формула вида $act_1(\bar{x})\wedge...\wedge act_N(\bar{x})$, где $act_i(\bar{x})$ – это *L*-формула вида $p(\bar{x}_p)$, $\neg p(\bar{x}_p)$ или $f(\bar{x}_f)=y$ (*p* – предикат, *f* – функция из *L*). Причём, кортежи переменных $\bar{x}_p$, $\bar{x}_f$ состоят из переменных кортежа $\bar{x}$, *y* – переменная того же сорта, что и значение функции *f*, и *y* также из $\bar{x}$. В *L* будем различать изменяемые и неизменяемые *L*-программой функции и предикаты. Под *изменяемыми* будем понимать функции и предикаты, входящие в действия правил *L*-программы.

Перед правилом или группой правил может стоять выражение вида **$** $\bar{z}$**, называемое *синхронизатором,* где z – кортеж переменных из числа переменных, входящих в условия этих правил. Группу правил с синхронизатором будем называть *синхрогруппой.*

Пример 1. В качестве примера рассмотрим задачу поиска наибольшего общего делителя произвольного числа *m* целых положительных чисел (где $m\geq 2$). Определим для этой задачи язык предметной области *L* и специфицирующую *L*-программу *НОД*. (Этот пример также предваряет следующее ниже строгое определение семантики *L*-программ).

Сорта: *Nat* – натуральные числа 1,2,...:

Loc – множество ячеек $\{b_1,..,b_m\}$, содержащих обрабатываемые числа.

Функции: s: $Loc\rightarrow Nat$ – тотальная функция, сопоставляющая каждой ячейке из *Loc*, содержащееся в ней значение:

_-_: $Nat\times Nat\rightarrow Nat$ – обычный арифметический минус.

Предикаты: _<_: $Nat\times Nat$. – обычное арифметическое отношение меньше.

Переменные: *u*,*v*: *Loc*, *h*:: *Nat*.

***L*-программа *НОД*:** $s(u)<s(v) \wedge h=s(v)-s(u) \Rightarrow s(v)=h$.

В соответствии с определяемой ниже семантикой *L*-программ, на каждом шаге работы *НОД* правило исполняется для некоторых пар ячеек *u* и *v*, для которых справедливо условие $s(u)<s(v)$. При этом, содержимое ячейки *v* (большее значение) заменяется на $s(v)-s(u)$ (разность большего и меньшего значений). Пары ячеек, с которыми это происходит одновременно, выбираются недетерминированно, но с таким условием, чтобы не делалось попытки записать в одну ячейку разные значения (чтобы не выполнялись противоречивые действия).

Синхронизатор уменьшает степень недетерминизма *L*-программы. Так, например, в *L*-программе *НОД* недетерминизм можно ограничить следующим образом: **$** *v*,*h* $s(u)<s(v) \wedge h=s(v)-s(u) \Rightarrow s(v)=h$. Благодаря синхронизатору **$** *v*,*h*, если на некотором шаге работы этой *L*-программы правило исполняется для пары ячеек $u=b_i$ и $v=b_j$ (где $1\leq i\leq m$, $1\leq j\leq m$), то на этом же шаге оно исполнится и для всех пар ячеек $u=b_i$ и $v=b_k$ (где $1\leq k\leq m$), для которых справедливо $s(b_i)<s(b_k)$. Одна ячейка с меньшим значением и все ячейки с большими (чем в первой) значениями, составляют группу пар ячеек, для которых одновременно исполняется правило. Недетерминированность остаётся на уровне

выбора ячейки с меньшим значением. Одновременное исполнение правила для нескольких таких групп пар ячеек невозможно, когда содержимое ячеек с меньшими значениями для этих групп не совпадает (из-за наличия в этом случае противоречивых действий).

Операционная семантика *L*-программ. Определим операционную семантику *L*-программ, но сначала введем некоторые понятия и обозначения. Пусть q – *L*-структуры из Q, C_q – множество констант для обозначения объектов *L*-структуры q. Пусть $r(\bar{x})$ обозначает правило *L*-программы $cond(\bar{x}) \Rightarrow act(\bar{x})$. *Ансамблем* на q, порождаемым правилом $r(\bar{x})$, назовём произвольную конкретизацию этого правила на q, т.е. запись вида $r(\bar{c})$, где $\bar{c}=<c_1,..,c_n>$ ($c_i \in C_q$ для всех $1 \le i \le n$). Ансамбль $r(\bar{c})$ называется *активным* на q, если на q истинна формула $cond(\bar{c})$.

Пусть правило $r(\bar{z},\bar{v})$ входит в группу с синхронизатором **$** $\bar{z}$ **и** на q активен ансамбль $r(\bar{a},\bar{c})$, где $\bar{a}$ и $\bar{c}$ – конкретизация соответствующих наборов переменных $\bar{z}$ и $\bar{v}$. *Синхрорасширением* активного ансамбля $r(\bar{a},\bar{c})$ назовём множество ансамблей $\{r(\bar{a}', \bar{c}) \mid r(\bar{a}', \bar{c})$ активен на $q\}$, где $\bar{a}'$ – произвольная конкретизация набора переменных $\bar{z}$.

Атомарными действиями назовём конъюнктивные члены формулы $act(\bar{c})$, т.е. формулы вида $p(\bar{c}_p)$, $p(\bar{c}_p)$, или $f(\bar{c}_f)=b$ (здесь $\bar{c}_p$, $\bar{c}_f$, и b – конкретизация набора переменных $\bar{x}_p$, $\bar{x}_f$, и y соответственно). Активность ансамбля $cond(\bar{c}) \Rightarrow act(\bar{c})$ может привести к одновременному выполнению всех его атомарных действий. При *выполнении на q* атомарное действие $p(\bar{c}_p)$ устанавливает истинным p на q, $p(\bar{c}_p)$ устанавливает ложным p на q, а $f(\bar{c}_f)=b$ устанавливает значение f на $\bar{c}_f$ равн**ым** b.

Коалицией на q будем называть произвольное непустое множество активных ансамблей M, удовлетворяющее следующим трём условиям:

Условие непротиворечивости: объединение всех атомарных действий коалиции образует непротиворечивую совокупность формул.

Первое условие синхронизации: если в M входит ансамбль, порожденный некоторым правилом синхрогруппы, то и для любого другого правила этой синхрогруппы в M должен входить, хотя бы один, порожденный этим правилом, активный ансамбль (если такой существует).

Второе условие синхронизации: если в M входит ансамбль, порожденный правилом синхрогруппы, то в M также должно входить всё его синхрорасширение.

Исполнение коалиции состоит в выполнении всех её атомарных действий. *Шаг работы L*-программы на заданной *L*-структуре состоит в исполнении на ней некоторой коалиции. При этом, коалиция выбирается недетерминировано. Если на q нет ни одной коалиции, *L*-программа завершает свою работу в q. Таким образом, *L*-программа задаёт систему переходов $S=(Q,T)$, где Q – произвольное множество *L*-структур, замкнутое относительно шага работы *L*-программы, а отношение перехода $T=\{(q,q') \mid q,q' \in Q$ и q' получается из q исполнением некоторой коалиции$\}$.

3. Анализ функциональной корректности L-программ

Частичная корректность системы переходов S относительно предусловия $I \subseteq Q$ и постусловия $E \subseteq Q$ характеризует правильность системы с точки зрения вычисляемой ею функции. Допустимые входные данные функции описываются предусловием I, а соответствующий этим данным результат вычисления функции должен удовлетворять требованиям постусловия E. Установить $\{I\}S\{E\}$ можно путём проверки выполнения следующих *условий корректности* [2]: Существует такое множество состояний $W \subseteq Q$, что справедливо:

1. $I \subseteq W$ (условие инициализации);
2. $\forall q',q \in Q\ ((q',q) \in T\ \&\ q' \in W) \Rightarrow q \in W)$ (условие инвариантности W);

3. W∩fin[S]⊆E (условие завершения).

Алгоритмы из [2] позволяют получить логические формулы, выражающие в языке L отношение перехода T, множества $post[S](P)$ и $fin[S]$ для L-программ. Если множества I, W и E также выражаются L-формулами, то анализ функциональной корректности сводится к доказательству истинности формул логики предикатов первого порядка.

Пример 2. Обоснуем правильность *НОД* относительно заданных пред- и постусловий. Пусть определённый в примере 1 язык L также содержит: все обычные арифметические функции и отношения; предикат *_делит_*: $Nat\times Nat$ – выражающий тот факт, что первый аргумент является делителем второго; функция s': $Loc\to Nat$ – имеет тот же смысл, что и функция s, только представляет содержимое ячеек из Loc в L-структуре q'; переменная $\boldsymbol{d}$ сорта Nat. Знаками $\wedge$, $\vee$, $\neg$, $\to$, $\forall$ и $\exists$ будем обозначать обычные логические связки и кванторы.

Пусть $W(\mathbf{D})$ обозначает формулу, выражающую тот факт, что константа **D** (сорта Nat) является наибольшим общим делителем (н.о.д.) чисел $s(b_1),...,s(b_m)$

$$W(\mathbf{D})\colon \forall u(\boldsymbol{D}\ \textit{делит}\ s(u)) \wedge \forall \boldsymbol{d}(\forall u(\boldsymbol{d}\ \textit{делит}\ s(u)) \to \boldsymbol{d}\le\mathbf{D}).$$

Эта формула утверждает, что **D** является делителем чисел $s(b_1),...,s(b_m)$ и для любого другого делителя $\boldsymbol{d}$ этих чисел справедливо $\boldsymbol{d}\le\mathbf{D}$. Формулу $W(\mathbf{D})$ будем рассматривать как кандидат в инварианты, а также примем её в качестве предусловия I. В качестве постусловия E выберем формулу $\forall u(s(u)=\mathbf{D})$, требующую, чтобы содержимое каждой ячейки совпадало с н.о.д.

Формула FIN, выражающая множество заключительных состояний $fin[S]$, утверждает, что условие правила L-программы *НОД* должно быть ложно для любой конкретизации её свободных переменных, и в соответствии с алгоритмом из [2], строится, как: $\forall u,v,h\neg(s(u)<s(v) \wedge h=s(v)-s(u))$. С учётом тотальности s на Q эта формула эквивалентна следующей формуле:

$$FIN\colon \forall u,v(s(u)=s(v)),$$

которая говорит о том, что содержимое всех ячеек должно совпадать. Формула $T(s',s)$, описывающая связь L-структур q' и q таких, что $(q',q)\in T$ (выражающая отношение перехода), в соответствии с алгоритмом из [2], имеет следующий вид:

$$T(s',s)\colon \forall v(s(v)=s'(v) \vee \exists u(s'(u)<s'(v) \wedge s(v)=s'(v)-s'(u))).$$

Эта формула утверждает, что при переходе из q' в q содержимое ячейки, либо не изменилось, либо уменьшилось на значение некоторой другой ячейки, содержимое которой меньше значения текущей ячейки. Помимо приведённых формул, будем использовать следующее ниже свойство рассматриваемой предметной области:

$$\forall d,e,e_1,..,e_k\in Nat(e=e_1\pm\ldots\pm e_k \wedge d\ \textit{делит}\ e_i,\ 1\le i\le k \Rightarrow d\ \textit{делит}\ e. \qquad (1)$$

Доказательство. Пусть $W<s'>(\mathbf{D})$ – это формула $W(\mathbf{D})$, в которой вхождения функции s заменены на s'. В соответствии с приведёнными выше теоретико-множественными условиями корректности, для обоснования $\{I\}НОД\{E\}$ достаточно доказать истинность следующих L-формул:

1. $I\to W(\mathbf{D})$ (условие инициализации *НОД*);
2. $T(s',s)\wedge W<s'>(\mathbf{D})\to W(\mathbf{D})$ (условие инвариантности $W(\mathbf{D})$ для *НОД*);
3. $W(\mathbf{D})\wedge FIN\to E$ (условие завершения *НОД*).

Условие инициализации истинно, т.к. I совпадает с $W(\mathbf{D})$. Условие инвариантности утверждает следующее: если функции s' и s связаны так, как это описывается формулой $T(s',s)$, и **D** – н.о.д. чисел $s'(b_1),...,s'(b_m)$, то **D** – н.о.д. чисел $s(b_1),...,s(b_m)$. Доказательство истинности условия инвариантности разобьём на две части.

1. Сначала докажем истинность формулы $T(s',s)\wedge W<s'>(\mathbf{D})\to\forall u(\boldsymbol{D}\ \textit{делит}\ s(u))$. Из $T(s',s)$ следует, что для произвольного $b_i\in Loc$ либо $s(b_i)=s'(b_i)$, либо $s(b_i)=s'(b_i)-s'(b_j)$ для некоторого $b_j\in Loc$. Из $W<s'>(\mathbf{D})$ следует $\forall u(\boldsymbol{D}\ \textit{делит}\ s'(u))$, тогда по свойству

(1) справедливо $\boldsymbol{D}$ *делит* $s(b_i)$. В силу произвольности b_i, истинна формула $\forall u(\boldsymbol{D}$ *делит* $s(u))$.

2. Докажем истинность формулы $T(s',s)\wedge W\langle s'\rangle(\mathbf{D}) \rightarrow \forall \boldsymbol{d}(\forall u(\boldsymbol{d}$ *делит* $s(u))\rightarrow \boldsymbol{d}\leq \mathbf{D})$. Она эквивалентна формуле $\exists \boldsymbol{d}(\forall u(\boldsymbol{d}$ *делит* $s(u))\wedge \boldsymbol{d}>\mathbf{D})\wedge T(s',s) \rightarrow \neg W\langle s'\rangle(\mathbf{D})$. Из $T(s',s)$ следует, что для произвольного $b_{i0}\in Loc$ либо $s'(b_{i0})=s(b_{i0})$, либо $s'(b_{i0})=s(b_{i0})+s'(b_{i1})$ для некоторого $b_{i1}\in Loc$ такого, что $s'(b_{i1})<s'(b_{i0})$. Во втором варианте $s'(b_{i1})$ выражается по формуле $T(s',s)$ через функцию s аналогично $s'(b_{i0})$ и т.д., а в силу конечности Loc существует n такое, что $s'(b_{i0})=s(b_{i0})+s(b_{i1})+...+s(b_{in})$. Тогда в силу свойства (1) и произвольности b_{i0} из формулы $\exists \boldsymbol{d}(\forall u(\boldsymbol{d}$ *делит* $s(u))\wedge \boldsymbol{d}>\mathbf{D})$ следует истинность формулы $\exists \boldsymbol{d}(\forall u(\boldsymbol{d}$ *делит* $s'(u))\wedge \boldsymbol{d}>\mathbf{D})$. Тогда истинна формула $W\langle s'\rangle(\mathbf{D})$, т.к она эквивалентна формуле $\exists u$ $(\boldsymbol{D}$ *делит* $s'(u)) \vee \exists \boldsymbol{d}(\forall u(\boldsymbol{d}$ *делит* $s''(u))\wedge \boldsymbol{d}>\mathbf{D})$. Таким образом, истинность условия инвариантности доказана.

Истинность условия завершения $W(\mathbf{D})\wedge\forall u,v(s(u)=s(v))\rightarrow\forall u(s(u)=\mathbf{D})$ по свойству н.о.д. очевидна. Тем самым доказана частичная корректность L-программы *НОД*.

Литература

1. Крицкий С.П. Модель асинхронных вычислений в структурах и языки программирования // Методы трансляции. Ростов-на-Дону, 1981. С. 92-100.
2. Крицкий С.П., Панков С.В. О верификации асинхронных программ продукционного типа // Программирование. № 5, 1994, С. 40-52.
3. Grahn H. Transactional Memory // J. Parallel Distrib. Comput. 2010. Vol. 70 (10). P. 993-1008.
4. Панков С.В. Логический подход к спецификации и анализу поведения клеточ-ных автоматов // Труды Всероссийской научной конференции «Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ», Новороссийск, 19-24 сентября 2007 г. Изд-во Московского университета. С. 85–88.

РАСПРЕДЕЛЕННАЯ МАШИНА ВЫВОДА ОБЪЕКТНО-СОБЫТИЙНЫХ МОДЕЛЕЙ

В.В. Пекунов

ОАО «Информатика», Иваново
E-mail: pekunov@mail.ru

Данная работа является развитием работы [1]. Ее целью является ввод понятия об инкрементальном логическом выводе объектно-событийных моделей (ОСМ), который может быть настолько трудоемок, что требует распределенных вычислений. Логический вывод применяется, в частности, для автоматизированного построения моделей, порождающих решающие программы, по формальной постановке проблемы.

Теорема об обратном логическом выводе (ТЛ1). Задача обратного логического вывода для произвольного целевого выражения, включающего единичные высказывания (предикаты), а также связки «И» и «ИЛИ», может быть реализована ОСМ, структура и параметры которой отражают форму такого целевого утверждения, если

а) единичному высказыванию сопоставить единственный узел ОСМ;

б) между узлами-высказываниями, подписанными на одно событие – связка «И»;

в) наличие связки «ИЛИ» может быть определено параметром некоего узла А, например, для исходящих из него параллельных ветвей;

г) существует предельная (атомарная) логическая ОСМ, способная для единичного предиката решить задачу унификации с единственным фактом.

Доказательство.

1. Докажем, что целевое высказывание может быть описано структурой и параметрами ОСМ. Пункт (а) формулировки доказуем, если узлы-объекты ОСМ относятся к соответствующим классам-предикатам. Пункт (б) следует из идеологии сетевого графика работ (одного из основных принципов ОСМ). Узлы-высказывания, между которыми существует отношение следования, в ходе одного события доказываются последовательно, при его отсутствии *возможно параллельное доказательство*. Рассмотрим пункт (в). В этом случае узел А подписывает на текущее событие одну произвольную ветвь, а в случае необходимости пересогласования – какую-либо из иных ветвей, в соответствии с алгоритмом обратного логического вывода (ОЛВ).

2. Процесс ОЛВ имеет процедурную трактовку, алгоритм которой может быть реализован ОСМ. Это подтверждается существованием трассы у машины ОЛВ, для которой легко проиллюстрировать возможность построения эквивалентной ОСМ. При этом задача доказательства сложных выражений сводится к сериям задач доказательства более простых выражений на вложенных ОСМ-методах с соответствующим управлением процессом доказательства, реализуемым по событийной схеме. Заключительными элементарными этапами такого алгоритма будут вызовы процедур (методов классов объектов-узлов), решающих задачи унификации единичного высказывания с единичным фактом и сводимые к ним задачи унификации термов. В роли таких методов будут выступать предельные логические ОСМ. Таким образом, условия, введенного пунктом (г) формулировки, необходимо и достаточно для завершения доказательства.

Следствие. Задача отката с генерацией следующих вариантов доказательства в предыдущих, включенных в цель предикатах, может быть тривиально реализована путем условного планирования события – следующей попытки доказательства целевого

утверждения, на которое будут подписаны подлежащие пересогласованию элементы цели. Текущая позиция вывода может храниться в полях объектов-узлов ОСМ.

Теорема о предельной логической ОСМ (ТП5). Предельная (атомарная) логическая объектно-событийная модель, решающая задачу унификации единственного предиката с одним фактом, тождественна абстрактной предельной ОСМ.

Доказательство. Абстрактная предельная ОСМ однозначно отображается в машину Тьюринга (МТ) и включает единственный узел. Набор возможных значений V_i каждого *i*-го аргумента предиката ($i = \overline{1,N}$) в реальной вычислительной системе в сочетании со значением $\varnothing$, указывающим на несвязанность аргумента, является конечным множеством $P_i = V_i \cup \{\varnothing\}$.

Декартово произведение множеств P_i в сочетании с символом nil («ложь») представляет собой алфавит МТ: $Q = (P_1 \times P_2 \times \ldots \times P_N) \cup \{nil\}$. Один из символов $p \in Q$ соответствует комплексу значений аргументов, заданных фактом. На ленту помещается символ, соответствующий значениям входных аргументов предиката, в ту же ячейку помещается символ p при успешной унификации или nil, если таковая невозможна.

Пусть в начале доказательства календарь содержит начальное событие q_1. Пусть также возможно конечное событие q_2.

Легко составить программу для единственного метода-обработчика события q_1, включающего команды условного планирования (переходы МТ) события q_2 с помещением на ленту символа p, соответствующего успешно унифицируемым значениям аргументов (символам входного алфавита МТ), или символа nil для таких значений аргументов, которые не могут быть унифицированы. Программа вычислима по Тьюрингу.

Следствие. Также доказано, что выполняется условие, содержащееся в ТЛ1.

Теорема об унификации ОСМ (ТМ7). Отобразим ОСМ в совокупность фактов о ее структуре и параметрах. Для унификации ОСМ знаниям об отображении возможных вариантов решения задач в структуру и параметры ОСМ, необходимо и достаточно инкрементально доказать серию логических утверждений, следующих из текущих структуры и параметров ОСМ, до полного согласования последних с фактами, изъятыми и введенными в базу знаний о модели в процессе доказательства.

Доказательство. Согласно ТЛ1, если отдельным объектам модели соответствуют некоторые логические утверждения, то их совокупность представляет некоторое целевое утверждение, которое может быть доказано или опровергнуто путем обратного логического вывода. Этот процесс может сопровождаться включением/изъятием фактов из базы знаний, в том числе фактов о структуре/параметрах модели. Если база знаний не изменилась, то она полностью согласована с текущей ОСМ. Если база знаний изменилась, то это может означать возникновение нового целевого утверждения, следующего из модифицированной модели, которое также нуждается в доказательстве. Даже если целевое утверждение не изменилось, оно нуждается в повторном рассмотрении во избежание противоречия с изменившимся содержанием базы фактов. Таким образом процесс согласования подразумевает инкрементальное (многоэтапное) доказательство динамически генерируемого целевого утверждения, следующего из структуры/параметров модели. Теорема доказана.

Теорема о когнитивном характере ОСМ (ТК1). В процессе построения ОСМ функционирует как когнитивная система.

Доказательство. По определению система когнитивной архитектуры подразумевает многоэтапное рассуждение на основе фактов, находящихся в данный момент в рабочей области, в процессе которого в такую область рассуждения включаются новые факты и/или изымаются уже имеющиеся. Такой процесс может продолжаться до полного согласования системы рассуждений с базой имеющихся и выведенных фактов.

Утверждение об обратной связи (ТМ8). Результат исполнения как конечной ОСМ, так и ОСМ, являющихся промежуточными этапами ее унификации, может содержать новые факты, нуждающиеся в согласовании со структурой/параметрами ОСМ.

Доказательство. Утверждение согласуется с теоремой о когнитивном характере процессов в ОСМ (ТК1).

Следствие. Необходима формальная машина вывода ОСМ, позволяющая поэтапно унифицировать ОСМ, планировать исполнение конечных и промежуточных ОСМ, исполнять их, обобщать и *при необходимости* согласовывать результаты.

Теорема о распределенной машине вывода ОСМ (ТМ9). Условиям, указанным следствием из ТМ8, удовлетворяет система T = (S, Q, I, A, M, P), включающая

а) стек S троек (C, B, F), где C – ОСМ или машина вывода, B – локальная база фактов ОСМ или пустое множество для машины, $F \in \{0,1\}$ – логический признак готовности к исполнению,

б) очередь Q асинхронно или синхронно запущенных машин,

в) очередь команд I, для которой операции помещения и извлечения являются атомарными,

г) рабочую область A,

д) активную систему M = (C, B, F),

е) ссылку P на машину-родителя.

Машина должна обладать следующими свойствами:

А. Способность выполнить следующие команды:

mov – копировать активную систему M в A.

push – переместить систему из A на вершину стека S со статусом «не готова к исполнению». Выполнение данной операции не запускает процесс унификации для указанной системы.

epush – выполняется аналогично **push**, системе присваивается статус «готова к исполнению».

pop – взять систему с вершины стека S в A.

fork – создать новую машину вывода и поместить ее в A. При этом с вершины стека S текущей машины снимается система и перемещается в стек созданной машины. В отличие от обычной модели, машина вывода может быть отправлена на исполнение на иное вычислительное устройство, что обеспечивает возможность работы в распределенном режиме.

assert(факт) – поместить факт в рабочую область A машины.

retract(факт) – удалить факт из рабочей области A машины.

consult – пополнить базу фактов ОСМ, находящейся в M, базой фактов из рабочей области A, которая при этом очищается. Если A изначально пуста, база фактов не изменяется.

Б. Алгоритм вывода:

1. Если M пуста, то

1.1. Если стек S пуст, то ожидаем очистки очереди Q (то есть завершения работы запущенных машин), в процессе ожидания обрабатываем команды, поступающие в очередь I. Затем машина заканчивает работу.

1.2. Если стек S не пуст, то система с вершины стека перемещается в M.

2. Если в M присутствуют данные, требующие обработки, то переходим к пункту 3. Иначе очищаем M и переходим к пункту 1.

К данным, требующим обработки, относятся: а) ОСМ, требующая унификации, б) иная машина вывода.

3. Если содержимое M относится к классу (б), то это иная машина вывода. Тогда:

3.1. Машина вывода из M помещается в очередь Q, где либо *асинхронно или синхронно* запускается на исполнение на свободном вычислительном устройстве, либо синхронно запускается на текущем, если свободных устройств нет. В процессе исполнения модели, обрабатываемые запущенной машиной, имеют возможность, помещая **assert** в P.I, включать факты в рабочую область A текущей машины. Машина удаляется из очереди тогда и только тогда, когда закончит работу.

3.2. Очищаем M и переходим к пункту 1.

4. Ожидаем очистки очереди Q, обрабатывая команды, поступающие в очередь I.

5. Если содержимое M относится к классу (a), то это тройка (ОСМ, БФ, ГОТ). Тогда выполняются следующие действия:

5.1. Унификация ОСМ и далее, если ГОТ=1, исполнение. В процессе работы ОСМ может помещать в очередь I текущей машины синхронно исполняемые команды.

5.2. Если в результате исполнения изменилась БФ, переходим к пункту 5.1. Иначе очищаем M и переходим к пункту 1.

Доказательство. Алгоритм содержит цикл 5.1-5.2, обеспечивающий поэтапную унификацию ОСМ, ее исполнение с последующим согласованием структуры и параметров ОСМ с фактами, полученными в ходе исполнения. Для реализации данного такта машине вывода необходим компонент M, а также возможность выполнить команды **assert, consult** и, возможно, **retract**, требующие наличия аккумулятора A. Команда **consult** не изменяет базу фактов при пустом A, это необходимое условие, предотвращающее зацикливание системы вывода. Таким образом, в части, относящейся к начальной ОСМ и результату ее унификации, условия следствия из ТМ8 выполняются.

Цикл 1-5 обеспечивает обработку моделей, непосредственно поступивших в стек текущей машины вывода, а также запуск динамически порожденных машин для обработки переданных им моделей. Этим обосновывается наличие стека S. Таким образом, поскольку существуют команды **mov** и **push/epush**, в стек могут быть помещены ОСМ, являющиеся промежуточными этапами унификации. В отличие от **push**, которая помещает в стек модель, ориентируя ее исключительно на унификацию, команда **epush** помимо унификации предполагает исполнение модели, что позволяет учесть соответствующие требования ТМ8. Команда **pop** позволяет при необходимости избежать исполнения некоторых ОСМ и машин вывода и учета соответствующих результатов. Команды **mov**, **push/epush** и **fork** позволяют поместить промежуточные ОСМ в динамически порождаемые машины вывода для запуска и обработки в своем физическом/логическом контексте (в том числе на ином вычислительном устройстве). Для хранения порождаемых машин была введена очередь Q, причем механизм очереди позволяет установить приоритеты их исполнения. Для реализации доступа из дочерних машин к данным родительской в систему вывода введена ссылка P. Задача удовлетворения условиям следствия из ТМ8 для моделей, переданных в дочерние машины, рассматриваемых независимо от текущей машины, сводится к исходной. Команды **assert/retract**, **consult** обеспечивают (*в случае такой необходимости*) взаимодействие с порожденными в ходе их работы базами фактов, что окончательно удовлетворяет условиям следствия из ТМ8 для текущей машины, поскольку в необходимых случаях обеспечивает согласование с результатами исполнения любых промежуточных ОСМ, сгенерированных самой машиной или любой порожденной ею машиной.

Команды, поступающие в машину, ставятся в очередь I и исполняются синхронно, в соответствии с принципом FIFO (First In First Out). Необходимость в наличии очереди объясняется требованием отсутствия коллизий при одновременном поступлении в машину нескольких команд от дочерних машин. Режим ожидания, реализованный пунктом 4 алгоритма, исключает одновременную работу родительской и дочерних машин,

предотвращая потенциальные коллизии поступления с их сторон команд работы с одними и теми же областями данных (в первую очередь – с аккумулятором).

Следствие. При асинхронном запуске машин из очереди Q имеем параллельный логический вывод, при синхронном – последовательный.

Рассмотрим реализацию машины вывода ОСМ в системе порождения программ PGEN++. В соответствии с ТМ9 машина вывода поддерживает порождение новых машин вывода для асинхронного решения подзадач. Эти машины содержатся в очереди Q родительской машины и могут либо последовательно исполняться на одном компьютере, либо передаваться на параллельную обработку на иные компьютеры. Таким образом, архитектура системы должна быть распределенной. Была выбрана схема, в которой любой из экземпляров системы порождения программ может выполнять как функции сервера, обеспечивающего взаимодействие с пользователем и распределение вычислительных ресурсов, так и функции клиента (исполнителя порожденных заданий).

Роль конкретного экземпляра системы определяется динамически, по факту очередности запуска: первый экземпляр берет на себя функции сервера, остальные – клиентов. Основная часть заданий обычно генерируется сервером, однако возможны и ситуации, когда в процессе исполнения заданий на клиенте генерируется новая их серия (имеется в виду случай, когда дочерняя машина вывода породила еще одну или несколько машин). В таком случае клиентские экземпляры системы порождения берут на себя часть серверных функций (засылка задания со всеми необходимыми файлами и ожидание результата его обработки) по отношению к порожденным заданиям, за исключением распределения вычислительных ресурсов – им занимается лишь серверный экземпляр системы. Он и предоставляет (по поступающим запросам) клиентские экземпляры системы в качестве исполнителя порожденных машин вывода.

Данная идеология работы определила и механизмы реализации управляющей выводом части системы порождения программ. Каждый экземпляр системы является многопоточным приложением, в котором *постоянно работает не менее трех потоков*: основного, управляющего и очереди заданий.

Основной поток содержит машину вывода и генерирует для сервера сообщение об окончании работы машины (об освобождении узла системы) и об окончании исполнения задания.

Управляющий поток реализует обработку запросов, поступающих от других экземпляров системы. В начале работы он посылает групповой запрос (IP Multicast) узлам локальной сети, чтобы определить, является ли текущий экземпляр первым из запущенных, то есть серверным, или же клиентским. Далее поток переходит в режим генерации и обработки запросов по сети, преимущественно касающихся занятости узлов и их выделения. Он также принимает задания от серверного экземпляра системы.

Поток, реализующий очередь заданий, также запускается при старте системы. Он отслеживает занятость текущего экземпляра системы заданием, сообщая серверу об его освобождении, ведет список генерируемых в системе заданий, запрашивает свободные узлы (экземпляры системы) для отсылки задания и ведет их список (это относится в первую очередь к серверу), при появлении свободного узла отсылает ему задание.

При отсылке каждого задания для него также порождается *отдельный поток*, который устанавливает связь с удаленным компьютером, на котором обрабатывается задание, и переходит в режим ожидания результатов.

Итак, в данной работе дано понятие о логическом выводе ОСМ, предложена распределенная машина вывода ОСМ, определена технология ее реализации на комплексе ЭВМ, особенно эффективная в случае применения многоядерных процессоров.

Литература

1. Пекунов В.В. Теория объектно-событийного моделирования последовательных и параллельных процессов. Следствия и приложения в программировании, моделировании и порождении программ // Сб. матер. XI Всеросс. конф. «Высокопроизводительные параллельные вычисления на кластерных системах». – Нижний Новгород: Изд-во ННГУ, 2011. – С.238-243.

ПРОГРАММНЫЙ КОМПЛЕКС «ВИРТУАЛЬНОЕ СЕРДЦЕ»

В.С. Петров, Г.В. Осипов, А.К. Крюков

Нижегородский госуниверситет им Н.И. Лобачевского

Приводится описание и результаты использования программного комплекса «Виртуальное сердце». Данный комплекс решает задачу моделирования целого сердца в 3D с учетом реальной геометрии сердца. Описываются подходы и методы, использованные при решении данной задачи. Представлены результаты моделирования, а также характеристики производительности.

Введение

Изучение механизмов развития различного рода аритмий, разработка методов их диагностики и способов их предотвращения и лечения являются сейчас исключительно важными вследствие того, что в экономически развитых странах сердечнососудистые заболевания являются основной причиной смертности.

Интенсивные исследования сердечных аритмий ведут медики и биологи, физики и специалисты в области математического моделирования. Так как сердце является динамической системой, происходящие в нем процессы могут быть описаны как эволюция некоторых переменных состояний: электрических мембранных потенциалов, проводимостей ионных каналов, ионных токов, то описание его работы можно получить, анализируя соответствующие математические модели. Сейчас наступает новый этап исследований сердечной деятельности, обусловленный наличием двух факторов: существованием достаточно реалистических моделей сердечной мышцы, базирующихся на новейших данных электрофизиологии, и доступностью высокопроизводительной вычислительной техники, позволяющей эффективно выполнять расчеты при моделировании сложных пространственно-временных процессов в сердце. Целью настоящего проекта являлось создание программного комплекса «Виртуальное сердце», предназначенного для моделирования сердечной активности. Такой комплекс создан. Его можно использовать в научных исследованиях для проведения крупномасштабных вычислительных экспериментов по анализу пространственно-временной динамики в сердце. Комплекс позволяет рассчитывать как текущее состояние сердечной активности, его эволюцию, так и интегральные характеристики, например, виртуальную электрокардиограмму (ВЭКГ).

1. Структура комплекса

Для использования программного комплекса «Виртуальное сердце» требуются специально обработанные препроцессорам данные МРТ. На основе этих данных проводится выделение сердца, построение сетки и сегментация сердца на основные составные части: желудочки и предсердия, синоатриальный и атриовентрикулярный узлы, пучок Гиса и волокна Пуркинье, фибробласты и др.

Программный комплекс «Виртуальное сердце» реализует следующую основную функциональность:

- препроцессор – предварительная обработка и сегментация данных МРТ;
- чтение геометрической конечно-элементной модели сердца, полученной на основе данных МРТ;

- ассемблирование конечно-элементных матриц масс и жесткости;
- решение конечно-элементных линейных уравнений;
- решение нелинейных обыкновенных дифференциальных уравнений, описывающих электрофизиологию;
- экспорт конечно-элементных матриц в заданном формате;
- экспорт результатов проведенных вычислений в формате VTK;
- визуализация данных и расчет ЭКГ.

2. Препроцессор

В качестве исходных данных для работы программного комплекса «Виртуальное сердце» используются данные МРТ. Трехмерная визуализация данных МРТ проводилась с помощью алгоритма «Marching Cubes» с незначительными изменениями. Пользователю предоставляется возможность с помощью мыши и клавиатуры на трехмерной сцене редактировать данные – удалять ненужные участки томограммы, а также сегментировать ткани сердца. Сегментирование проводится на шесть основных областей сердца: желудочки, предсердия, непроводящая ткань, АВ узел, синусовый узел, область, соответствующая пучкам Гиса.

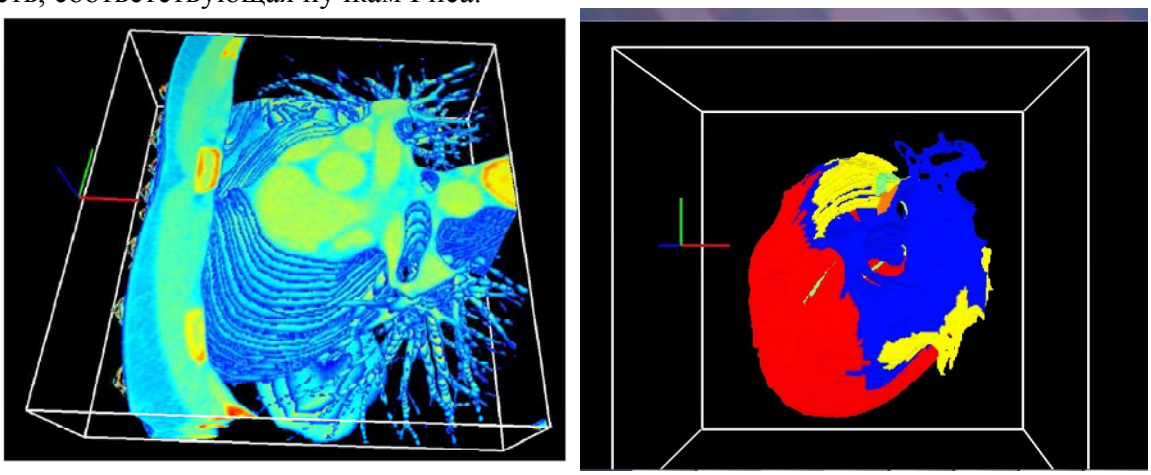


Рис. 1. Препроцессор программного комплекса. Слева: визуализация данных МРТ сердца без предварительной обработки. Справа: данные после очистки и сегментации

Из полученных результатов консольного генератора строим воксельную модель сердца. Воксельная модель - это трехмерный растр, где каждый воксел - элемент объема. Любой воксел имеет также свой цвет и прозрачность. Полная прозрачность воксела означает пустоту соответствующей точки объема. Поскольку вокселы располагаются в узлах сетки, то, обычно, чем меньше шаг сетки, тем больше количество вокселов помещается в определенном объеме, и тем меньше размер каждого отдельного воксела. Основная характеристика воксельной модели - разрешающая способность - количество вокселов в определенном объеме. Она и определяет точность моделирования трехмерного сердца. К сформированной воксельной модели сердца применяется генерация тетраэдральной сетки, основанная на методе триангуляции Делоне. Реализация ядра генератора тетраэдральной сетки выполнена на основе пакета библиотек CGAL (Computational Geometry Algorithms Library) с отрытым исходным кодом, которые обеспечивают легкий доступ к эффективным и надежным геометрическим алгоритмам в виде библиотек C++.

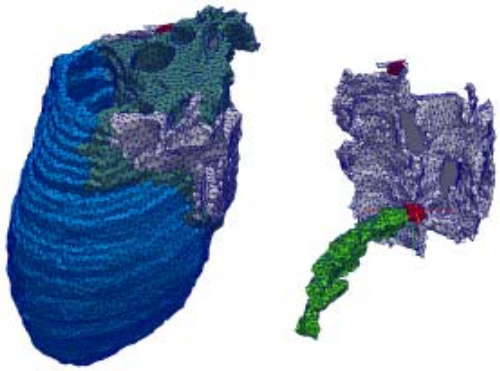


Рис. 2. Конечно-элементная сетка, получаемая в результате триангуляции

3. Блок решения СЛАУ

В процессе моделирования сердечной активности необходимо решать задачу Пуассона вида

$$\nabla U^2 = f. \tag{1}$$

С использованием метода конечных элементов данная задача сводится к решению системы алгебраических уравнений. Примечательно, что данная системы получается сильно разреженной, что позволяет использовать эффективные солверы. Реализация прямого решателя была выполнена в ННГУ им. Н.И. Лобачевского и используется в комплексе «Виртуальное сердце» посредством предоставляемых высокоуровневых API. В основе реализации лежит факторизация Холецкого (которую можно выполнять один раз на этапе инициализации) с последующими вызовами обратного хода. Также реализован альтернативный параллельный солвер с использованием библиотеки PETSc. В последнем случае комплекс запускается на кластере с использованием MPI.

4. Блок решения ОДУ

Данный модуль реализует схему численного интегрирования систем нелинейных обыкновенных уравнений. В качестве данных систем выступают наиболее точные и биологически релевантные модели сердечных клеток (Grandi_Pasqualini_Bers_2010, Luo_Rudy_phase_I_1991, TenTusscher_Panfilov_Noble_2006). Решение всей системы уравнений целого сердца также выполняется параллельно, как на кластерных системах с использованием MPI, так и на машинах с общей памятью с использованием TBB.

5. Визуализация

Программный комплекс «Виртуальное сердце» предусматривает ряд возможностей по визуализации результатов выполняемых расчетов. Во-первых, пользователь комплекса может регулировать частоту сохранения данных на диск, таким образом файл результата будет содержать различное число моментальных снимков моделируемого процесса. Во-вторых, программный комплекс автоматически обрабатывает и конвертирует данные в открытый формат VTK. С данным форматом данных можно работать как самостоятельно, так и использованием многочисленного открытого программного обеспечения, например Paraview. Таким образом, у пользователя есть возможность просматривать результаты вычислений в 3D, и при этом остается свобода выбора наиболее удобного визуализатора. В-третьих, в графическом интерфейсе программного комплекса «Виртуальное сердце» имеется блок обработки и визуализации вычисляемой виртуальной электрокардиограммы. Этот блок способен считывать файл результата, содержащий ВЭКГ, вычислять на его основе кривую и визуализировать ее. При этом пользователь имеет возможность менять «на лету» направление вектора проекции ЭКГ, что позволяет наиболее удобно проанализировать процесс.

6. Результаты моделирования

В данном разделе представлены результаты 3D-моделирования сердечной активности изолированного сердца в норме и при аритмиях, полученные с помощью программного комплекса «Виртуальное сердце».

Продемонстрируем сначала результаты вычислений нормальной сердечной активности. Далее представлены мгновенные снимки, сделанные в различные моменты времени. Рассматривались стандартные значения параметров, при которых распространяются псевдо-концентрические волны возбуждений. С течением они постепенно охватывают практически все сердце. Полученная эволюция пространственно-временной активности является биологически релевантной.

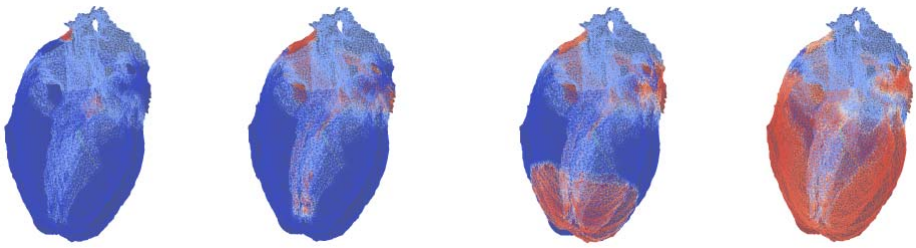


Рис. 3. Результаты моделирования нормальной активности сердца

Рассмотрим процессы возбуждения, соответствующие режимам аритмий. В среду были искусственно введены неоднородности – невозбудимый участок ткани. Изначально инициированные псевдо-концентрические волны разрушаются на этом участке, это привело к образованию спиральных волн – математических образов аритмий.

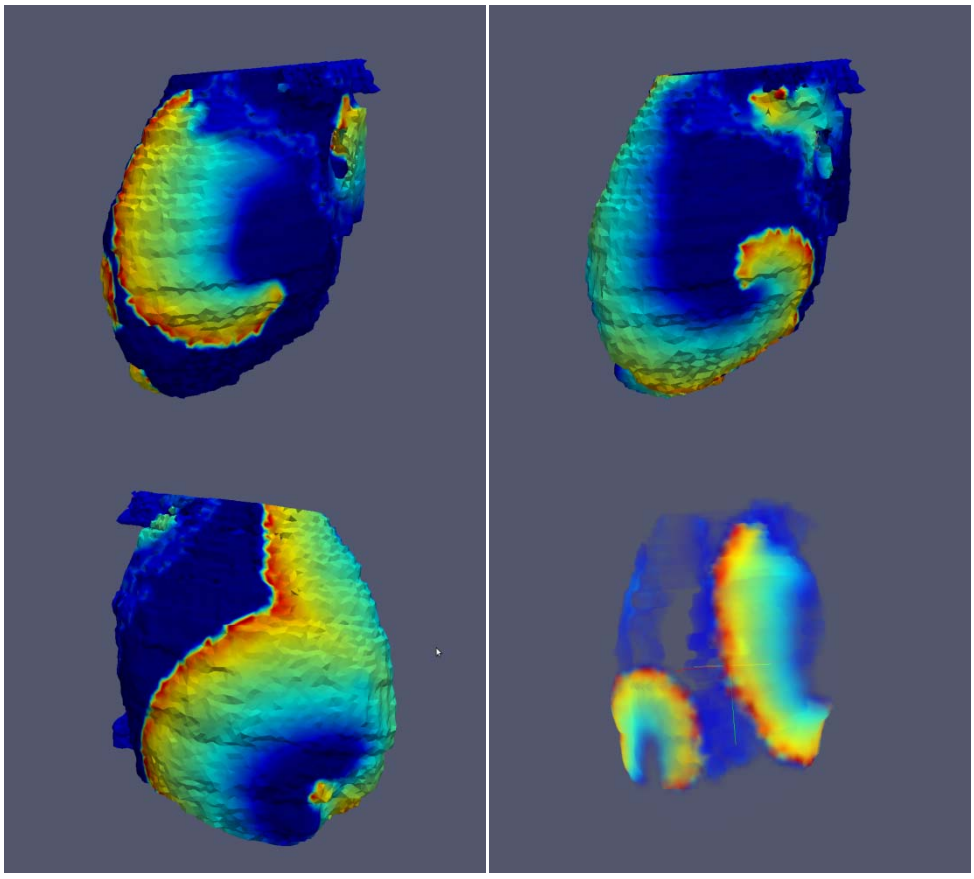


Рис. 4. Моделирование аритмии сердца

7. Производительность

Ниже представлен график масштабируемости решателя программного комплекса «Виртуальное сердце» на кластерной системе.

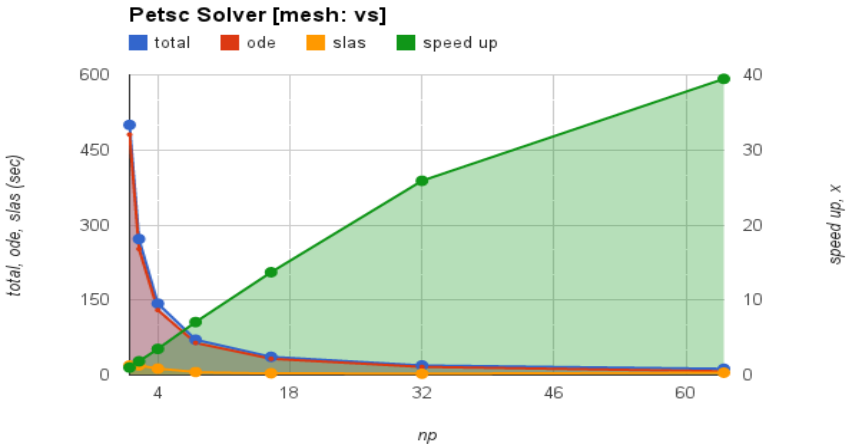


Рис. 5. Масштабируемость

Литература

1. Bastrakov S., Meyerov I., Gergel V., et al. High Performance Computing in Biomedical Applications // Procedia Computer Science 01/2013; 18: P. 10–19.
2. Reddy J.N. An Introduction to the Finite Element Method, third edition. McGraw-Hill Science/Engineering/Math 2005. ISBN-10: 0072466855.
3. Trayanova N.A. Whole-Heart Modeling: Applications to Cardiac Electrophysiology and Electromechanics // Circ Res. 2011. 108. P. 113-128.
4. Greenstein J.L., Winslow R.L. Integrative Systems Models of Cardiac Excitation–Contraction Coupling // Circ Res. 2011. 108. P. 70-84.
5. CGAL User and Reference Manual – [http://www.cgal.org].
6. Paraview docs – [http://www.paraview.org/paraview/help/documentation.html].

РАЗРАБОТКА НОВОГО ПЕРЕУПОРЯДОЧИВАТЕЛЯ СИММЕТРИЧНЫХ РАЗРЕЖЕННЫХ МАТРИЦ

А.Ю. Пирова

Нижегородский госуниверситет им. Н.И. Лобачевского

В работе рассматривается задача уменьшения заполнения фактора разреженной матрицы при разложении Холецкого. Предлагается модификация алгоритма переупорядочения на основе многоуровневого метода вложенных сечений. Приводятся результаты вычислительных экспериментов, показывающие сопоставимость реализации с библиотекой METIS.

Введение

Решение систем линейных алгебраических уравнений вида Ax=b с разреженной симметричной положительно определенной матрицей A лежит в основе многих вычислительных задач компьютерной алгебры [20]. Необходимость решения таких задач возникает при моделировании различных физических процессов, в частности, в конечно-элементных расчетах, а также при моделировании в экономике, химии и других областях. Принцип работы прямых методов основан на факторизации матрицы системы с последующим решением треугольных систем (для симметричных положительно определенных матриц – разложение методом Холецкого). В этом случае решение системы выполняется в несколько этапов.

1. *Переупорядочение матрицы*: перестановка строк и столбцов матрицы для уменьшения размера фактора Холецкого.
2. *Символическая фаза*: анализ структуры исходной матрицы и построение портрета фактора.
3. *Численная фаза*: заполнение полученного портрета численными значениями.
4. *Решение треугольных систем*.

При применении разложения Холецкого для решения разреженной СЛАУ число ненулевых элементов фактора матрицы в разы больше числа ненулевых элементов исходной матрицы – матрица претерпевает *заполнение*. Это приводит к значительным затратам памяти для хранения матрицы, а также увеличению времени на ее факторизацию. Для того чтобы снизить заполнение фактора, к исходной матрице применяется процедура перестановки ее строк и столбцов. Нахождение перестановки, минимизирующей размер фактора или число вычислительных операций для ее факторизации, является NP-полной задачей [18]. В практических приложениях для ее решения применяются эвристические методы.

Разработка новых методов переупорядочения симметричных разреженных матриц, а также их перенос на современные вычислительные архитектуры, представляют большой практический интерес. Исследования в указанном направлении ведут такие ученые как Дж. Карипис, В. Кумар, Дж. Лю, К. Эшкрафт, П. Амстой, Ф. Пеллигрини, Б. Хендриксон др. В ННГУ данную область исследуют М.Х. Прилуцкий, Н.В. Старостин и др. Среди разработанных библиотек, предназначенных для переупорядочения матриц и разделения графов, отметим METIS [8], Scotch [14] и их аналоги для систем с распределенной памятью ParMETIS [10] и PT-Scotch [4].

В данной работе представлены текущие результаты по разработке нового алгоритма для переупорядочения разреженных матриц, а также его высокопроизводительной программной реализации. Работа выполняется в рамках проекта по созданию прямого решателя СЛАУ на кафедре МО ЭВМ факультета ВМК ННГУ им. Н.И. Лобачевского [19, 20].

1. Постановка задачи

Рассмотрим постановку задачи переупорядочения в следующем виде [18]. Пусть дана разреженная симметричная матрица $A = (a_{ij})$ размера $n \times n$. Построим *граф матрицы* $G = (V, E)$, в котором каждая вершина v_i $(i = \overline{0, n-1})$ сопоставлена строке матрицы i, а ребро $(v_i, v_j) \in E$ тогда и только тогда, когда $a_{ij} \neq 0$ $(i, j \in \{0, 1, \dots, n-1\}, i \neq j)$. Смоделируем процесс гауссова исключения при разложении Холецкого. При выполнении *исключения* вершины v из графа G в граф добавляются ребра таким образом, чтобы множество смежных с v вершин стало кликой, сама вершина v удаляется из множества вершин вместе с инцидентными ей ребрами. Пусть $\pi = (\pi_0, \pi_1, \dots, \pi_{n-1})$ – перестановка множества вершин V. *Заполнение* $F(\pi)$, порожденное перестановкой π, – это множество ребер, добавленных при исключении вершин $\pi_0, \pi_1, \dots, \pi_{n-1}$. Поставим задачу нахождения перестановки π^* такой, что $|F(\pi^*)| \to min$. Под *качеством перестановки* будем понимать число ненулевых элементов фактора Холецкого матрицы после применения к ней данной перестановки.

2. Методы переупорядочения разреженных матриц

Основные подходы к переупорядочению симметричной разреженной матрицы с целью снижения заполнения фактора Холецкого можно разделить на две группы – подходы, основанные на методе вложенных сечений и подходы, основанные на методе минимальной степени. Так, метод минимальной степени был предложен Тиннеем и Уолкером в 1969 г. [17]. Этот метод основан на локальной стратегии уменьшения заполнения фактора матрицы и в некотором смысле моделирует процесс гауссова исключения. Ряд модификаций алгоритма, направленных на сокращение времени его работы, был предложен Лю (1985 г.), Амстеем, Дэвисом и Даффом (1996 г.). Метод вложенных сечений, впервые предложенный Джорджем [5] в 1973 г., основан на многократном разбиении графа матрицы при помощи вершинных разделителей. Модификации метода различаются способами выделения разделителя.

Начиная с 1996 г., широко используются модификации метода вложенных сечений, основанные на многоуровневой процедуре разделения графа. Такой подход был применен в работах Карписа и Кумара [9], Эшкрафта и Лю [2], Хендриксона и Ротберга [7] и др. В многоуровневом методе выделение разделителя графа $G = (V, E)$ выполняется в три этапа:

- *Огрубление графа (coarsening)*: построение последовательности сжимающихся графов $G_1, G_2, \dots, G_m$ с помощью построения паросочетания. При этом $|V| > |V_1| > \dots > |V_m|$ и структура каждого следующего в последовательности графа отражает структуру предыдущего.
- *Разделение графа (partitioning)*: поиск разделителя и выделение двух несвязных компонент наименьшего графа G_m.
- *Развертывание графа (uncoarsening)*: проекция разделения наименьшего графа G_m на исходный G через последовательность графов $G_m, G_{m-1}, \dots, G_1$. На каждом шаге i выполняется улучшение разделения графа G_{m-i}, полученного проекцией разделения графа G_{m-i+1}. Для этого используются модификации итерационного метода Кернигана–Лина [11].

3. Программная реализация

В рамках данной работы выполняется разработка и высокопроизводительная программная реализация новой модификации алгоритма переупорядочения симметричных разреженных матриц, основанного на многоуровневом методе вложенных сечений.

В настоящий момент выполнена программная реализация последовательной версии переупорядочения на базе классического многоуровневого метода, описанного выше, с модификацией отдельных стадий. Предварительно выполняется сжатие структуры графа [4] для ускорения работы переупорядочения. В многоуровневом методе на этапе сжатия графа используются методы случайных паросочетаний и паросочетаний тяжелых ребер [5], на этапе разделения – построение структуры уровней смежности графа [3], на этапе улучшения разделения предлагается использовать модификацию итерационного метода, основанного на работах Эшкрафта–Лю [1] и Хендриксона–Ротберга [4]. Суть модификации заключается в изменении условий выполнения итерации алгоритма (перемещения вершины из одной части разделения в другую), а также использовании функции оценки разделения [3]. В ряде случаев для улучшения качества получаемых разделений модифицированный метод Эшкрафта–Лю используется с перезапуском.

4. Результаты вычислительных экспериментов

Для текущей программной реализации был проведен ряд вычислительных экспериментов по переупорядочению матриц из широко распространенной коллекции университета Флориды [16]. Параметры тестовых матриц и аппаратного окружения приведены ниже (таблица 1, таблица 2).

Таблица 1. Характеристики тестовых матриц

Название матрицы	**Порядок**	**Число ненулевых элементов в верхнем треугольнике (NZ)**	**Заполненность, %**
Pwtk	217 918	5 926 171	0,000245
Msdoor	415 863	10 328 399	0,000117
parabolic_fem	525 825	2 100 225	0,000013
tmt_sym	726 713	2 903 837	0,000010
ecology2	999 999	2 997 995	0,000005
G3_circuit	1 585 478	4 623 152	0,000003

Таблица 2. Параметры тестового окружения

Процессор	Четырехъядерный процессор Intel® Xeon L5630 (2.13 GHz)
Память	24 GB
Операционная система	Windows Server SP2
Компилятор	Intel® C++ Composer (в составе Intel® Parallel Studio XE 2013)

Также был проведен ряд экспериментов на указанных тестовых матрицах с использованием библиотек METIS из ParMetis 4 [8], а также решателей СЛАУ SuperLU v. 4.1 [12], MUMPS v. 4.10 [1] (переупорядочиватель PORD [15]). В них использовались параметры переупорядочения по умолчанию. Качество выполненной реализации оценивалось по получаемому числу ненулевых элементов фактора матрицы, а также времени, необходимом для переупорядочения матрицы. Сравнение результатов переупорядочения по числу ненулевых элементов фактора представлено ниже (таблица 3). Для собственной реализации приведено два результата: с лучшим числом ненулевых элементов фактора и с лучшим временем работы (при приемлемом размере фактора). Сравнение времени получения перестановки собственной реализации и в пакете METIS представлено ниже (таблица 4).

Таблица 3. Сравнение результатов переупорядочения по числу ненулевых элементов

Матрица	Порядок, n	Собственное переупорядочение		Metis из Parmetis 4	SuperLU	PORD
		Лучший фактор	Лучшее время + фактор			
pwtk	217 918	46 970 027	50 061 339	49 542 235	108 904 188	48 988 990
msdoor	415 863	51 780 765	55 250 683	51 716 753	111 245 898	54 793 824
parabolic_fem	525 825	25 281 824	25 882 293	27 287 157	52 361 272	28 496 994
tmt_sym	726 713	28 759 956	34 342 694	32 353 685	88 165 169	36 156 598
ecology2	999 999	30 452 184	34 905 653	38 980 242	68 677 613	32 956 605
G3_circuit	1 585 478	92 360 925	103 071 640	95 701 654	Ошибка	130 972 526

Таблица 4. Сравнение времени переупорядочения (в секундах)

Матрица	Порядок, n	Собственное переупорядочение		Metis из Parmetis 4
		Лучший фактор	Лучшее время + фактор	
pwtk	217 918	0,953	0,563	0,83
msdoor	415 863	1,765	1,219	1,35
parabolic_fem	525 825	7,953	3,313	5,82
tmt_sym	726 713	17,344	6,500	8,17
ecology2	999 999	15,656	4,093	8,85
G3_circuit	1 585 478	21,047	10,219	18,22

Как видно из результатов экспериментов, для всех тестовых матриц удалось получить лучшее в смысле минимизации заполнения переупорядочение в сравнении с лидером данной области – пакетом METIS, в котором использовались настройки переупорядочения по умолчанию. Выигрыш в размерах фактора составляет 5–12% на большинстве матриц, на матрице ecology2 – 28%. Время получения такой перестановки в 1,2 – 2,1 раза больше, чем у METIS (в среднем – в 1,5 раза). При этом для каждой матрицы за время в 1,5–2 раза меньшее, чем у METIS, можно получить перестановку, дающую заполнение фактора не хуже, чем на 8%. Для матриц parabolic_fem и ecology2 полученные заполнения на 5 – 10% лучше и в этом случае.

Перестановки, полученные в результате работы метода, использовались при решении СЛАУ одним из академических решателей – MUMPS. Время решения системы при собственном переупорядочении и при переупорядочении из METIS очень близко, что говорит о приемлемости разработанной реализации для задачи решения СЛАУ.

Заключение

В настоящий момент разработана программная реализация алгоритма переупорядочения разреженных матриц, показывающая результаты, близкие по качеству и скорости работы к библиотеке METIS, – одной из наиболее известных и распространенных библиотек в данной области. Данная реализация пока не показывает однозначного превосходства над METIS, но допускает настройку параметров для достижения лучшего, чем у METIS, качества или скорости работы при некритическом ухудшении значения второго критерия.

Направлениями дальнейших исследований являются развитие алгоритма и программной реализации с целью повышения качества и скорости решения задачи, а также разработка параллельной реализации для систем с общей памятью.

Работа выполнена при организационной поддержке лаборатории Intel-ННГУ «Информационные технологии». Автор благодарит И.Б. Меерова, А.В. Сысоева, Е.А. Козинова и А.В. Линева за полезные обсуждения и внимание к работе.

Литература

1. Amestoy P. R., Duff I., L'Excellent J.Y. MUMPS MUltifrontal Massively Parallel Solver. – Version 2.0. // Technical Report TR/PA/98/02. – 1998.
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. Ashcraft C., Liu J.W.H. Using domain decomposition to find graph bisectors //BIT Numerical Mathematics. – 1997. – Vol. 37. – No. 3. – P. 506–534.
4. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering // Parallel Computing. – 2008. – Vol. 34, No. 6. – P. 318–331.
5. George A. Nested dissection of a regular finite element mesh // SIAM J. on Numerical Analysis. – 1973. – Vol. 10, No. 2. – P. 345–363.
6. George A., Liu J.W.H. An automatic nested dissection algorithm for irregular finite element problems // SIAM J. on Num. Anal. – 1978. – Vol. 15, No. 5. – P. 1053–1069.
7. Hendrickson B., Rothberg. Improving the runtime and quality of nested dissection ordering // SIAM J. on Scientific Computing. – 1999. – Vol. 20. – P. 468–489.
8. 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. – 2011. URL: [http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf].
9. 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.
10. Karypis G., Kumar V. ParMETIS: Parallel graph partitioning and sparse matrix ordering library // Technical Report TR 97-060, University of Minnesota. – 1997.
11. Kernighan B.W., Lin S. An efficient heuristic procedure for partitioning graphs // The Bell System Technical J. – 1970. – Vol. 29. – P. 291–307.
12. Li X.S. An overview of SuperLU: Algorithms, implementation, and user interface // ACM Transactions on Math. Software. – 2005. – Vol. 31, No. 3. – P. 302–325.
13. Pellegrini F. Shared memory parallel algorithms in Scotch 6. – URL: [http://graal.ens-lyon.fr/MUMPS/doc/ud_2013/Pellegrini.pdf].
14. Pellegrini F., Roman J. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs // High-Performance Computing and Networking. – Springer Berlin Heidelberg, 1996. – P. 493–498.
15. 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.
16. The University of Florida Sparse Matrix Collection – URL.: [www.cise.ufl.edu/research/sparse/matrices/].
17. 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.
18. 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.
19. Bastrakov S., Meyerov I., Gergel V. et al. High performance computing in biomedical applications // Procedia Computer Science, 2013. – Vol. 18. – P. 10–19.
20. 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 ННГУ. – Н. Новгород: Изд-во ННГУ, 2012. – №5(2) – С. 376-384.

СТРУКТУРНОЕ ПРЕОБРАЗОВАНИЕ ГРАФА УПРАВЛЕНИЯ ПАРАЛЛЕЛЬНЫМИ ВЫЧИСЛЕНИЯМИ ДЛЯ СИСТЕМ С РАСПРЕДЕЛЁННОЙ ПАМЯТЬЮ MPI

Д.А. Попова-Коварцева

Самарский государственный аэрокосмический университет им. С.П. Королёва

Выделены формальные правила описания моделей параллельных алгоритмов, принятые в технологии графосимволического программирования. Предложены алгоритмы, реализующие эквивалентные преобразования исходной граф-модели параллельного алгоритма первоначально с помощью суперпозиции пользовательской версии модели к виду обобщенного суперагрегата. В дальнейшем, с помощью алгоритмов F-нумерации и декомпозиционного преобразования суперагрегат приводится к стандартному эквивалентному виду, способному выполняться на высокопроизводительных системах с распределённой памятью.

Введение

В настоящее время наиболее популярным способом написания параллельных программ для высокопроизводительных систем с распределенной памятью является создание параллельных кодов программ с использованием библиотек MPI или подобных им средств. Однако использование MPI связано с изменением стиля программирования и дополнительными исследованиями корректности введенного параллелизма [1]. Даже имея хорошую идею распараллеливания некоторого вычислительного процесса, исполнитель вынужден произвести фактически системный анализ разрабатываемого параллельного алгоритма и преобразовать его к виду, реализуемому соответствующими программными средствами, например, C++ и MPI.

Возникающая необходимость равнозначного владения технологиями программирования на языках высокого уровня (C++, C#) и средством распараллеливания программ MPI еще дальше отдаляет «конечных» пользователей от возможности участия в разработке авторских параллельных приложений.

В связи с чем, актуально направление исследований, связанное с автоматизацией процессов преобразования моделей параллельных алгоритмов к стандартизованному виду, пригодному для исполнения на современных суперкомпьютерах.

1. Основные положения

Существует множество графических моделей описания алгоритмов параллельных вычислений. В области представления вычислительных алгоритмов наиболее удобной является форма, близкая по форме к блок-схемам, которая реализована в параллельной версии технологии графосимволического программирования (ГСП) в виде системы PGRAPH [3, 4].

В системе PGRAPH [3] вычислительная модель параллельных алгоритмов описывается четверкой M_G=<D, A, Ψ, G>, где D – множество данных некоторой предметной области, A – множество вычислимых функций, определенных над данными предметной области, Ψ – множество дуг управления, $G = \{A, \Psi\}$ – ориентированный помеченный граф, описывающий граф-модель вычислительного процесса.

Тип дуги Ψij, исходящей из вершины i и входящей в вершину j, определим формально как функцию $T(\Psi ij) \in \{1,2,3\}$, где:

1 – последовательная дуга (описывает передачу управления на последовательных участках вычислительного процесса);

2 – параллельная дуга (обозначает начало нового параллельного вычислительного процесса);

3 – терминирующая дуга (завершает параллельный вычислительный процесс).

Для описания отдельного вычислительного процесса в ГСП вводится понятие параллельной ветви β модели - подграфа графа G, начинающегося параллельной дугой (тип этой дуги $T(\Psi ij) = 2$, на схеме помечается «кружочком») и заканчивающегося терминирующей дугой (тип этой дуги $T(\Psi ij) = 3$, на схеме помечается перечёркнутой дугой).

Формально параллельную ветвь можно определить тройкой:

$$\beta = <A_\beta, \Psi_\beta, R_\beta> ,$$

где A_β – множество вершин ветви, Ψ_β – множество дуг управления ветви, R_β – отношение над множествами вершин и дуг ветви, определяющее способ их связывания.

Графическая модель обычно содержит несколько параллельных ветвей, каждая из которых образует отдельный процесс. В этом смысле модель параллельных вычислений можно представить как объединение нескольких параллельных ветвей: $G = \bigcup \beta_k$, где β_k – параллельные ветви графа G. Каждая параллельная ветвь исполняется на отдельном процессоре.

Число параллельных ветвей в модели фиксируется при ее построении, при этом максимальное количество ветвей не ограничивается. Каждая ветвь имеет ровно один вход и один выход.

2. Декомпозиция Р-графа

Технология ГСП не налагает серьёзных ограничений на форму представления моделей параллельных алгоритмов. В частном случае, с учетом введенной выше семантики, модель параллельного алгоритма может быть представлена совокупностью иерархически вложенных двухполюсных последовательно-параллельных ориентированных P-графов, которая при «развертывании» всех вложений может быть представлена обобщенным P-графом, полюсами которого являются одна начальная (исток) и одна конечная (сток) вершины. Первоначальную последовательную ветвь агрегата, содержащую исток называют мастер-ветвью агрегата.

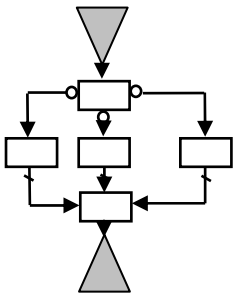


Рис. 1.

Введём понятие *правильно построенного агрегата* (ППА), схематично представленного на рис. 1. Правильно построенный агрегат имеет только одну вершину ветвления, одну терминирующую вершину и множество параллельных вершин, связанных с вершинами ветвления и терминации (см. рис.1).

Приведение обобщенной модели P-графа алгоритма к «реализационному» виду (структурное преобразование исходной модели – десуперпозиция) основывается на введении понятия F-нумерации.

В нашем случае под *F-нумерацией* графа G будем понимать инъективное отображение множества вершин $A(G)$ графа на множество слов X^* алфавита X: $F : A(G) \to X^*$.

Пусть алфавит имеет вид: $X = \{"H", "V", "E", ".", "0", "1", "2",...\}$. Слова нумерации имеют следующую семантику: $M =< N_{ur} \ . \ Type \ . \ N_1 \ . \ N_2. \ \cdots . N_{ur} >$, где N_{ur} – количество уровней иерархии при вложенности параллельных структур друг в друга; N_i – номер вершины на *i*-м уровне иерархии (если $i=N_{ur}$) или номер иерархии вложения; *Type* – тип вершины (*Type=H* для вершины ветвления; *Type=E* для терминальных вершин; для остальных *Type=V*). Представленный код вершины точно идентифицирует структуру направленного последовательно-параллельного *P*-графа.

F-нумерация начинается с начальной вершины и реализуется на основе стратегии поиска в глубину с возвратом, пока не будет реализован обход всех вершин графа [2].

На рис. 2 представлен пример модели параллельного алгоритма (модель А). Здесь вершины ветвления обозначены символом “H”. Терминирующие вершины - символом “E”. Вершины, принадлежащие параллельным ветвям, обозначены символом “#”.

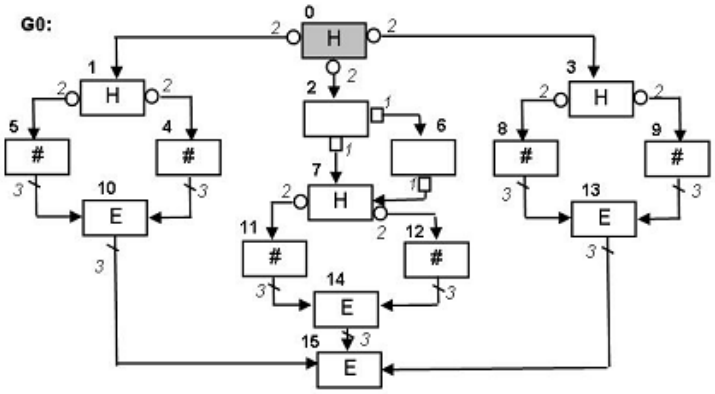


Рис. 2. Исходный *P*-граф G0 (модель А)

Мастер-ветвь агрегата G0 содержит 3 параллельные ветви, в каждую из которых вложены еще по две параллельные ветви, и в этом смысле G0 не является ППА. После реализации алгоритма разметки графа получим *F*-нумерацию вершин графа, представленную на рисунке 3.

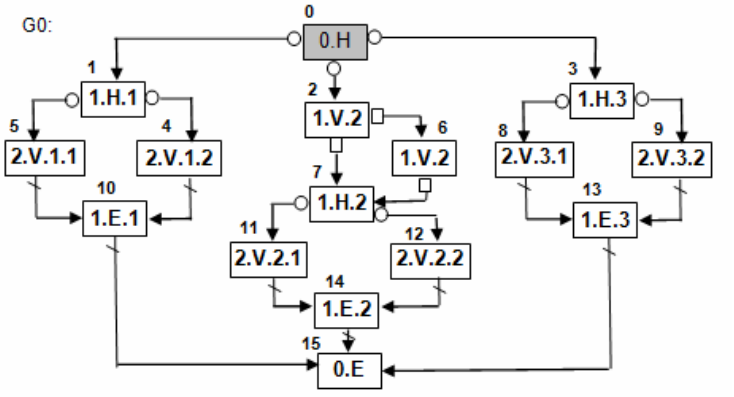


Рис. 3. Размеченный граф G0 (модель В)

Для выполнения параллельных вычислений необходимо произвести разбиение структуры графа G0 на совокупность правильно построенных агрегатов (модель С), например, как это показано на рис. 4.

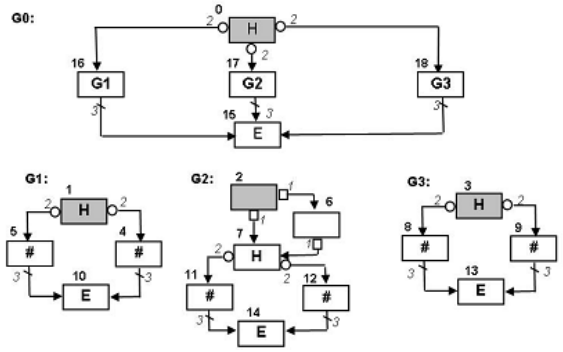


Рис. 4. Разбиение агрегата G0 на правильные компоненты (модель C)

Представленная на рис. 3 нумерация вершин графа позволяет организовать процедуру редукции (десуперпозицию) исходного *P*-графа на совокупность вложенных правильно построенных агрегатов, как это показано на рис. 4, т.е. реализовать преобразование модели A в модель C.

Десуперпозиция реализуется в соответствии со следующими правилами:

Правило 1. Вершины каждой из параллельных веток графа, находящиеся между параллельной и терминирующей вершинами одного уровня, агрегируются в отдельный подграф. Вложенному агрегату присваивается новое имя. Выделенный подграф заменяется в исходном подграфе новой вершиной.

Правило 2. Если новый подграф не является ППА, то к нему применяется правило 1.

Правило 3. Процедура десуперпозиции завершается, если для исходного и всех новых подграфов невозможно применить правило 1.

3. Преобразование графа управления модели параллельного алгоритма к стандартному (реализационному) виду

Завершающий этап приведения описания графа управления модели параллельного алгоритма к стандартизованному виду фактически сводится к удалению из модели C всех терминирующих дуг и добавлению для каждого подграфа по одной управляющей дуге, направленной от параллельной к терминальной вершине. В результате формируется стандартная форма модели параллельных вычислений (модель D), представленная на рис. 5. В модели D все концевые вершины графов (кроме терминирующей, имеющей тип E) закрепляются за одним из компьютеров кластера, т.е к процессорам p1, p2 и т.д., на которых запускаются соответствующие вычислительные задачи. После завершения всех вычислений управление передаётся на терминирующую вершину типа E.

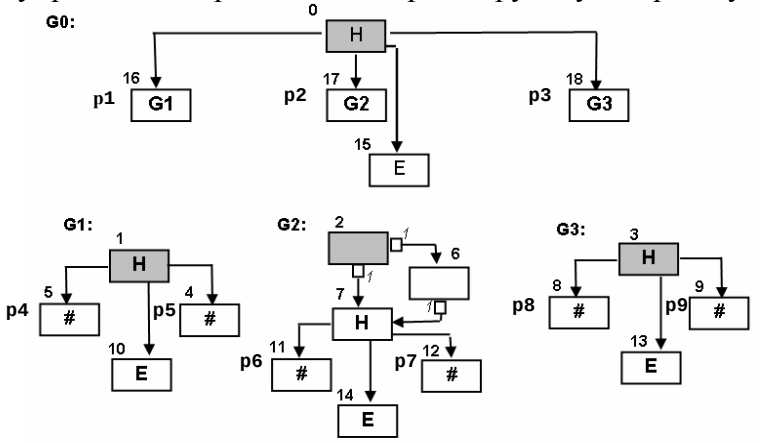


Рис. 5. Стандартная форма модели графа G0 (модель D)

В результате предложенный метод автоматизации организации параллельных вычислений для произвольной модели параллельного алгоритма сводится к последовательности эквивалентных декомпозиционных преобразований по цепочке: модель A -> модель B -> модель C -> модель D.

Заключение

Предложенный алгоритм структурного преобразования модели параллельного алгоритма позволяет без участия разработчика программы произвести все необходимые структурные преобразования графа управления модели к виду, пригодному для реализации вычислений в среде MPI.

Литература

1. Гергель В.П. Теория и практика параллельных вычислений. – М.: БИНОМ, 2007.- 423с.
2. Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. - 352 с.
3. Жидченко В.В., Коварцев А.Н. Моделирование синхронных параллельных вычислений при построении математических моделей сложных систем // Первая международная конференция «Системный анализ и информационные технологии» САИТ-2005: Труды конференции. В 2т. Т.2. – М.: КомКнига, 2005. – С. 154-160.
4. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. – Самара: Самарский государственный аэрокосмический университет, 1999. – 150 с.

МНОГОКРИТЕРИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ ОДНОРОДНОГО ОГРАНИЧЕННОГО РЕСУРСА В ИЕРАРХИЧЕСКИХ СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

М.Х. Прилуцкий, У.С. Кулакович

Нижегородский госуниверситет им. Н.И. Лобачевского

Задача распределения однородного ограниченного ресурса в иерархических системах формализуется в виде системы линейных алгебраических неравенств. Ставится задача многокритериальной оптимизации, решение которой осуществляется путём сведения её к задаче поиска оптимальной вершины многомерного многозначного куба с использованием параллельной вычислительной среды.

Широкий класс прикладных задач связан с распределением ограниченных ресурсов в иерархических системах. Особое место среди таких задач занимают задачи, формализация которых возможна в виде многокритериальных задач с линейными ограничениями. Примерами таких задач являются: задачи объёмно-календарного планирования [1, 2]; транспортные задачи с промежуточными пунктами [3]; задачи распределения мощностей каналов передачи данных провайдерами сети Интернет [4] и др.

Пусть $G=(V,A)$, $A\subseteq V^2$ – ориентированный граф без петель и контуров, моделирующий систему распределения однородного ограниченного ресурса. Множество вершин графа V соответствует элементам системы, множество дуг A моделирует связи между элементами системы, посредством которых происходит передача ресурса.

Пусть x_j – количество ресурса, которое поставим в соответствие элементу системы с номером j, а B_j и C_j – соответственно нижняя и верхняя границы допустимых значений распределяемого ресурса, $0\le B_j\le C_j<\infty$, $j\in V$.

Тогда ограничения на допустимые значения распределяемого ресурса задаются следующей системой линейных алгебраических неравенств:

$$B_j\le \sum_{j|(i,j)\in A} x_j\le C_j,\ j\in V. \qquad (1)$$

Среди ограничений системы (1) выделим так называемые «контролируемые» ограничения, которые определяют условия «эффективного» функционирования системы. Пусть K - множество «контролируемых» ограничений, $K\in 2^V$, $|K|=q_0$. Для задачи распределения однородного ограниченного ресурса в иерархических системах эффективность функционирования зависит от нескольких факторов. Как показал опыт решения таких задач ([2, 3]), формализация этих факторов в виде непрерывных функций для пользователей затруднительна. Пользователю проще оценить показатели искомого плана, указав границы для величин отклонений, в которых эти величины являются «отличными», «очень хорошими», «хорошими», «удовлетворительными» и др. Тогда функции отклонений $\chi_i(x)$, $i\in K$, должны быть кусочно-постоянными, разбивающими множество величин отклонений по каждому критерию на области «качества» отклонений. Пусть функции отклонений имеют вид $\chi_i(\sum_{j|(i,j)\in A} x_j, s_i^0, s_i^1,..., s_i^p)$ и определены

на множестве $[B_i, C_i]$, со значениями из множества $\{0,1,...,\text{p}\}$, где s_i^v, $v \in 0,1,..., p$ – совокупность вложенных друг в друга интервалов, $s_i^v \subseteq s_i^{v+1}$, $s_i^p = [B_i, C_i]$, $v \in 0,1,..., p$, причём $\chi_i(\sum_{j|(i,j)\in A} x_j, s_i^0, s_i^1, ..., s_i^p) = t$, если $\sum_{j|(i,j)\in A} x_j \in s_i^t$ и $\sum_{j|(i,j)\in A} x_j \notin s_i^{t-1}$, $i \in K$.

Задача распределения однородного ограниченного ресурса в иерархических системах в этом случае заключается в определении такого допустимого решения системы (1), при котором функции предпочтений принимают экстремальные значения:

$$\chi_i(\sum_{j|(i,j)\in A} x_j, s_i^0, s_i^1, ..., s_i^p) \to \min, \; i \in K. \qquad (2)$$

Поставленная задача (1)-(2) является задачей многокритериальной оптимизации. Предполагая, что выбранные показатели «качества» распределения ресурса упорядочены с точки зрения их значимости при определении эффективности функционирования системы, в качестве схемы компромисса будем рассматривать лексикографическое упорядочивание частных критериев оптимальности.

Как и в [2] рассмотрим $(p+1)$-ичный (количество интервалов) q_0-мерный (количество контролируемых ограничений) куб, на котором зададим линейный порядок π. Множество вершин куба будем обозначать $V_{(p+1),q_0}$. Каждая вершина куба $\vec{v}$ определяется q_0-мерным вектором, где $v_q \in \{0,1,\ldots,\text{p}\}$, $q = \overline{1,q_0}$. Каждой вершине куба $\vec{v} \in V_{(p+1),q_0}$ поставим в соответствие систему $D(\vec{v})$. Система $D(\vec{v})$ содержит независящие от вершины куба неконтролируемые ограничения из (1), и q_0 ограничений, зависящих от вершины куба: если $v_q = t$, то ограничение с номером q преобразуется в ограничение $\sum_{j|(i,j)\in A} x_j \in s_i^t$, $q = \overline{1,q_0}$.

В качестве порядка π выберем лексикографическое отношение порядка: $v^1 \pi v^2$ тогда и только тогда, если $v_l^1 = v_l^2$, $l = 1,2,...,s$, $v_{s+1}^1 < v_{s+1}^2$. Задача распределения однородного ограниченного ресурса в этом случае будет заключаться в определении такой (оптимальной) вершины куба $\vec{v}^0$, которой соответствует совместная система ограничений типа (1), и для которой выполняется $v^0 \pi v$ для всех вершин куба, для которых совместны соответствующие им системы типа (1).

Вычислительная сложность рассмотренного в [2] алгоритма дихотомического поиска оптимальной вершины куба имеет порядок $q_0 \log_2(p+1)$. При этом число вершин многомерного многозначного куба имеет порядок $(p+1)^{q_0}$.

В качестве примера рассматривается проблема формирования планов для цехов предприятия в объёмных показателях. Сформулируем подобную задачу не в общей постановке, с учётом всех зависимостей и связей, а с некоторой степенью идеализации [2] – вместо конкретных работ с их длительностями рассмотрим объемные показатели (нормо-часы, рубли, условные тонны), выступающие в роли ресурсов, используемых в производственных процессах. Содержательно задача объёмно-календарного планирования формулируется следующим образом. Требуется распределить общий план предприятия в объёмных показателях по различным группам: цехам, проектам, заказам, изделиям, деталям и тактам планирования. Показатели искомого плана делятся на «жесткие», выполнение которых обязательно, и «желательные», к выполнению которых нужно стремиться. Жесткие показатели формализуются в виде ограничений, а «жела-

тельные» - в виде критериев оптимальности. Тогда задача объёмно-календарного планирования ставится как многокритериальная задача (учет «желательных» показателей) с ограничениями (учет «жестких» показателей), которые в рассмотренной идеализации являются линейными.

Пусть $s=1,...,S$ – номера цехов предприятия,
$p=1,...,P$ – номера проектов,
$j=1,...,J$ – номера заказов,
$i=1,...,I$ – номера изделий,
$l=1,...,L$ – номера деталей,
$t=1,...,T$ – номера тактов планирования.

К показателям искомого плана могут относиться, например, величины: A_t – объем работ, который должен быть выполнен предприятием в *t-ый* такт планирования, B_{jt} – объем работ, который должен быть выполнен по *j-му* заказу в период t, C_{jit} – объем работ, который должен быть выполнен по *i-му* изделию *j*-го заказа в период t, D_{sj} - объём работ, который может быть выполнен по заказу j в цехе s, E_j – объём работ, который необходимо выполнить по заказу j, F_p – объём работ, который должен быть выполнен предприятием по *p*-ому проекту, G_{sl} – объём работ, который должен быть выполнен в *s-ом* цехе по *l*-ой детали, a_{spjilt} – объём работ, оставшийся невыполненным в цехе s по изделию i заказа j проекта p в период t, $s=\overline{1,S}$, $p=\overline{1,P}$, $j=\overline{1,J}$, $i=\overline{1,I}$, $l=\overline{1,L}$, $t=\overline{1,T}$. Пусть к «жестким» показателям искомого плана относятся показатели A_t, B_{jt}, C_{jit}, D_{sj}, F_p, а E_j и G_{sl} являются «желательными» показателями. Тогда задача объемно-календарного планирования формулируется следующим образом: требуется определить такие величины x_{spjilt} – объем работ, который будет выполнен в цехе s по детали l изделия i заказа j проекта p в период планирования t, для которых выполняются соотношения:

$$\sum_{s\in S}\sum_{p\in P}\sum_{j\in J}\sum_{i\in I}\sum_{l\in L} x_{spjil} \ge A_t,\ t\in T; \tag{3}$$

$$\sum_{s\in S}\sum_{p\in P}\sum_{i\in I}\sum_{l\in L} x_{spil} \ge B_{jt},\ j\in J,\ t\in T; \tag{4}$$

$$\sum_{s\in S}\sum_{p\in P}\sum_{l\in L} x_{spl} \ge C_{jit},\ j\in J,\ i\in I,\ t\in T; \tag{5}$$

$$\sum_{p\in P}\sum_{i\in I}\sum_{l\in L}\sum_{t\in T} x_{pilt} \le D_{sj},\ s\in S,\ j\in J; \tag{6}$$

$$\sum_{s\in S}\sum_{j\in J}\sum_{i\in I}\sum_{l\in L}\sum_{t\in T} x_{sjilt} \ge F_p,\ p\in P; \tag{7}$$

$$0 \le x_{spjilt} \le a_{spjilt},\ s=\overline{1,S},\ p=\overline{1,P},\ j=\overline{1,J},\ i=\overline{1,I},\ l=\overline{1,L},\ t=\overline{1,T}; \tag{8}$$

с учетом минимизируемых критериев:

$$f_{sl}\left(\sum_{p=1}^{P}\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{t=1}^{T} x_{spjilt}, G_{sl}\right), s=\overline{1,S},\ l=\overline{1,L}; \tag{9}$$

$$\varphi_j\left(\sum_{s=1}^{S}\sum_{p=1}^{P}\sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{t=1}^{T} x_{spjilt}, E_j\right),\ j=\overline{1,J}. \tag{10}$$

Функции $f_{sl}\left(\sum_{p=1}^{P}\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{t=1}^{T} x_{spjilt}, G_{sl}\right), s=\overline{1,S},\ l=\overline{1,L}$ и $\varphi_j\left(\sum_{s=1}^{S}\sum_{p=1}^{P}\sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{t=1}^{T} x_{spjilt}, E_j\right),\ j=\overline{1,J}$, являются функциями оценок отклонений объемов выполняемых работ от заданных величин E_j и G_{sl} - «желательных» показателей искомого плана, $s=\overline{1,S},\ l=\overline{1,L},\ j=\overline{1,J}$.

Поставленная задача (3) – (10) является многокритериальной задачей с линейными ограничениями и критериями, вид которых зависит от выбранных функций.

Среднее машиностроительное предприятие имеет следующие размеры: количество цехов предприятия – 10, проектов – 5, заказов – 20, изделий – 100, деталей – 10000, тактов планирования – 70. В этом случае общее число переменных имеет порядок 10^{10}, количество критериев задачи – порядок 10^5. Даже в случае, когда точность решения задачи не велика и, например, $p=1$, область поиска (количество вершин многомерного многозначного куба) оптимальной вершины куба имеет порядок 2^{10^5}.

Таким образом, на практике реальные задачи объёмно-календарного планирования могут иметь большую размерность (количество контролируемых элементов q_0, количество вложенных сегментов $p+1$ могут быть «велики»), поэтому предлагается параллельная схема поиска оптимальной вершины многомерного многозначного куба.

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 оптимального планирования производства //Automation and Remote Control, 2010. Vol. 71, № 10, P. 2145-2151.
2. Prilutskii M.Kh. Multicriterial Multi-Index Resourse Sheduling Problems // Journal of Computer and Sciences International, 2007. Vol. 46, № 1, P. 83-87.
3. Афраймович Л.Г., Прилуцкий М.Х. Многоиндексные задачи распределения ресурсов в иерархических системах // Автоматика и телемеханика. 2006. № 6. С.194-205.
4. Прилуцкий М.Х. Распределение однородного ресурса в иерархических системах древовидной структуры // Труды международной конференции "Идентификация систем и задачи управления SICPRO 2000". М.: Институт проблем управления им. В.А.Трапезникова РАН, 2000. С. 2038-2049.

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ РАСПРОСТРАНЕНИЯ СЕЙСМИЧЕСКИХ ВОЛН В СЛОЖНО ПОСТРОЕННЫХ СРЕДАХ НА ГИБРИДНОМ КЛАСТЕРЕ

А.Ф. Сапетина

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассматривается численное моделирование распространения упругих волн в 3D неоднородных средах на основе решения полной системы уравнений теории упругости с соответствующими начальными и граничными условиями. Моделирование проводится на основе конечно-разностного метода с использованием вспомогательного алгоритма CFS-PML для поглощения отражений упругих волн от границы расчетной области. Предлагаются способы распараллеливания программы для численных расчетов на гибридном кластере, оснащенном графическими картами общего назначения, с помощью технологий CUDA и MPI.

Введение

Одним из путей изучения строения геологических объектов является активный вибросейсмический мониторинг. Проведение натурных геофизических экспериментов позволяет получить некоторые представления о скоростных параметрах упругой среды, а также о геометрии изучаемого объекта. Но в процессе обработки результатов полевого эксперимента могут возникнуть интересные эффекты, требующие дальнейшего исследования. Создание математических моделей изучаемых объектов и дальнейшее моделирование сейсмических полей помогают в изучении реальных объектов.

В связи с тем, что реальная область исследования зачастую имеет довольно сложный рельеф, не всегда удается поставить площадную систему наблюдения для решения обратной задачи геофизики. Поэтому приходится решать набор прямых задач с целью определения параметров изучаемой среды, соответствующих экспериментальным наблюдениям на поверхности изучаемых грязевых вулканов.

Программный комплекс, решающий поставленную задачу на основе выбранного математического аппарата, должен иметь возможность моделирования упругой неоднородной среды со специфической геометрией. В связи со сложностью и масштабом моделируемой области, решение задачи численного моделирования распространения упругих волн от сосредоточенного источника может требовать значительных вычислительных ресурсов. Поэтому необходима разработка параллельных программ для уменьшения времени расчета и возможностью моделирования «больших» 3D-моделей упругих сред.

Д.А. Караваевым были разработаны и обоснованы методы численного моделирования сейсмических полей для 3D сложно построенных сред, характерных для грязевых вулканов. Им же разработан инструментарий для решения прикладных задач численного моделирования сейсмических полей, включающий построитель 3D-моделей неоднородных упругих сред и параллельную программу для численного моделирования распространения упругих волн, реализованную на кластерах с MPP-архитектурой, с использованием технологий MPI и OpenMP [1, 2].

Стоит отметить консервативность в развитии алгоритмов по сравнению с современными вычислительными системами, которые развиваются быстрее. С началом активного применения графических карт для вычислений общего назначения возникает проблема модификации алгоритмов под новую архитектуру.

В начале 2012 года в ССКЦ при ИВМиМГ СО РАН запущен гибридный кластер с графическими процессорами NVidia Tesla M2090. Также подобный вычислительный комплекс функционирует на базе Информационно-вычислительного центра НГУ.

В свете приведенных фактов стала актуальной адаптация существующих и создание новых алгоритмов и программного обеспечения для решения задачи численного моделирования распространения упругих волн на гибридном кластере.

1. Постановка задачи

Численное моделирование распространения сейсмических волн в сложно построенных упругих неоднородных средах проводится на основе решения полной системы уравнений теории упругости с соответствующими начальными и граничными условиями, записанной в терминах вектора скоростей смещений $\vec{u} = (U, V, W)^T$ и тензора напряжений $\vec{\sigma} = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})^T$. В качестве области моделирования рассматривается изотропная 3D неоднородная сложно построенная упругая среда, представляющая собой параллелепипед, одна из граней которого является свободной поверхностью (плоскость $z = 0$).

Основные уравнения в векторной форме могут быть представлены в следующем виде:

$$\rho \frac{\partial \vec{u}}{\partial t} = [A]\vec{\sigma} + \vec{F}(t,x,y,z),\ \frac{\partial \vec{\sigma}}{\partial t} = [B]\vec{u},$$

$$A = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} & 0 \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix},\ B = \begin{bmatrix} (\lambda+2\mu)\frac{\partial}{\partial x} & \lambda\frac{\partial}{\partial y} & \lambda\frac{\partial}{\partial z} \\ \lambda\frac{\partial}{\partial x} & (\lambda+2\mu)\frac{\partial}{\partial y} & \lambda\frac{\partial}{\partial z} \\ \lambda\frac{\partial}{\partial x} & \lambda\frac{\partial}{\partial y} & (\lambda+2\mu)\frac{\partial}{\partial z} \\ \mu\frac{\partial}{\partial y} & \mu\frac{\partial}{\partial x} & 0 \\ \mu\frac{\partial}{\partial z} & 0 & \mu\frac{\partial}{\partial x} \\ 0 & \mu\frac{\partial}{\partial z} & \mu\frac{\partial}{\partial y} \end{bmatrix},$$

где t – время, $\rho(x,y,z)$ – плотность, $\lambda(x,y,z)$, $\mu(x,y,z)$ – параметры Ламе.

Начальные условия имеют вид:

$$\sigma_{xz}|_{t=0} = 0,\ \sigma_{yz}|_{t=0} = 0,\ \sigma_{xy}|_{t=0} = 0,\ \sigma_{xx}|_{t=0} = 0,\ \sigma_{yy}|_{t=0} = 0,\ \sigma_{zz}|_{t=0} = 0,$$

$$U(x,y,z)|_{t=0} = 0,\ V(x,y,z)|_{t=0} = 0,\ W(x,y,z)|_{t=0} = 0.$$

В качестве граничных условий на свободной поверхности выступают:

$$\sigma_{xz}|_{z=0} = 0,\ \sigma_{yz}|_{z=0} = 0,\ \sigma_{zz}|_{z=0} = 0.$$

В данной постановке предполагается, что правая часть (массовая сила) имеет следующий вид:

$$\vec{F}(t,x,y,z) = F_x\vec{i} + F_y\vec{j} + F_z\vec{k},$$

где $\vec{i}$, $\vec{j}$, $\vec{k}$ – единичные направляющие векторы координатных осей.

2. Метод решения задачи

Метод решения поставленной задачи основан на использовании конечно-разностного метода. Алгоритм построения конечно-разностной схемы изложен в статье [3]. Расчет сеточных коэффициентов в разностной схеме проводится на основе интегральных законов сохранения (поскольку параметры λ, μ и ρ могут быть разрывными). Используемая конечно-разностная схема имеет второй порядок аппроксимации по времени и пространству, рассматриваются только равномерные сетки

В численном моделировании упругих волн возникают фиктивные отражения от границ моделируемой области из-за усечения реальной области моделирования. Для устранения этого эффекта использован метод CFS-PML [4, 5]. Он дает более «качественную» картину волнового поля для данной задачи, в сравнении с классическим методом PML, и более экономичен с вычислительной точки зрения.

Поскольку область расчета представляет параллелепипед, а свободная поверхность располагается на верней его грани ($z = 0$), то каждая из его границ, за исключением свободной поверхности, окружается поглощающим слоем. Во внутренности все рассчитывается по первоначальным конечно-разностным уравнениям, а при попадании волны в зону поглощения, происходит расчет по новым формулам, описывающим подход к созданию поглощающих границ.

3. Адаптация алгоритмов для реализации на гибридном кластере

В ходе работы создана параллельная программа для численного моделирования распространения упругих волн в трехмерных неоднородных упругих средах, реализующая указанный конечно-разностный метод и метод поглощающих слоев и рассчитанная на использование нескольких GPU. Для реализации процесса распараллеливания на GPU используется технология CUDA. Для моделирования упругих сред со сложной геометрией используется построитель 3D моделей неоднородных упругих сред, разработанный Д.А. Караваевым.

В данной реализации проводится декомпозиция расчетной области на несколько слоев (по числу графических карт) вдоль направления одной из координатных осей (например, вдоль оси Oz) с перекрытием подобластей. Каждая графическая карта рассчитывает свою сеточную область на каждом временном шаге независимо от другой, за исключением точек, находящихся на границе между двумя соседними областями. Эти точки являются общими для каждой из областей и для продолжения счета необходимо производить обмен информацией об искомых величинах между «соседями». Переключение управления между картами и обмен данными производится за счет средств CUDA. При этом на каждой графической карте параллельная часть кода выполняется как большое количество нитей.

На данный момент осуществляется задействование всех узлов гибридного кластера ССКЦ СО РАН для численного моделирования данной задачи. Архитектура этого кластера представляет из себя 40 вычислительных узлов HP SL390s G7, по три карты NVIDIA Tesla M2090 на архитектуре Fermi (Compute capability 2.0) на узле. У каждой карты 1 GPU с 512 ядрами и 6 ГБ оперативной памяти. Суммарно гибридный кластер содержит 80 процессоров (480 ядер) CPU и 120 процессоров (61440 ядер) GPU. Пиковая производительность – 85 Тфлопс.

Для использования такой архитектуры проводится одномерная декомпозиция расчетной области на слои по количеству узлов кластера, и декомпозиция каждого слоя на подслои по количеству используемых на узле графических ускорителей. Обмены между узлами проводятся с помощью технологии MPI.

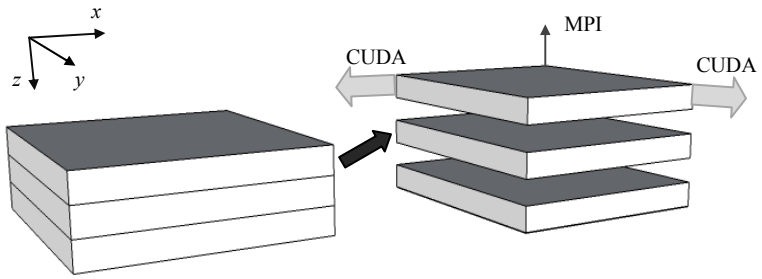


Рис. 1. Декомпозиция расчетной области для гибридного кластера

В ходе работы была решена проблема выбора сетки блоков и сетки нитей в блоке для осуществления вычислений на GPU. С этим выбором естественно связаны модификации выбранных алгоритмов. Были рассмотрены два наиболее целесообразных (по степени параллелизма) варианта задания сеток и проведен анализ времени исполнения выбранных вариантов. На основе полученных выводов последующая работа велась для полностью трехмерной декомпозиции расчетной области на нити и блоки, размер которых по компоненте *x* по возможности должен быть равен или хотя бы кратен длине warp'а.

Для реализации метода поглощающих границ добавлены отдельные функции-ядра для каждого поглощающего слоя и своя трехмерная сетка нитей и блоков, соответствующая геометрии каждого поглощающего слоя. Этот вариант значительно увеличвает количество функций в программе, но позволяет избежать увеличения времени счета из-за ветвления.

Для уменьшения времени доступа к памяти используется константная память графической карты. В ней хранятся самые часто используемые константы. В будущем планируется задействовать разделяемую память графической карты.

В результате всех оптимизаций получено ускорение во времени работы программы использующей 2 GPU, по сравнению с аналогичной программой, использующей один CPU более чем в 10 раз.

В дальнейшем планируется провести сравнение времени работы параллельной программы, реализованной для гибридного кластера, со временем работы параллельной программы, реализованной на суперкомпьютере НКС-30 (классическая MPP-архитектура) с использованием технологий MPI и OpenMP на базе ССКЦ СО РАН. Ожидается получить существенное ускорение по сравнению с реализацией алгоритма на кластере с MPP-архитектурой.

Работа поддержана проектом РФФИ №13-07-00589.

Литература

1. Караваев Д. Параллельная реализация метода численного моделирования волновых полей в трехмерных моделях неоднородных сред // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. – № 6 (1), С. 203–209.
2. Глинский Б., Караваев Д., Ковалевский В., Мартынов В. Численное моделирование и экспериментальные исследования грязевого вулкана «Гора Карабетова» вибросейсмическими методами // Вычислительные методы и программирование. Москва, 2010. Т. 11. С. 95-104.
3. Bihn M., Weiland T. A Stable Discretization scheme for the simulation of elastic waves // Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics (IMACS 1997). – Vol. 2, P. 75-80.

4. Drossaert F., Giannopoulos A. Complex frequency shifted convolution PML for FDTD modelling of elastic waves // Wave Motion. 2007. 44, P. 593–604.
5. Komatitsch D., Martin R. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation // Geophysics. 2008. Vol. 73, № 4. P. T51–T61.

ПАРАЛЛЕЛЬНЫЕ ОТКАЗОУСТОЙЧИВЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СТРУКТУРЫ

А.Б. Саркисов, М.И. Калмыков, Е.М. Яковлева

Северо-Кавказский федеральный университет, Ставрополь

Возрастание требований к временным характеристикам систем цифровой обработки сигналов (ЦОС) привело к созданию вычислительных устройств (ВУ), использующих параллельные вычисления. Однако при этом возникает следующая проблема: с одной стороны, постоянный рост требований к скоростным характеристикам ВУ приводит к необходимости организации параллельных вычислений, а с другой стороны, при этом увеличивается частота возникновения отказов. Решить данное противоречие можно за счет применения непозиционных арифметических кодов, позволяющих за счет независимой параллельной обработки данных в вычислительных трактах обеспечить получение корректного результата обработки сигналов в условиях воздействия помех при передаче и отказа оборудования.

Введение

Современный этап применения вычислительных устройств в системах передачи и обработки сигналов показал, что использование методов ЦОС позволяет относительно легко обеспечить высокую помехоустойчивость систем обработки данных, необходимую точность и разрешающую способность, простое сопряжение подсистем обработки, стабильность параметров тракта обработки информации и ряд других преимуществ [1-3]. При этом задачи цифровой обработки сигналов требуют выполнение в реальном масштабе времени больших объемов вычислений над большими массивами данных. Добиться качественных изменений в производительности современных систем передачи и обработки данных можно за счет применения новой математической модели ЦОС, поддерживающей параллельные вычисления. При этом такая технология ЦОС должна не только повысить скорость и точность обработки сигналов, но и обеспечить отказоустойчивость вычислительного устройства цифровой обработки сигналов.

Постановка задачи исследования

Постоянный рост требований к скоростным характеристикам вычислительных устройств приводит к необходимости не только совершенствования известных, но и разработке новых алгоритмов и структур. Высокие требования к производительности можно обеспечить только на основе нетрадиционных параллельных методов в вычислительных системах. Это приводит к активизации работ в области поиска новых архитектурных решений для систем обработки данных и новых методов организации параллельных вычислений. Особо наглядно это проявляется в системах цифровой обработки сигналов [2-4].

В подавляющем большинстве приложений задача ЦОС сводится к нахождению значений ортогонального преобразования конечной реализации сигнала для большого числа точек, что предопределяет повышенные требования к скорости обработки и разрядности вычислительного устройства. Решить данную проблему можно за счет перехода от одномерных вычислений к многомерным. В основу данного преобразования положена китайская теорема об остатках (КТО) [4, 5].

Для эффективной реализации ортогональных преобразований высокой точности необходимо доказать возможность реализации ДПФ в кольце полиномов. Пусть имеем кольцо полиномов $P(z)$ с коэффициентами в виде элементов поля GF(p), определяющего точность вычисления ортогональных преобразований сигналов. Положим, что кольцо разлагается в сумму

$$P(z) = P_1(z) + P_2(z) + ... + P_n(z), \quad (1)$$

где $P_l(z)$ – локальное кольцо полиномов, образованное неприводимым полиномом $p_l(z)$ над полем $GF(p)$; $l = 1, ..., n$.

Тогда в данной системе существует ортогональное преобразование, представляющее собой обобщенное ДПФ, если выполняются условия:

1) $\beta_i(z)$ – первообразный элемент порядка d для локального кольца $P_l(z)$;

2) d имеет мультипликативный обратный элемент d^*.

Ортогональное преобразование является обобщенным ДПФ для кольца вычетов $P(z)$ если существуют преобразования над конечным кольцом $P_l(z)$

$$X_l^k(z) = \sum_{n=0}^{d-1} x_l^n(z)\beta_l^{kn}(z), \quad (2)$$

где $\{ X_l^k(z), x_l^n(z), \beta_l^{kn}(z) \} \in P_l(z)$, l=1,2,...,m; k=0,1,...d–1.

Полученная циклическая группа имеет порядок d. Поэтому ДПФ над $P_l(z)$ можно обобщить над кольцом $P(z)$, если конечное кольцо $P_l(z)$ содержит корень d-ой степени из единицы и d имеет мультипликативный обратный элемент d^*, такой что справедливо

$$d^* d = p^v - 1. \quad (3)$$

Представленная математическая модель цифровой обработки сигналов использует модулярный полиномиальный код (МПК). При этом вычисления организуются параллельно, помодульно и независимо друг от друга, т.е. для суммы, разности и произведения двух полиномов $A(z)$ и $B(z)$, имеющих соответственно модулярные коды $(\alpha_1(z), \alpha_2(z),..., \alpha_n(z))$ и $(\beta_1(z), \beta_2(z),..., \beta_n(z))$ справедливы соотношения при i=1, ... , n

$$\left| A(z) \otimes B(z) \right|^+_{p(z)} = \left| \alpha_i(z) \otimes \beta_i(z) \right|^+_{p_i(z)}, \quad (4)$$

где $\otimes$ – операции сложения, вычитания и умножения в поле Галуа.

Тогда ортогональное преобразование сигнала и ему обратное определяются

$$(X_1(l), ..., X_n(l)) = \left(\sum_{j==0}^{d-1} x_1(j)\beta_1^{jl}, ..., \sum_{j=0}^{d-1} x_n(j)\beta_n^{jl} \right), \quad (5)$$

$$(x_1(j), ..., x_n(j)) = \left(d^* \sum_{l==0}^{d-1} X_1(l)\beta_1^{-jl}, ..., d^* \sum_{l=0}^{d-1} X_n(l)\beta_n^{-jl} \right). \quad (6)$$

При этом справедливо

$$\begin{aligned} &x_i(j) \equiv x(j) \bmod p_i(z); \ \beta_i^{\pm jl} \equiv \beta^{\pm jl} \bmod p_i(z); \\ &X_i(l) \equiv X(l) \bmod p_i(z). \end{aligned} \quad (7)$$

Приравнивая соответствующие координаты, получаем n пар прямого преобразования

$$\begin{cases} X_1(l) = \sum_{j=0}^{d-1} x_1(j)\beta_1^{jl} \bmod p_1(z) \\ \vdots \\ X_n(l) = \sum_{j=0}^{d-1} x_n(j)\beta_n^{jl} \bmod p_n(z) \end{cases}, \quad (8)$$

и n пар обратного преобразования

$$\begin{cases} x_1(j) = d^* \sum_{l=0}^{d-1} X_1(l)\beta_1^{-jl} \bmod p_1(z) \\ \vdots \\ x_n(j) = d^* \sum_{l=0}^{d-1} X_n(l)\beta_n^{-jl} \bmod p_n(z) \end{cases}. \quad (9)$$

Применение выражений (8) и (9) позволяет свести вычисление ортогональных преобразований сигналов в поле Галуа над кольцом $P(z)$ к n независимым вычислениям, проводимым по модулям $p_i(z)$ кода МПК. Повысить скорость обработки сигналов можно за счет перехода к быстрым алгоритмам, которые используют матрицы меньшей размерности.

При этом применение модулярного полиномиального кода позволяет не только повысить скорость обработки данных, но и восстановить искаженные результаты, которые возникают из-за отказов вычислительного устройства ЦОС [5].

Если на диапазон возможного изменения кодируемого множества полиномов наложить ограничения, то есть выбрать k из n оснований МПК ($k < n$), то это позволит разбить полный диапазон *P(z)* поля Галуа $GF(p^v)$ на два непересекающихся подмножества [3-5]. Первое подмножество называется рабочим диапазоном и задается

$$P_{раб}(z) = \prod_{i=1}^{k} p_i(z). \quad (10)$$

При этом второе подмножество определяется произведением $r = n - k$ контрольных модулей

$$P_{конт}(z) = \prod_{i=k+1}^{k+r} p_i(z). \quad (11)$$

Многочлен $A(z)$ с коэффициентами из поля $GF(p)$ будет считаться разрешенным, в том и только том случае, если $\deg A(z) < \deg P_{раб}(z)$. В противном случае $A(z)$, представленный к МПК, считается ошибочным.

Отсутствие взаимосвязи между вычислительными трактами процессоров МПК не позволяет ошибкам перемещаться по другим основаниям. При этом изменение кратности ошибки не приводит к размножению ошибки как между разрядами внутри основания, так и от одного основания к другому.

В работе [2] проведены исследования корректирующих способностей модулярных полиномиальных кодов. Как показали исследования, для исправления однократной ошибки необходимо в состав упорядоченного МПК, для которого справедливо $\deg p_1(z) \le \deg p_2(z) \ldots \le \deg p_k(z)$, ввести два контрольных оснований $p_{k+1}(z)$ и $p_{k+2}(z)$, удовлетворяющих условию

$$\deg p_{k-1}(z) + \deg p_k(z) \le \deg p_{k+1}(z) + \deg p_{k+2}(z). \quad (12)$$

Пусть ошибка произошла по модулю $p_i(z)$. В этом случае ошибка изменяет значение остатка $\alpha_i(z)$ на величину $\Delta\alpha_i(z)$, так, что получается новое значение $\alpha_i^*(z) = \alpha_i(z) + \Delta\alpha_i(z)$. При этом правильный полином $A(z) = (\alpha_1(z), \ldots, \alpha_i(z), \ldots, \alpha_{k+2}(z))$, принадлежащий рабочему диапазону, преобразуется в запрещенный полином $A^*(z) = (\alpha_1(z), \ldots, \alpha_i^*(z), \ldots \alpha_{k+2}(z))$, лежащий вне рабочего диапазона. Таким образом, зная номер интервала, куда попал искаженный полином $A^*(z)$, можно определить основание и глубину этой ошибки.

Согласно китайской теореме об остатках, используемой для преобразования из модулярного кода в позиционный код, значение ошибочного полинома $A^*(z)$ в этом случае определяется выражением

$$A^*(z) = \sum_{i=1}^{k+2} \alpha_i(z) B_i(z) \bmod P(z) = A(z) + \left| \Delta\alpha_j(z) B_j(z) \right|^{+}_{P_{пол}(z)}, \quad (13)$$

где $B_i(z)$ – ортогональный базис i-го основания МПК.

Анализ (13) показывает, что местоположение ошибочного полинома $A^*(z)$ относительно рабочего диапазона $P_{раб}(z)$ определяется величиной второго слагаемого.

Данное свойство модулярных кодов и предопределило повышенный интерес разработчиков к позиционной характеристике – интервальному номеру полинома $l(z)$ [5]. Процесс определения данной характеристики осуществляется согласно

$$l_{инт}(z) = \left\lfloor A(z) / P_{раб}(z) \right\rfloor. \quad (14)$$

Несмотря на то, что процедура (14) относится к немодульным, ее сводят к совокупности модульных операций. В работе [6] представлено устройство, осуществляющее обнаружение и коррекцию ошибки в модулярном коде на основе вычисления интервального номера. В основу данного алгоритма положено свойство подобия ортогональных базисов полной, содержащей контрольные основания, и безизбыточной системы МПК, согласно которому

$$B_i^*(z) \equiv B_i(z) \bmod P_{раб}(z), \quad (15)$$

где $B_i^*(z)$ и $B_i(z)$ - ортогональные базисы безизбыточной и полной системы.

Тогда согласно (15) справедливо

$$B_i(z) \equiv R_i(z) P_{раб}(z) + B_i^*(z), \quad (16)$$

где $R_i(z) = \left\lfloor B_i(z) \big/ P_{раб}(z) \right\rfloor$.

Подставив последнее равенство в выражение (13) получаем

$$l_{инт}(z) = \sum_{i=1}^{k+2} \alpha_i(z) (R_i(z) P_{раб}(z) + B_i^*(z)) + K(z) P_{полн}(z)) \big/ P_{раб}(z), \quad (17)$$

где $K(z)$ – ранг полной системы оснований МПК.

Проведя упрощения, имеем

$$l_{инт}(z) = \sum_{i=1}^{k+2} \alpha_i(z) R_i(z) + \left[\sum_{j=1}^{k} \alpha_j(z) B_j^*(z) \Big/ P_{раб}(z) \right] + K(z) P_{полн}(z)) \big/ P_{раб}(z). \quad (18)$$

Так как множество значений интервального номера $l_{инт}(z)$ представляет собой кольцо по модулю $P_{конт}(z) = \prod_{i=k+1}^{k+2} p_i(z)$, то выражение (18) имеет вид

$$l_{инт}(z) = \left| \sum_{i=1}^{k+r} \alpha_i(z) R_i(z) + K^*(z) \right|^{+}_{P_{конт}(z)}, \quad (19)$$

где $K^*(z)$ - ранг безизбыточной системы определяется выражением

$$K^*(z) = \left[\sum_{j=1}^{k} \alpha_j(z) B_j^*(z) \Big/ P_{раб}(z) \right]. \quad (20)$$

Следовательно, если $l_{инт}(z) = 0$, то исходный полином $A(z)$ лежит внутри рабочего диапазона и не является запрещенным. В противном случае $A(z)$ – ошибочная комбинация. Пусть задан модулярный полиномиальный код, который имеет рабочие основания $p_1(z)=z+1$; $p_2(z)=z^2+z+1$; $p_3(z)=z^4+z^3+z^2+z+1$. В качестве контрольных оснований используются полиномы $p_4(z)=z^4+z^3+1$; $p_5(z)=z^4+z+1$, которые удовлетворяют условию (12). Тогда рабочий диапазон равен $P_{раб}(z) = \prod_{i=1}^{3} p_i(z) = z^7 + z^6 + z^5 + z^2 + z + 1$. В таблице 1

представлены номера интервалов, в которые попадают ошибочные полиномы $A_l^*(z)$ при возникновении однократной ошибки по основаниям МПК.

Таблица 1. – Распределение однократных ошибок кода МПК

Основание МПК	Глубина $\Delta\alpha_i(z)$	Интервал, представленный в полиномиальной форме
$p_1(z)=z+1$	1	$z^7+z^4+z^2+z$
$p_2(z)=z^2+z+1$	1	$z^7+z^5+z^2+z+1$
	z	$z^7+z^6+z^5+z^4+z^2$
$p_3(z)=z^4+z^3+z^2+z+1$	1	$z^7+z^4+z^3+z+1$
	z	z^7+z^3+z+1
	z^2	z^7+z^5
	z^3	$z^7+z^6+z^5+z^4+z^3+z+1$
$p_4(z)=z^4+z^3+1$	1	$z^7+z^4+z^3$
	z	z^7+z^3+z+1
	z^2	$z^7+z^5+z^3+z^2$
	z^3	$z^7+z^6+z^5+1$
$p_5(z)=z^4+z+1$	1	z^5+z^4+z
	z	$z^6+z^3+z^2$
	z^2	$z^7+z^6+z^3$
	z^3	z^5+z^3+z+1

Анализ таблицы показывает, что ошибка переводит разрешенную модулярную комбинацию в соответствующий интервал полного диапазона. Очевидно, что использование двух контрольных оснований, удовлетворяющих (12), позволяет по величине $l_{инт}(z)$ определить местоположение и глубину $\Delta\alpha_i(z)$ ошибки. При этом такой избыточный модулярный код способен исправить более 90 процентов двукратных ошибок.

Заключение

Применение модулярного полиномиального кода позволяет повысить точность и скорость обработки сигналов за счет выполнения арифметических операций над малоразрядными остатками. Кроме этого использование модулярного полиномиального кода позволяет обеспечить коррекцию искаженного результата в условиях воздействия помех при передаче и отказа оборудования.

Литература

1. Червяков Н.И., Калмыков И.А., Галкина В.А., Щелкунова Ю.О., Шилов А.А. Элементы компьютерной математики и нейроноинфроматики – М.: Физматлит, 2003. – 216 с.
2. Калмыков И.А. Математические модели нейросетевых отказоустойчивых вычислительных средств, функционирующих в полиномиальной системе классов вычетов / Под ред. Н.И. Червякова – М: Физматлит, 2005. – 276 с.
3. Калмыков И.А., Зиновьев А.В., Емарлукова Я.В. Высокоскоростные систолические отказоустойчивые процессоры цифровой обработки сигналов для инфокоммуникационных систем // Инфокоммуникационные технологии. 2009. Том 7, № 2. С. 31–37.
4. Чипига А.Ф., Калмыков М.И., Яковлева Е.М. Применение полиномиальной системы классов вычетов в схеме разделения секрета // Материалы международной НПК

«Информационная безопасность 2012», Таганрог: ТРТУ, 25-29 июня, 2012. Ч.2. С. 324-332.
5. Калмыков И.А.,. Саркисов А.Б., Яковлева Е.М., Калмыков М.И. Модулярный систолический процессор цифровой обработки сигналов с реконфигурируемой структурой // Вестник Северо-Кавказского федерального университета. – 2013. – Вып. 2. – С. 30-34.

ПАРАЛЛЕЛЬНЫЕ ТЕХНОЛОГИИ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ НА ОСНОВЕ НЕПОЗИЦИОННЫХ МОДУЛЯРНЫХ КОДОВ

А.Б. Саркисов, М.И. Калмыков, Е.М. Яковлева

Северо-Кавказский федеральный университет, Ставрополь

Главной целью исследований является разработка параллельной технологии цифровой обработки сигналов, применение которой позволяет, за счет использования целочисленной непозиционной математической модели ЦОС, обеспечить высокоскоростную обработку сигналов в условиях воздействия помех и отказа оборудования. Для решения данной проблемы в работе предлагается использовать математические модели ЦОС, использующие непозиционные модулярные коды, которые за счет распараллеливания на уровне операций и обработки малоразрядных данных позволяют не только увеличить скорость вычислений, но и обеспечивает получение корректного результата в условиях воздействия помех при передаче и отказа оборудования.

Введение

Цифровая обработка сигналов относится к числу наиболее динамично развивающихся областей информационных технологий. При этом задачи цифровой обработки сигналов требуют выполнения в реальном масштабе времени больших объемов вычислений над большими массивами данных. Добиться качественных изменений в возможностях современных систем передачи и обработки данных можно за счет применения параллельных технологий ЦОС, т.е. за счет применения новой математической модели реализации ортогональных преобразований сигналов в алгебраических модульных системах. При этом такая технология ЦОС должна не только повысить скорость и точность обработки сигналов, но и обеспечить отказоустойчивость вычислительного устройства цифровой обработки сигналов.

Постановка задачи исследования

Важность задач ЦОС делает целесообразной разработку специализированных устройств для их решения. При этом эффективность работы системы цифровой обработки сигналов во многом определяется математической моделью ЦОС.

В настоящее время все технические реализации ВУ ЦОС используют несколько математических моделей, которые можно разделить на следующие группы. Основу первой составляют математические модели, базирующиеся на реализации ортогональных преобразований сигналов над полем комплексных чисел, в частности дискретном преобразовании Фурье (ДПФ) и быстром преобразовании Фурье (БПФ). Так в системах широкополосного беспроводного доступа (ШБД) для борьбы с помехами при многолучевом приеме применяется технология ортогонального частотного мультиплексирования OFDM. Технически метод OFDM реализуется путем выполнения обратного дискретного преобразования Фурье (Fast Fourier Transform, FFT) в модуляторе передатчика и прямого дискретного преобразования Фурье – в демодуляторе приемника приемопередающего устройства [1-3].

Для повышения скорости реализации таких ортогональных преобразований в ряде работ [4-6] предлагается использовать систему остаточных классов (СОК). В этом слу-

чае число A представляется в виде совокупности остатков, полученных путем его деления на попарно взаимно простые модули p_i,

$$A = (\alpha_1, \alpha_2, ..., \alpha_n), \quad (1)$$

где $A \equiv \alpha_i \bmod p_i(z)$; $i = 1,2,...,n$.

Основным достоинством СОК является высокая скорость выполнения модульных операций, к которым относятся сложение, вычитание и умножение. Если два числа A и B представлены в непозиционном модулярном коде $A = (\alpha_1, \alpha_2, ..., \alpha_n)$ и $B = (\beta_1, \beta_2, ..., \beta_n)$, то операции сложения, вычитания и умножения можно свести к соответствующим операциям над остатками

$$A + B = ((\alpha_1 + \beta_1) \bmod p_1, (\alpha_2 + \beta_2) \bmod p_2, \ldots, (\alpha_n + \beta_n) \bmod p_n). \quad (2)$$

$$A - B = ((\alpha_1 - \beta_1) \bmod p_1, (\alpha_2 - \beta_2) \bmod p_2, \ldots, (\alpha_n - \beta_n) \bmod p_n). \quad (3)$$

$$A \cdot B = ((\alpha_1 \cdot \beta_1) \bmod p_1, (\alpha_2 \cdot \beta_2) \bmod p_2, \ldots, (\alpha_n \cdot \beta_n) \bmod p_n). \quad (4)$$

Тогда, используя систему остаточных классов, можно реализовать ортогональные преобразования сигналов в виде n параллельно выполняемых вычислений ДПФ по основаниям СОК

$$\begin{cases} X(k) \bmod p_1 = \left| \sum_{l=0}^{N-1} \left| x(l) \right|_{p_1}^{+} \cdot \left| W^{lk} \right|_{p_1}^{+} \right|_{p_1}^{+} \\ \vdots \\ X(k) \bmod p_n = \left| \sum_{l=0}^{N-1} \left| x(l) \right|_{p_n}^{+} \cdot \left| W^{lk} \right|_{p_n}^{+} \right|_{p_{n1}}^{+} \end{cases}. \quad (5)$$

где W^{lk} – поворачивающий коэффициент; k =0,1,2,…, N–1; $N = 2^v$.

Для выполнения выражения (5) требуется схемная реализация спецпроцессора СОК, показанная на рис. 1.

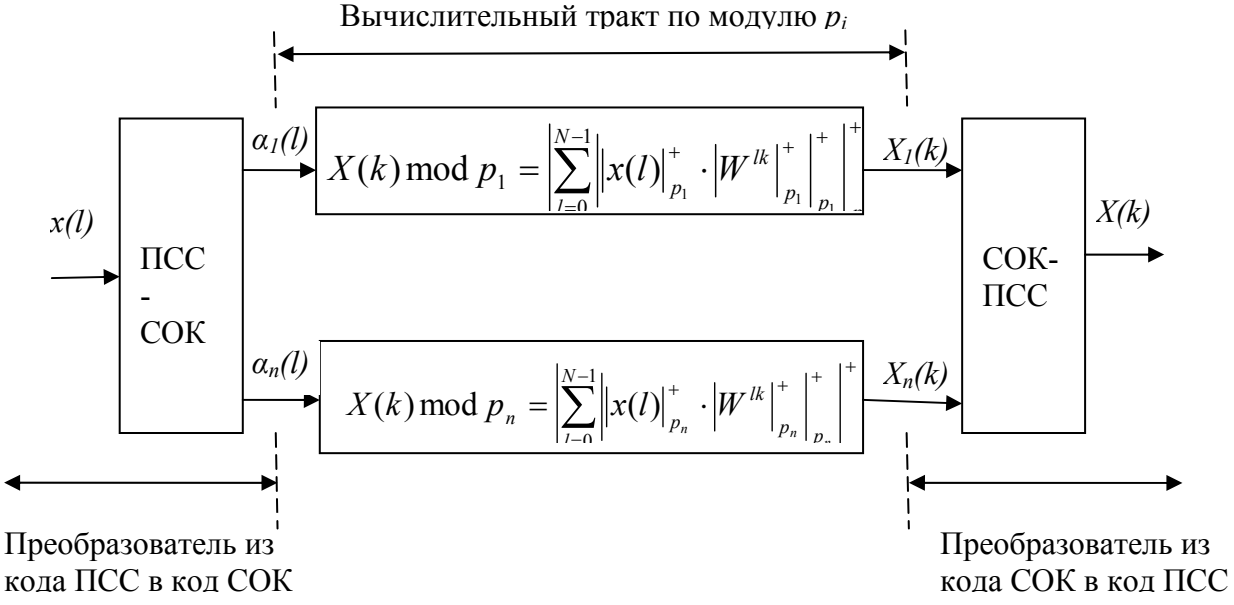


Рис. 1. Структура параллельного спецпроцессора СОК

Характерной чертой такого спецпроцессора является наличие n вычислительных трактов, которые функционируют параллельно и независимо от друг от друга. В каждом из них реализуются ортогональные преобразования сигналов по соответствующему модулю. Распараллеливание на уровне операции позволяет отнести данный спецпроцессор к виду SIMD. Используя один поток команд, который одновременно посту-

пает во все вычислительные тракты, вычислительное устройство реализует ортогональные преобразования согласно (5). Малоразрядность остатков, отсутствие обмена данных между вычислительными трактами, позволяет повысить производительность спецпроцессора по сравнению с классической архитектурой сигнальных процессоров. При этом для повышения точности вычислений достаточно увеличить число вычислительных трактов, т.е. увеличить число оснований СОК.

Следует отметить, что схемная реализация вычислительного тракта может быть построена в соответствии с выбранным параллельным алгоритмом ЦОС. Так при использовании параллельно-конвейерного алгоритма БПФ выражение (5) принимает вид

$$X(k) \bmod p_i = \left\| \sum_{n=0}^{N/2-1} x_{v-1,0}(n) W_N^{2nk} \right|_{p_i}^{+} + \left| \sum_{n=0}^{N/2-1} x_{v-1,1}(n) W_N^{2(n+1)k} \right|_{p_i}^{+} \Bigg|_{p_i}^{+}, \quad (6)$$

где $x_{v-1,0}(n) = x(2nT), x_{v-1,1}(n) = x((2n+1)T)$ – соответственно последовательности с четными и нечетными номерами; $W_N^2 = e^{-\frac{2\pi}{N/2}}$.

Наряду с модульными операциями для реализации параллельных вычислений в непозиционных кодах необходимо выполнять и немодульные операции. К таким можно отнести:

– прямое преобразование из позиционной системы счисления (ПСС) в код СОК;

– прямое преобразование из СОК в код позиционной системы счисления;

– операцию масштабирования;

– вычисление позиционных характеристик для реализации процедур поиска и коррекции ошибок.

Для любого параллельного спецпроцессора ЦОС, функционирующего в СОК, одной из обязательных немодульных операций является прямое преобразование из ПСС в код СОК. Так как операция деления в СОК не определена, то вычисление остатка принято сводить к совокупности модульных операций. Поэтому для высокоскоростной системы ЦОС целесообразно выбрать такой алгоритм перевода, который бы в наибольшей степени соответствовал предъявляемым требованиям с точки зрения обеспечения минимальных временных затрат. Кроме того, данный алгоритм должен быть оптимальным для микроэлектронной реализации.

В настоящее время известно несколько алгоритмов, реализующих эту операцию. Проведенный анализ работ [4-8] позволил разделить на несколько групп. Основу первой группы составляют алгоритмы, использующие метод понижения разрядности, который сводится к совокупности операций

$$A_1 = A_k \cdot C_k + A_{k-1} \cdot C_{k-1} + ... + A_1 \cdot C_1 + A_0 \cdot C_0 < A_k \cdot S^k + ... + A_1 \cdot S^1 + A_0 . \quad (7)$$

С целью повышения эффективности данного устройства было предложено использовать конвейерную организацию вычислений остатка. В работе [6] предложена структура преобразователя ПСС-СОК, реализующего прямое преобразование позиционного двоичного кода в код класса вычетов, на основе конвейерной организации. Число ступеней в таком конвейере определяется количеством итераций l, необходимых для преобразования входных данных в код СОК. В этом случае итеративный алгоритм преобразования A по модулю p определяется

$$A(l+1) = \sum_{i=0}^{\deg A(l)} \left| 2^i \right|_p^{+} \left\| \left[\frac{A(l)}{2^i} \right] \right\|_2^{+}, \quad (8)$$

где $l = 0,1,2,...$ – число итераций.

Наряду с отмеченными ранее методами и алгоритмами вычисления остатков α_i по значению входного двоичного вектора $A(z)$ данная процедура может быть реализована

на основе метода непосредственного суммирования [4-6]. Для осуществления такого преобразования необходимо вычислитель значения констант, являющихся эквивалентами степеней оснований 2^i, и коэффициентов при соответствующих степенях оснований a_i, где $i = 1, 2, ..., n$, представленных в системе класса вычетов. Тогда справедливо

$$\alpha_i = \sum_{j=0}^{v-1} (a_j 2^j) \bmod \ p_i, \tag{9}$$

где $i = 1,2,3,..,n$; $v = \lceil \log_2 p_i \rceil$.

Проведенные исследования показали, что меньшими временными затратами обладают алгоритмы, базирующиеся на методе непосредственного суммирования.

Второй обязательной процедурой является преобразование из кода СОК в позиционный код. В настоящее время при реализации непозиционных вычислительных устройств наибольшее распространение получил способ обратного перевода, базирующийся на китайской теореме об остатках [4-6]. Применение данной теоремы позволяет выполнить обратное преобразование модулярный код – позиционный код с использованием ортогональных базисов.

В этом случае задача перевода представляется следующим образом: необходимо для заданного набора модулей p_i, $i = 1,2,...,n$, осуществить преобразование n-мерного образа $A = (\alpha_1, \alpha_2, ..., \alpha_n)$ в систему с основанием $P = \prod_{i=1}^{n} p_i$ так, чтобы выполнилось условие

$$A = \alpha_1 B_1 + \alpha_2 B_2 + ... + \alpha_n B_n - r_A P, \tag{10}$$

где B_i – ортогональные базисы системы ПСКВ; r_A – ранг числа.

Чтобы вычислить значения ортогональных базисов для полиномиальной системы классов вычетов необходимо воспользоваться условием

$$B_i = \begin{cases} 0 \bmod p_j, j \neq i \\ 1 \bmod p_j, j = i \end{cases}, \tag{11}$$

где $B_i = m_i \prod_{\substack{j=1 \\ j \neq i}}^{n} p_j$; m_i – вес ортогонального базиса, такой что

$$m_i \prod_{\substack{j=1 \\ j \neq i}}^{n} p_j \equiv 1 \bmod p_i. \tag{12}$$

Проведенные исследования показали, что преобразователь СОК-ПСС обеспечивает минимальную временную задержку, связанную с перекодировкой результата, которая определяется выражением

$$T_{СОК-ПСС} = 2\tau_з \left(1 + \frac{8Q_{СОК-ПСС}}{4}\right), \tag{13}$$

где $Q_{сок\text{-}псс}$ – количество разрядов выходного вектора отсчетов.

Обобщая сказанное выше можно сделать заключение, что реализации ортогональных преобразований с использованием математической модели ЦОС в кольце полиномов будет задаваться выражением

$$T_{СП}^{СОК} = T_{ПСС-ПСКВ} + T_{ортг} + T_{ПСКВ-ПСС}. \tag{14}$$

Так как алгоритм БПФ представляет собой рекурсивный алгоритм, то для его реализации необходимо провести совершенную тасовку, которая используется для передачи данных между этапами вычислений. Следовательно, перед началом вычислений БПФ необходимо получить все N отсчетов и произвести их перестановку. Кроме того, после вычисления спектральных составляющих требуется N тактов, необходимых для

выгрузки N комплексных отсчетов вычисленных коэффициентов Фурье входного массива данных. Тогда полное время выполнения БПФ будет составлять

$$T_{БПФ}^{полн} = \lceil \log_2 N \rceil \left(\left(1 + \frac{3Q_{ПСС}}{4} \right) 2\tau_з + 8Q_{ПСС}\tau_з + 7\tau_з \right) + 2NT_{загр}, \quad (15)$$

где $T_{загр}$ – длительность такта загрузки/выгрузки.

Подставив равенство (14) в равенство (13), получаем выражение, с помощью которого можно оценить временные затраты, необходимые на реализацию ортогонального преобразования сигналов в СОК. Сравнительный анализ времени выполнения БПФ в позиционной системе (Т1) и с использованием параллельных вычислений (Т2) показан на рис. 2. Для оценки был выбран относительный коэффициент $К=T_{ПСС}/T_{СОК}$.

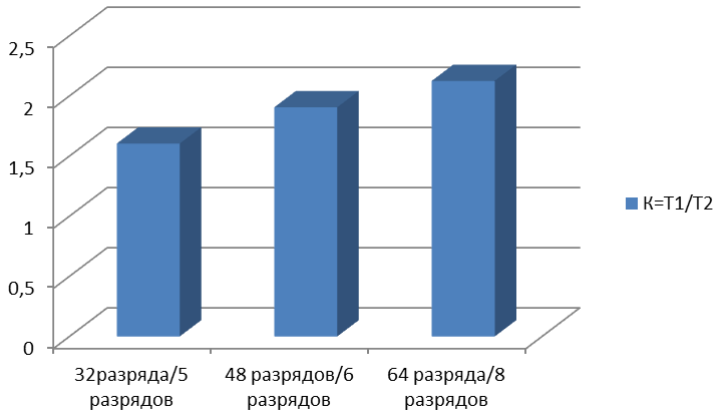


Рис. 2. Сравнительный анализ времени выполнения БПФ в ПСС (Т1) и в СОК (Т2)

Анализ рис. 2 показывает, что применение СОК позволяет повысить скорость цифровой обработки сигналов с использованием алгоритма БПФ более чем в 1,5 раза. Причем при увеличении разрядности обрабатываемых данных выигрыш возрастает. Так при обработке 64 разрядных данных скорость непозиционного процессора более чем в 2 раза превосходит скорость работы позиционного СП ЦОС, даже с учетом необходимости выполнения операций прямого и обратного преобразований кодов.

Заключение

В работе показана целесообразность использования параллельных вычислений для реализации ЦОС. Применение СОК за счет независимой параллельной обработки малоразрядных остатков позволяет сократить временные затраты на выполнение БПФ по сравнению с позиционным спецпроцессором цифровой обработки сигналов.

Литература

1. Rohde & Schwarz. R&S FSQ-K96 OFDM Vector Signal Analysis with the R&S FSQ Signal Analyzer. Product Brochure, V. 1.00, March 2008.
2. Yong Soo Cho, Jaekwon Kim, Won, Young, Chung G. Kang MIMO-OFDM Wireless Communications with MATLAB, – Wiley, 2010.
3. Калмыков И.А., Зиновьев А.В., Емарлукова Я.В. Высокоскоростные систолические отказоустойчивые процессоры цифровой обработки сигналов для инфокоммуникационных систем // Инфокоммуникационные технологии. 2009. Том 7, № 2. С. 31–374.
4. Червяков Н.И., Калмыков И.А., Галкина В.А., Щелкунова Ю.О., Шилов А.А. Элементы компьютерной математики и нейроноинфроматики. – М.: Физматлит, 2003. – 216 с.

5. Hosseinzadeh M., Navi K., Gorgin S. A New moduli set for residue number system. Electrical Engineering, 2007. ICEE'07. International Conference on. 11–12 April 2007, P. 1–6.
6. Калмыков И.А. Математические модели нейросетевых отказоустойчивых вычислительных средств, функционирующих в полиномиальной системе классов вычетов / Под ред. Н.И. Червякова – М: Физматлит, 2005. – 276 с.

О РЕАЛИЗАЦИИ АЛГОРИТМА ПОИСКА ЭКСТРЕМУМА В КЛАССАХ ФУНКЦИЙ, ОПРЕДЕЛЯЕМЫХ КУСОЧНО-ЛИНЕЙНОЙ МАЖОРАНТОЙ

В.М. Сморякова

Нижегородский госуниверситет им. Н.И. Лобачевского

Эффективность методов поиска экстремума функций многих переменных во многом определяется априорной информацией о функции. При этом на основе имеющейся информации можно сформулировать свойства функций, которые порождают класс, допускающий построение алгоритма, удовлетворяющего заданным требованиям. К числу таких требований может быть отнесена простота в реализации алгоритма, оценка погрешности в определении экстремума.

В данной работе рассматривается задача построения такого рода алгоритма для класса многоэкстремальных функций, определяемого кусочно-линейной мажорантой. Данный класс содержит вогнутые, выпуклые и удовлетворяющие условию Липшица функции и замкнут относительно ряда естественных операций таких, как взятие минимума, максимума, суммирования с неотрицательными коэффициентами по конечному набору функций.

При этом основной процедурой такого рода алгоритмов является метод поиска экстремума функций одной переменной. Предлагается простой в реализации алгоритм поиска экстремума в классе функций одной переменной, порождаемом описанным выше классом. При этом для данного алгоритма приводятся оценки погрешности в определении наибольшего значения.

Приводятся результаты вычислительного эксперимента, в котором построенный алгоритм выступает в качестве вспомогательной процедуры в компонентном методе поиска экстремума функций многих переменных.

Литература

1. Коротченко А.Г., Сморякова В.М. Алгоритм поиска наибольшего значения в классе функций, определяемом кусочно-линейной мажорантой // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы XII Всероссийской конференции. Нижний Новгород: Изд-во Нижегородского университета, 2012. С. 210-211.
2. Коротченко А.Г., Сморякова В.М. Об оценке погрешности алгоритмов поиска экстремума в классах функций, определяемых кусочно-линейной мажорантой // Вестник нижегородского университета им. Н.И. Лобачевского. 2013. №3. С. 188-194.

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО МНОГОУРОВНЕВОГО АЛГОРИТМА РАВНОМЕРНОГО РАЗБИЕНИЯ НЕРАСПРЕДЕЛЕННОГО ГРАФА

Н.В. Старостин, А.В. Филимонов

Нижегородский госуниверситет им. Н.И. Лобачевского

В работе формально поставлена задача равномерного k-разбиения графа, предложен адаптивный параллельный алгоритм решения задачи для случая нераспределенного графа, базирующийся на многоуровневой парадигме, представлены результаты вычислительных экспериментов для серии тестовых графов.

Введение

Численное решение задач математической физики на распределенных ВС предполагает использование распределенной между вычислительными узлами сетки, аппроксимирующей расчетную область. При этом каждый вычислительный узел обрабатывает данные на своей части сетки. Эффективность параллельных вычислений в значительной мере зависит от объёма межпроцессорных коммуникаций и сбалансированности загрузки вычислительных узлов. Выбор «хорошего» варианта декомпозиции сетки может привести к значительному ускорению параллельных расчетов на распределенных данных. Задача сбалансированного разбиения сетки по узлам вычислительной системы сводится к задаче сбалансированного k-разбиения графа, в которой требуется распределить вершины графа по заданному числу подграфов. Критерий задачи направлен на минимизацию внешних связей (коммуникация между вычислительными узлами). Под сбалансированностью понимаются по возможности равные веса подграфов разбиения (загрузка вычислительных узлов).

Задача k-разбиения графа

Рассматривается взвешенный неориентированный граф $G(V,E,w,u)$, где $V=\{v_1,...,v_n\}$ – множество вершин; $E\subseteq V^{(2)}$ – множество ребер; $w:V\to N$ отображение, приписывающее натуральный «вес» каждой вершине; $u:E\to N$ отображение, указывающее натуральный «вес» каждому ребру графа. Задача *k*-разбиения графа $G(V,E,w,u)$ состоит в распределении всех вершин из множества V по k подмножествам $V_1,...,V_k$, таким образом, чтобы удовлетворялось ограничение (1), а оптимизируемая функция (2) принимала экстремальное значение

$$\max_{i=1,k} W(V_i) \le c\frac{W(V)}{k},$$

где $W(S)=\sum_{v\in S} w(v)$, $S\subseteq V$, (1)

$c\in[1,\infty)$ – параметр разбиения.

Часто в задаче разбиения графа вместо параметра c используют параметр $\varepsilon = 100\cdot(c-1)$, который называют дисбалансом (с англ., imbalance).

$$F(V_1,...,V_k) = \sum_{e \in Q(V_1,...,V_k)} u(e) \to \min, \tag{2}$$

где $Q(V_1,...,V_k) = \left\{ \{v_i, v_j\} \in E \mid v_i \in V_l,\ v_j \in V_j, i = \overline{1,k}, j = \overline{1,k}, i \neq j \right\}$.

Задачу (1), (2) разбиения графа часто обозначают (k,c) задачей разбиения. Задачу $(k,1)$ называют задачей сбалансированного *k*-разбиения. В общем случае (k,c)-задача принадлежит к классу NP-трудных задач.

Общая схема многоуровневого алгоритма

Ключевая идея многоуровневого алгоритма (ML) разбиения состоит в следующем: вместо того, чтобы строить k-разбиение непосредственно для исходного графа, сначала строится ряд его приближений (загрублений). Каждое загрубление уменьшает (редуцирует) размерность исходной задачи. Процесс редукции продолжается до тех пор, пока порядок графа не снизится до сотен или даже десятков. На графе таких размерностей и отыскивается так называемое начальное разбиение. Полученное решение используется для построения решения для исходной задачи. Это достигается путем выполнения серии переносов решения задачи меньшей размерности на задачу большей размерности с последующим улучшением перенесенного решения. Таким образом, работу многоуровневого алгоритма можно разделить на три фазы: первая – фаза загрубления, когда производится ряд последовательных редукций размерности задачи, вторая – фаза поиска начального разбиения и третья – фаза восстановления решения, когда производится серия последовательных переносов решения задачи меньшей размерности на менее редуцированную задачу с последующим улучшением спроецированного решения. Работа алгоритма для задачи *k*-разбиения показана на рис.1.

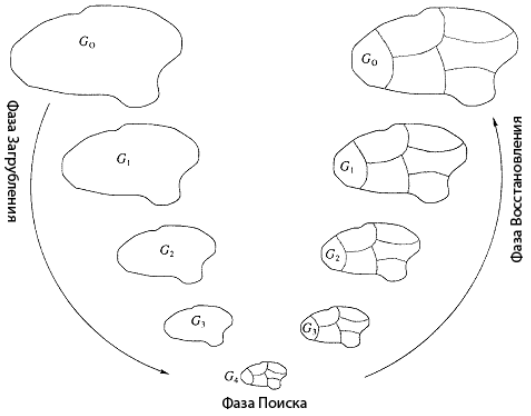


Рис. 1. Многоуровневый алгоритм 5-разбиения графа

Важным достоинством многоуровневых схем являются широкие возможности по их настройке под конкретные классы задач, так как они позволяют относительно просто интегрировать в свою структуру различные эффективные эвристические алгоритмы, направленные как на поиск сбалансированных разбиений, так и на улучшение получаемых решений. Подобные многоуровневые алгоритмы различаются: 1) стратегиями, используемыми для упрощения задачи и/или редукции графа; 2) алгоритмами, строящими начальное разбиение редуцированного графа; 3) эвристиками, используемыми для получения «удачной» проекции решения на этапе восстановления; 4) моделями, используемыми для настройки параметров многоуровневого алгоритма.

Структура многоуровневого итерационного алгоритма

По своей природе известные многоуровневые алгоритмы решения задачи (k,c)-разбиения ближе к конструктивным алгоритмам. На фазе загрубления используются стратегии, главная цель которых – получить относительно небольшой граф, который структурно «похож» на исходный граф. На фазе поиска используются конструктивные схемы генерации разбиений. Фаза восстановления использует улучшающие схемы, но они направлены исключительно на улучшение проекций начальных разбиений, полученных в фазе поиска.

Известные многоуровневые схемы позволяют «с нуля» строить новые решения. Эти алгоритмы можно использовать для генерации нескольких решений методом включения в общую схему элементов рандомизации либо изменением параметров алгоритма. Главный минус подобных схем в том, что они не обладают «обратной связью» – не используют информацию о предыдущих итерациях своей работы.

Предлагается многоуровневый итерационный алгоритм (MLI-алгоритм) задачи (k,c)-разбиения. Алгоритм на вход получает не только исходные данные задачи (G, k, c), но и одно из допустимых решений (k,c) задачи. Цель алгоритма – исследование областей вокруг начального решения и поиск лучшего решения.

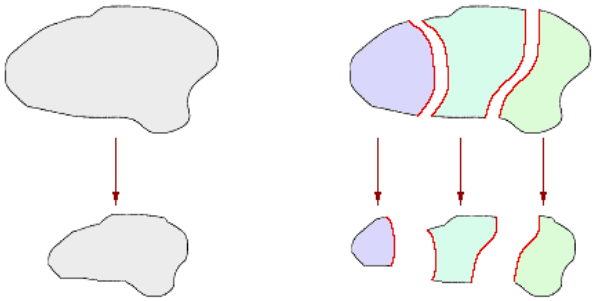


Рис. 2. Различия в концепциях загрубления классического (слева) и итерационного (справа) многоуровневых алгоритмов

Процесс работы MLI также включает три фазы. На первой фазе (фаза загрубления) алгоритм в отличии от MA осуществляет последовательную редукцию и графа (загрубление задачи), и начального разбиения (загрубление решения), если оно имеется. На рисунке 2 показаны отличия принципов работы фазы загрубления классического ML и MLI. Каждое загрубление уменьшает (редуцирует) размерность подграфа разбиения исходной задачи. При этом происходит уменьшение размерности исходного графа, а что самое важное – множество редуцированных подграфов разбиения определяет допустимое разбиение редуцированного графа.

Процесс редукции продолжается не более определенного числа раз (предельный уровень редукции). Процесс может прерываться в случае нехватки оперативной памяти, бесперспективности редукции или по ограничению MLI по времени работы.

Вторая фаза – фаза поиска решения самой загрубленной задачи. ML на этом этапе применяет две стратегии поиска лучшего решения. Первая стратегия заключается в локальном улучшении загрубленного начального решения. Вторая стратегия предполагает поиск нового решения загрубленной задачи. В завершении фазы поиска из двух найденных допустимых разбиений MLI выбирает лучшее с точки зрения критерия (2).

Третья фаза – фаза восстановления решения. Здесь MLI последовательно переходит от задачи более редуцированной к задаче менее редуцированной. При каждом переходе алгоритм восстанавливает и улучшает решение более редуцированной задачи. В этот момент MLI владеет парой решений менее редуцированной задачи: начальным загрубленным и улучшенным решением и найденным восстановленным и улучшенным реше-

нием. МА выбирает лучшее с точки зрения критерия (2) и переходит к следующей менее редуцированной задаче.

Описанная схема работает в итерационном режиме так: начальное решение улучшается MLI, найденное лучшее решение становится начальным для следующей итерации MLI. Итерационный процесс продолжается не более определенного числа раз (максимальное число итераций), и может прерываться в случае нехватки памяти, бесперспективности итерации или в связи с нехваткой времени на работу алгоритма.

В качестве алгоритма поиска начального решения в описанной схеме используется алгоритм рекурсивной бисекции, цель которого - из задачи k-разбиения графа получить серию более «простых» задач 2-разбиения графа, при этом в случае наличия начального решения исходной задачи для каждой задачи 2-разбиения графа формируется ее начальная бисекция. Решение каждой из задач бисекции происходит с помощью многоуровневого алгоритма бисекции

Анализ выполнения алгоритма показал, что наиболее ресурсоемкой процедурой, стоящей около 90% общего времени работы алгоритма, является именно бисекция. Поэтому наиболее эффективным с точки зрения сокращения времени работы алгоритма было бы построение параллельной реализации этой процедуры.

Параллельный вариант этапа рекурсивной бисекции

Декомпозиция рекурсии заключается в получении из одной задачи k-разбиения двух новых задач k_1-разбиения и k_2-разбиения, где $k = k_1 + k_2$, причем $|k_1 - k_2| \leq 1$. При этом из одной задачи две новые задачи получаются методом решения задачи несбалансированного 2-разбиения. Процесс многоуровневой рекурсии продолжается до тех пор, пока все задачи не будут решены, и как следствие, не будет решена задача k-разбиения. Получающиеся в процессе декомпозиции новые задачи можно решать независимо друг от друга. Верхняя оценка ускорения параллельного варианта алгоритма при независимом решении задач бисекции на всех уровнях рекурсии составит

$$S(k,p) = \frac{p\lfloor \log_2 k \rfloor}{2p - 2 + \lfloor \log_2 k \rfloor - \lceil \log_2 p \rceil}, \qquad (3)$$

здесь p – количество процессоров.

Вычислительный эксперимент

Испытывалась программная реализация предложенных методов решения задач k-декомпозиции под кодовым названием PARMATRUZ. Качество получаемых решений сравнивалось с качеством решений известной реализации параллельного многоуровневого алгоритма k-way в рамках библиотеки PARMETIS v.4.0.2.

Вычислительный эксперимент проводился на КС-ЭВМ 1,1 TFlops ОС Unix x64.

Выбиралась настройка параметров по умолчанию для программ PARMETIS и PARMATRUZ. Все решения отыскивались в пределах 5% дисбаланса.

Таблица 1. Сравнение параллельных версий программ на КС-ЭВМ

Граф	Вершины ребра	Подграфы	Ядер	PARMETIS	PARMATRUZ	
				Сечение	Сечение	Время (сек)
800835	526 941 15 78 501	32	1	11553	11552	20
			4	12627	11552	7
			8	12711	11552	6
800835		1024	1	85207	87052	164
			4	91827	87052	43
			8	92081	87052	25
trubka	2 428 323	32	1	1214582	1211023	509

	67 910 216		4	1231592	1211023	169
			8	1233087	1211023	129
trubka		1024	1	7145238	7232042	3360
			4	7161643	7232042	844
			8	7145126	7232042	508
grid_200x200x250	10^7 29 860 000	32	1	394805	400634	710
			4	418197	400634	245
			8	432961	400634	230
grid_200x200x250		1024	1	1589231	1681869	3881
			4	1801544	1681869	1103
			8	1784045	1681869	695

Выводы

По результатам проведенных экспериментов можно сформулировать выводы:

- MATRUZ продемонстрировал возможность находить решения, сравнимые по качеству, а в ряде случаев превосходящие, решения PARMETIS.
- MATRUZ способен решать задачи k-разбиения графов размером порядка 10^7 числом вершин и значением k порядка 10^4.
- Ускорение параллельной версии MATRUZ согласуется с теоретическими оценками.
- Число процессоров влияет на время работы алгоритма, но не оказывает влияния на качество работы MATRUZ.

СТРУКТУРЫ ДАННЫХ И ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ ДЛЯ ПОСТРОЕНИЯ АЖУРНЫХ СЕТОК КОНЕЧНЫХ ЭЛЕМЕНТОВ

В.Л. Тарасов, Д.Т. Чекмарев, П.Д. Чекмарев

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассмотрены вопросы эффективной организации процесса прореживания сплошных сеток тетраэдральных конечных элементов в ажурные. Предложено использовать упорядоченный контейнер для хранения списка инцидентности конечных элементов их ребрам. Для распараллеливания вычислений исходная область разбивается на части, топологически эквивалентные сферическим слоям. Приведены оценки времени работы различных этапов алгоритма.

Введение

Методы конечных элементов предполагают, как правило, что расчетная область заполняется конечными элементами сплошь – без промежутков и наложения друг на друга. В [1, 2] предложены ажурные численные схемы МКЭ решения трехмерных задач механики сплошных сред, в которых конечные элементы заполняют расчетную область с промежутками. Это позволяет более эффективно использовать информацию в узлах сетки о напряженно-деформированнном состоянии сред и избегать лишних вычислений при описании процессов деформирования без ущерба для точности получаемых численных решений. Теоретические обоснования подобных схем даны в работах [3-5].

Существующие расчетные комплексы, использующие МКЭ, позволяют строить сплошные сетки конечных элементов. В работе [6] рассмотрено построение изначально ажурных сеток.

В настоящей работе предложен алгоритм, позволяющий преобразовывать сплошные сетки конечных элементов в ажурные, и его реализация. Для создания эффективных структур данных использована стандартная библиотека шаблонов (STL) языка C++ [7, 8]. Многопоточность реализована с помощью библиотеки Qt [9].

1. Постановка задачи, основные этапы алгоритма

Имеется тетраэдральная сетка, сгенерированная сторонним сеточным генератором, задаваемая координатами узлов и четверками номеров узлов, определяющих конечные элементы сетки.

Каждая пара узлов, относящихся к некоторому элементу, образует ребро. Конечный элемент в виде тетраэдра имеет шесть ребер, которые могут принадлежать сразу нескольким элементам. Будем называть число конечных элементов, которым принадлежит ребро, его весом.

Необходимо исключить максимальное число элементов так, чтобы выполнялись условия аппроксимации ажурной схемы. Достаточным условием этого является сохранение всех ребер сетки. Это значит, что если два узла исходной сетки принадлежат одному или нескольким конечным элементам, в ажурной сетке они будут принадлежать хотя бы одному конечному элементу.

В качестве входных данных выступает список элементов с указанием принадлежащих ему узлов:

List[1.. ElementNum]({Node1, Node2, Node3, Node4});

Предлагаемый алгоритм прореживания конечноэлементной сетки включает следующие этапы:

1. Составление списка взвешенных ребер и списков инцидентности ребер элементам.
2. Разбиение расчетной области на части для выполнения прореживания в параллельном режиме.
3. Прореживание сетки.

2. Составление списка взвешенных ребер

Предлагаемый процесс прореживания сетки заключается в последовательном переборе всех элементов с принятием решения об удалении конкретного элемента. Элемент можно удалить, если при этом не будет удалено ни одного ребра сетки. Таким образом, возникает необходимость в создании списка взвешенных ребер и списка инцидентности ребер элементам.

Общий вид класса ребер:

```
class Rib
{
 unsigned int node1; // Узел ребра с меньшим номером
 unsigned int node2; // Узел ребра с большим номером
 vector<unsigned int> BTcells; // Массив элементов, инцидентных ребру
 void AddCell(unsigned int cell); // Добавление элемента с номером cell
 bool operator==(const Rib& with)const;
 bool operator!=(const Rib& with)const;
 bool operator<(const Rib& with)const;
};
```

Необходимо уметь сравнивать ребра разных элементов, а значит нужно будет единственным образом их идентифицировать, поэтому класс ребра определен как класс с отношением эквивалентности (operator==() и operator!=()).

При решении вопроса, следует ли удалить элемент или оставить его в составе сетки необходимо знать веса всех его ребер, для чего необходимы списки инцидентности ребер элементам.

Процесс построения списка взвешенных ребер и списка инцидентности ребер элементам состоит из трех шагов:

1. При переборе всех элементов его ребра либо добавляются в список ребер с указанием инцидентности текущему элементу, либо, если они уже в нем есть, помечается, что данное ребро также принадлежит еще одному элементу. Фактически создается список инцидентности элементов ребрам.
2. Перебирается список ребер. Сохраняются лишь количество элементов, которым принадлежит ребро. Заполняются списки инцидентности ребер элементам.
3. Удаляется список инцидентности элементов ребрам.

При создании списков инцидентности элементов ребрам необходимо хранить небольшие массивы, содержащие номера элементов, инцидентных конкретному ребру. Поскольку в конце процесса от списка ребер остаются только веса (что означает удаление всех объектов Rib), то эту временную информацию естественно разместить в классе Rib – этой цели служит vector<unsigned int> BTcells. Также в класс добавлен метод AddCell(), с помощью которого отмечается инцидентность ребра очередному элементу.

В первом варианте программы временный список инцидентности элементов ребрам был реализован с помощью контейнера vector<Rib>. При тестировании было замечено, что построение этого временного списка происходит нелинейно по времени. Это

является следствием того, что в неупорядоченном массиве сложность поиска равна O(N), где N – количество ребер. При этом количество поисков пропорционально N. Таким образом, сложность первого шага (построения взвешенного списка ребер) составляет $O(N^2)$.

Чтобы избавиться от излишней сложности, было принято решение заменить контейнер для хранения ребер на set<Rib>. Контейнер set описывает автоматически упорядочивающийся массив, реализованный внутри как красно-черное дерево с постоянной сложностью добавления и сложностью поиска O(ln(N)) ([7, 8]). Чтобы использовать контейнер set, пришлось ввести в запись Rib лексикографическое отношение порядка (operator<()). Данный подход обеспечивает сложность построения временного списка инцидентности элементов ребрам O(Nln(N)), однако замедляет его удаление.

Хотя в данной работе не рассматривался вопрос ускорения выполнения третьего шага, связанного с удалением списка инцидентности элементов ребрам, представляется, что соответствующее время можно минимизировать с помощью ручного управления памятью (перегружая new и delete), то есть можно распределять память под хранение Rib в специально создаваемой программой куче и удалять эту кучу целиком.

3. Прореживание. Однопоточный и параллельный алгоритмы

Когда создан список взвешенных ребер и для каждого элемента известно, какие ребра ему принадлежат (имеется в виду вновь полученная единая нумерация ребер), сам процесс прореживания выполняется путем перебора всех элементов и проверки весов принадлежащих ему ребер – если все они больше единицы, то элемент помечается, как удаленный, а веса его ребер уменьшаются на единицу. Этот процесс многократно повторяется до тех пор, пока в очередной итерации не будет удалено ни одного элемента.

Для процедуры удаления элементов реализовано распараллеливание путем разбиения области на несколько частей и запуска в каждой из них отдельного потока, выполняющего прореживание. При этом возникает проблема совместного доступа к элементам, находящимся на границе двух областей, так как к элементам границы только один поток в течении одного кванта времени должен иметь доступ. Поэтому элементы границы областей записываются отдельно, так как только для них используется синхронизация доступа при прореживании. В элементах внутренних частей областей доступ не требует межпоточного согласования. Для синхронизации используются кроссплатформенные мьютексы библиотеки Qt.

Если используется больше двух потоков, возникает вопрос о количестве простаивающих потоков при совместном доступе к границам. Было решено использовать такую конфигурацию областей, при которой захват мьютекса будет означать, что освобождения этого мьютекса может ожидать максимум еще один поток - топологически такая конфигурация напоминает сферические слои. Любая граница находится только между двумя областями, к каждой из которых приписано по потоку. При этом разумно каждой области приписать две границы – внешюю и внутреннюю (с точки зрения сферических слоев), и к каждой из этих границ приписать по отдельному мьютексу. Так минимизируются издержки на согласование потоков. Общий вид области таков:

```
struct CellPart
{
 vector<unsigned int> cells;              // Внутренняя область
 vector<unsigned int> bound1;  // Внутренняя граница
 vector<unsigned int> bound2;  // Внешняя граница
};
```

Такая разбивка на подобласти легко достигается с помощью фронтообразного перебора области (рис. 1): выбирается произвольный элемент, затем перебираются все его соседи, после чего перебираются их внешние соседи, затем их внешние соседи и так далее до исчерпания всей области.

Организация подобной последовательности доступа производится с помощью структуры данных «очередь» и меток, сигнализирующих, что данный элемент уже был посещен в процессе перебора. Подобный перебор выглядит как расширяющийся сферический фронт, откуда и возникло его название.

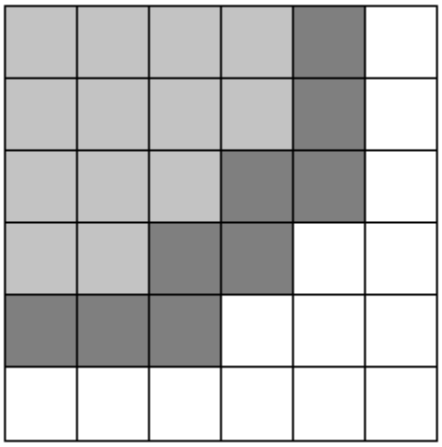


Рис. 1. Сбор элементов в подобласть

Сама разбивка производится следующим образом: во фронтообразном переборе в очередную подобласть вносится элементы. Как только сумма числа выбранных в подобласть и имеющихся в очереди элементов становится равной требуемому числу элементов в подобласти, мы помечаем уже выбранные элементы как внутренние для подобласти, а оставшиеся в очереди - как внешнюю границу. Граница выбирается исчерпанием очереди с одновременным наполнением новой очереди, состоящей из новых соседей-элементов. Эта новая очередь станет внутренней границей новой подобласти. Структура подобластей и границ схематично изображена на рис. 2.

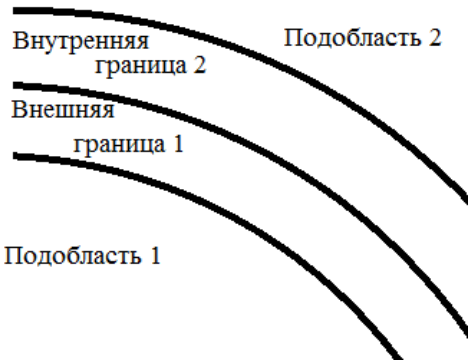


Рис. 2. Структура подобластей и границ

4. О тестировании

Программа тестировалась на нескольких сетках. При числе элементов сетки порядка десятков тысяч время работы программы составляет порядка минуты. Для сеток, содержащих порядка миллиона конечных элементов, время работы составляет десятки минут. Замечено, что самыми затратными по времени этапами были чтение/запись и процесс удаления списка инцидентности элементов ребрам. Этот этап занял около половины времени работы программы, что мотивирует на дальнейшее совершенствование кода.

Количество удаляемых элементов показывает, что ажурная схема способна уменьшать количество вычислений примерно втрое.

В то же время, тесты на параллельном алгоритме не оправдали использования параллелизма, потому что этап прореживания и без того оказался практически моментальным, а вследствие того, что его сложность по времени – линейная, очевидно, что он будет выполнятся быстро на любых сетках. С другой стороны, рассмотренный подход организации совместного доступа может быть использован для эффективного распараллеливания изначально медленных задач, например, расчетных программ МКЭ.

Литература

1. Чекмарев Д.Т. Ажурные схемы метода конечного элемента // Прикладные проблемы прочности и пластичности. Нижний Новгород: ННГУ, 1997. Вып. 55. С. 157-159.
2. Чекмарев Д.Т. Численные схемы метода конечного элемента на «ажурных» сетках // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2009. Вып. 2. С. 49-54.
3. Баженов В.Г., Чекмарев Д.Т. Об индексной коммутативности численного дифференцирования // Ж. вычисл. математики и мат. физики. 1989. Т. 29. № 5. С. 662-674.
4. Баженов В.Г., Чекмарев Д.Т. Вариационно-разностные схемы в нестационарных волновых задачах динамики пластин и оболочек. Н.Новгород: Изд-во Нижегород. ун-та, 1992. 159 с.
5. Баженов В.Г., Чекмарев Д.Т. Решение задач нестационарной динамики пластин и оболочек вариационно-разностным методом. Нижний Новгород: ННГУ, 2000. 118 с.
6. Чекмарев Д.Т., Кастальская К.А. О построении трехмерных ажурных сеток // Труды XII Межд. семинара «Супервычисления и математическое моделирование». Саров. 2011. С.374-381.
7. Джосьютис Н. C++ Стандартная библиотека. Для профессионалов. СПб.: Питер, 2004. – 730 с.
8. The C++ Resources Network. – http://www.cplusplus.com/.
9. Земсков Ю.В. Qt 4 на примерах. СПб.: БХВ-Петербург, 2008. – 608 с.

АДАПТАЦИЯ КОМПЛЕКСА ПРОГРАММ РАМЗЕС-КП ДЛЯ РЕШЕНИЯ ЗАДАЧ ГАЗОВОЙ ДИНАМИКИ И ТЕПЛОПРОВОДНОСТИ НА ГИБРИДНЫХ ПАРАЛЛЕЛЬНЫХ ЭВМ

А.А. Федоров, А.Н. Быков, Е.А. Сизов, В.Г. Куделькин, Н.Н. Жильникова, Д.Г. Гордеев

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Введение

В институте теоретической и математической физики (ИТМФ) РФЯЦ-ВНИИЭФ для решения научно-технических и технологических задач разработаны различные прикладные программы. Одна из них основана на эйлерово-лагранжевой методике РАМЗЕС-КП численного решения трехмерных нестационарных задач газовой динамики с учетом теплопроводности. Среди особенностей этой методики можно выделить наличие многокомпонентности (присутствие в одной ячейке нескольких веществ) и использование неявной разностной схемы при решении не только уравнения теплопроводности, но и системы уравнений газовой динамики.

В 2011 году начались работы по адаптации методики РАМЗЕС-КП для счета задач на гибридных ЭВМ с АрУ (арифметическими ускорителями). За прошедшее время были переписаны программы газовой динамики и теплопроводности для работы на нескольких ускорителях, реализована связь с библиотекой УРС-ОФ, которая предназначена для расчета уравнений состояний и пробегов фотонов в веществе, разработаны методы решения СЛАУ для работы на нескольких ускорителях.

В работе будут приведены доказательство работоспособности полученной программы, результаты численных экспериментов по исследованию эффективности распараллеливания и сравнение времени счета задачи на АрУ и времени счета задачи на ЦП (центральном процессоре).

1. Реализация программ на АрУ

В самом начале работы было проведено профилирование комплекса программ РАМЗЕС-КП, которое показало, что в коде нет «горячих пятен», то есть нельзя получить высокое ускорение за счет переписывания только некоторых программ.

Распараллеливание программ было проведено в основном поточечным методом, то есть каждую точку обсчитывает отдельная нить. Также следует отметить, что ЦП используется лишь для обмена сообщениями между процессами.

В ходе адаптации программ для гибридных параллельных ЭВМ было реализовано следующее:

1. Логическое связывание ускорителя с ядром ЦП, чтобы каждый АрУ работал только с одним ядром ЦП.
2. Обмен данными между ЦП и АрУ для программ, содержащих MPI-обмены.
3. Для реализации метода прогонки, использующегося для нахождения решения систем линейных алгебраических уравнений с трехдиагональной матрицей, которая возникает из-за использования неявной схемы, на нескольких АрУ, был реализован метод распараллеливания прогонки, предложенный Н.Н. Яненко.

4. Обработка списочных массивов, использующихся для хранения информации о смешанных ячейках (содержащих несколько веществ), была организована с помощью атомарных операций.

2. Оптимизация реализованных программ

После первых полученных характеристик эффективности распараллеливания началась работа по оптимизации кода. Существенную часть счёта на АрУ занимал обмен данными между ЦП и АрУ. Значительно сократить долю такого обмена нам позволили следующие действия:

1. Оптимизация объемов копирования информации между ЦП и АрУ.
2. Использование pinned-памяти для временных массивов, которые создаются на ЦП при работе с АрУ, в два раза сокращает время их копирования. Такие массивы возникают при выполнении операций на ЦП над данными, которые были рассчитаны на АрУ (основные массивы в комплексе РАМЗЕС-КП объявлены с атрибутом save, поэтому для них, к сожалению, нельзя использовать атрибут pinned).
3. Использование константной памяти для некоторых переменных, которые задаются при считывании файла расчета начальных данных задачи и не меняются в процессе расчёта.
4. Передача данных по значению в аргументах функций. Это параметры, которые используются только для чтения и управления внутри ядер, их можно явно не копировать.
5. Наиболее эффективным, но при этом и самым трудоёмким процессом, стала реализация асинхронного копирования при обмене данными между ЦП и АрУ там, где это было возможно.

Помимо вышеперечисленных действий для уменьшения времени работы на АрУ прогонок по первому направлению реализовано транспонирование исходной трехдиагональной матрицы и метод встречных прогонок, что позволило умень-шить время работы этого участка примерно в 2 раза.

3. Результаты тестирования

Программа тестировалась на задаче обжатия легкого вещества тяжелой оболочкой. Задача была посчитана как на ускорителях, так и на ядрах ЦП с числом точек 256*800 и числом временных шагов 1400. Для измерения эффективности задача считалась методом умножения последовательно на 1, 2, 4, 8, 16, 32 ускорителях. Ускорение на гибридной ЭВМ по сравнению с многопроцессорной ЭВМ составило от 10,74 на 32-х ускорителях до 15,57 на одном ускорителе, эффективность распараллеливания составила не ниже 59%.

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. Новаев Д.А., Бартенев Ю.Г., Липов Д.И. и др. Программные средства STK для исследования эффективности выполнения параллельных приложений // ВАНТ. Сер. Математическое моделирование физических процессов. 2011. Вып.4. С. 72-81.

4. Федоров А.А., Быков А.Н., Сизов Е.А. Метод распараллеливания прогонки на гибридных ЭВМ // Вычислительные методы и программирование. 2013. Раздел 2. С. 43-47.

РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ В ЗАДАЧАХ ВОГНУТОГО ПРОГРАММИРОВАНИЯ

О.В. Хамисов

Институт систем энергетики им. Л.А. Мелентьева СО РАН, Иркутск

Рассматривается задача минимизации (квази)вогнутой функции на ограниченном многогранном множестве. Хорошее локальное решение можно получить, многократно решая однотипные задачи линейного программирования. Для этого предлагается использовать систему GAMS со встроенными в ней возможностями распараллеливания. Глобальное решение ищется посредством методов внешней и внутренней аппроксимации и тестируется на языке Python также с распараллеливанием.

Введение

Заданы (квази)вогнутая функция $f: \mathbb{R}^n \to \mathbb{R}$, матрица A размера $m \times n$, вектор $b \in \mathbb{R}^m$. Требуется решить задачу вогнутого программирования

$$f(x) \to min, x \in X = \{x: Ax \leq b\}.$$

Хорошо известно, что глобальный минимум в такой задаче достигается в вершине X. Этот факт систематически используется в предлагаемой методике.

Локальный поиск

Для нахождения хорошего локального решения применяется следующий широко распространённый подход. Сначала находятся решения $2n$ задач линейного программирования:

$$v^k = argmin\{< c^k, x >: x \in X\}, k = 1, \dots, 2n,$$

где $c^k = (0, \dots, 1, \dots 0)^T, k = 1, \dots, n$, $c^k = (0, \dots, -1, \dots 0)^T, k = n+1, \dots, 2n$. Данный процесс хорошо распараллеливается в GAMS, существенно (в разы) сокращая время решения. Затем из всех найденных вершин выбирается вершина с минимальным значением целевой функции. Далее определяются вершины, соседние с выбранной. Если хотя бы в одной из соседних значение целевой функции меньше, то переходим в эту вершину, снова находим соседние и т.д.

Глобальный поиск

Для гарантированного нахождения глобального минимума используется процедура внешней аппроксимации [1], реализованная на языке Python. Данная процедура носит переборный характер, и именно переборный алгоритм был распараллелен. Тестирование показало, что на 4-х ядерном ноутбуке Intel Core i7/8GB/2.3GHz можно достичь почти 3-х кратного ускорения.

Вычислительный эксперимент

Предварительный вычислительный эксперимент был проведён на задачах от 5 до 15 переменных для процедуры внешней аппроксимации, и от 10 до 100 переменных для процедуры локального поиска. В обоих случаях удалось добиться хорошего ускорения вычислений.

Литература

1. Tuy H. Convex analysis and global optimization. Dordrecht: Kluwer Academic Publishers, 1998. – 339 p.

ПРИМЕНЕНИЕ МЕТОДА НЬЮТОНА В СИММЕТРИЧНОЙ ПРОБЛЕМЕ СОБСТВЕННЫХ ЗНАЧЕНИЙ

О.О. Хамисов

Иркутский госуниверситет

Существует множество различных проблем, таких как устойчивость системы линейных уравнений или решение нелинейной задачи минимизации, для решения которых необходимо найти собственные векторы и собственные значения матриц. Данная работа описывает метод, позволяющий решить эту задачу. В ней дано математическое описание методов для решения частной и полной проблем собственных значений и представлены результаты предварительного численного эксперимента. В силу того, что данная проблема требует большого объема вычислений, рассматривается возможность распараллеливания представленного в работе метода.

Введение

Рассматриваемая задача имеет следующую постановку. Дана симметричная матрица $A \in \mathbb{R}^{n\times n}$. Необходимо вычислить собственные векторы и собственные значения этой матрицы. Данная задача имеет несколько способов численного решения таких, как трехдиагональная QR-итерация, RQ-итерация, метод «Разделяй и властвуй» и другие [1]. В этой работе представлен метод, основанный на том, что собственный вектор является стационарной точкой отношения Релея.

Основные формулировки

Пусть $A \in \mathbb{R}^{n\times n}$ – симметричная матрица. Функция

$$\rho(x) = \frac{\langle x, Ax\rangle}{\langle x, x\rangle}$$

называется отношением Релея. Если x – собственный вектор матрицы A, то $\rho(x)$ – соответствующее ему собственное значение. При этом градиент отношения Релея равен нулю

$$\nabla\rho(x) = 0. \qquad (1)$$

Поэтому частичную проблему собственных значений можно решить, найдя при помощи метода Ньютона решение уравнения (1).

Метод 1 (метод Ньютона).

Входные данные: $F\colon \mathbb{R}^n \to \mathbb{R}^n, x_0$ – начальное приближение.

Общий шаг метода:

$$x^{k+1} = x^k - \left(F'(x)\right)^{-1} F(x^k).$$

Теорема 1 [2]. Пусть отображение $F\colon \mathbb{R}^n \to \mathbb{R}^n$ непрерывно дифференцируемо в некоторой окрестности решения $x \in \mathbb{R}^n$, тогда для любого начального приближения $x^0 \in \mathbb{R}^n$ верно $\lim_{k\to\infty} x_k = x$. Если производная F непрерывна по Липшицу в окрестности точки x, то скорость сходимости квадратичная.

Для уравнения (1) метод Ньютона может быть записан в следующей форме.

Метод 2.

Входные данные: $A \in \mathbb{R}^{n\times n}$ – симметричная матрица, x_0 – начальное приближение.

Общий шаг метода ($k \geq 0$):

вектор y^k находится как решение системы линейных уравнений

$$\nabla^2\rho(x^k)y^k = \nabla\rho(x^k),$$

$(k\ +\ 1)$-ое приближение находится по формуле

$$x^k + 1 =\ x^k -\ y^k.$$

Здесь $\nabla^2\rho(x)$ – гессиан отношения Релея,

$$\nabla^2\rho(x) = \frac{1}{\langle x,x\rangle}\left[2A - \frac{4Axx^T + 4xx^TA^T + 2I\langle x,Ax\rangle}{\langle x,x\rangle} + \frac{8xx^T\langle x,Ax\rangle^2}{\langle x,x\rangle}\right],$$

где $I \in \mathbb{R}^{n\times n}$ – квадратная единичная матрица.

Неприменимость метода Ньютона

Пусть x – собственный вектор, λ – собственное число симметричной матрицы $A \in \mathbb{R}^{n\times n}$, тогда верна следующая цепочка равенств:

$$\nabla^2\rho(x) = \frac{1}{\langle x,x\rangle}\left[2A - \frac{4Axx^T + 4xx^TA^T + 2I\langle x,Ax\rangle}{\langle x,x\rangle} + \frac{8xx^T\langle x,Ax\rangle^2}{\langle x,x\rangle}\right] =$$

$$= \frac{1}{\langle x,x\rangle}\left[2A - \frac{4Axx^T + 4xx^TA^T}{\langle x,x\rangle} - 2I\lambda + \frac{8xx^T\langle x,Ax\rangle^2}{\langle x,x\rangle}\right] =$$

$$= \frac{2}{\langle x,x\rangle}[A - I\lambda].$$

$$\det\nabla^2\rho(x) = \frac{2}{\langle x,x\rangle}\det[A - I\lambda] = 0.$$

Таким образом, гессиан отношения Релея не удовлетворяет условиям теоремы 1 о сходимости, и метод 2 расходится на некоторых матрицах.

Пример 1. Для невырожденной матрицы

$$B = \begin{pmatrix}3 & 2\\ 2 & 0\end{pmatrix}$$

и начальной аппроксимации $x^0 = (0,1)^T$ мы имеем

$$x^k = \begin{pmatrix}0\\ 2^k\end{pmatrix}.$$

Модификация метода Ньютона 1

Для того чтобы избавиться от вырожденности в гессиане отношения Релея от собственного вектора введем следующее отображение:

$$\Phi(x,\epsilon) =\ \nabla^2\rho(x) + \frac{2\epsilon}{\langle x,x\rangle}I.$$

Теорема 2. Пусть $A \in \mathbb{R}^{n\times n}$– симметричная матрица, $\overline{x}$ – ее собственный вектор. Тогда для любого ϵ такого, что $\rho(x) - \epsilon$ не является собственным числом, определитель $\det\Phi(x,\epsilon) \neq 0$.

Метод 3.

Входные данные: $A \in \mathbb{R}^{n\times n}$ – симметричная матрица, $\epsilon > 0$ – число, удовлетворяющее условию теоремы 2, x_0 – начальное приближение такое, что $||x_0|| = 1$.

Общий шаг метода ($k \geq 0$):

$$z^{k+1} =\ x^k - \left(\Phi(x^k,\epsilon)\right)^{-1}\nabla\rho(x^k),$$

$$x^{k+1} = \frac{z^{k+1}}{||z^{k+1}||}.$$

Условие остановки:

$$||\nabla\rho(x^k)|| \leq \epsilon.$$

Теорема 3. Пусть $A \in \mathbb{R}^{n\times n}$, $\epsilon > 0$ – число, удовлетворяющее условию теоремы 2, тогда

$$||x^{k+1} - x^k|| \leq D||x^k - \overline{x}||^2$$

для любого k, удовлетворяющего условию

$$||\nabla\rho(x^k)|| \geq \epsilon,$$

где $D = const, \overline{x}$ – собственный вектор.

Модификация метода Ньютона 2

Отрицательной стороной метода 3 является тот факт, что при малых ϵ отображение $\Phi(x, \epsilon)$ является близким к вырожденному. Поэтому введем новое отображение

$$\Psi(x) = A - \rho(x)I + cxx^T,$$

где $c \neq 0$ – константа. Заменив отображение $\Phi(x^k, \epsilon)$ на $\Psi(x^k)$ в методе, получим метод 4.

Метод 4.

Входные данные: $A \in \mathbb{R}^{n\times n}$– симметричная матрица, $\epsilon > 0$ – точность вычислений, x_0 – начальное приближение такое, что $||x_0|| = 1$.

Общий шаг метода ($k \geq 0$):

вектор y^k находится как решение системы линейных уравнений

$$\Psi(x^k)y^k = \nabla\rho(x^k),$$
$$z^{k+1} = x^k - y^k,$$
$$x^{k+1} = \frac{z^{k+1}}{||z^{k+1}||}.$$

Условие остановки:

$$||\nabla\rho(x^k)|| \leq \epsilon.$$

Теорема 4. Метод 4 сходится при любом начальном приближении x_0 таком, что $||x_0|| = 1$ к собственному вектору $\overline{x}$ при условии, что соответствующее ему собственное число имеет кратность 1.

Применение метода Ньютона для решения полной симметричной проблемы собственных значений

Пользуясь тем фактом, что собственные вектора симметричной матрицы ортогональны, можно исключить уже вычисленные вектора из задачи. Для этого рассмотрим следующую задачу:

$$\rho(x) \to min,$$
$$\langle \tilde{x}^i, x\rangle = 0, i = \overline{1, m}, m < n,$$

где $\tilde{x}^i$ – уже вычисленные собственные векторы. Для решения этой задачи достаточно решить уравнение

$$L'(x, v) = 0, \quad (3)$$

где

$$L\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, L(x, v) = \rho(x) + \sum_{i=1}^{m} v_i \langle \tilde{x}^i, x\rangle$$

– функция Лагранжа. Для решения (3) используется метод Ньютона.

Метод 5.

Входные данные: $A \in \mathbb{R}^{n\times n}$– симметричная матрица, не имеющая кратных собственных чисел, $\epsilon > 0$ – точность вычислений, x_0 – начальное приближение такое, что $||x_0|| = 1, \tilde{x}^i, i = \overline{1, m}$ – уже вычисленные собственные векторы.

Общий шаг метода ($k \geq 0$):

x^{k+1} является решением уравнения

$$\begin{pmatrix} \Phi(x^k) & \left(\omega'(x^k)\right)^T \\ \omega'(x^k) & 0 \end{pmatrix}\begin{pmatrix} x^{k+1}-x^k \\ \nu^k \end{pmatrix} = \begin{pmatrix} \nabla\rho(x^k) \\ \omega(x^k) \end{pmatrix},$$

где

$$\omega(x) = (\langle\tilde{x}^1, x\rangle, \dots, \langle\tilde{x}^m, x\rangle)^T.$$

Условие остановки:

$$||\nabla\rho(x^k)|| \le \epsilon.$$

Используя метод 4 для вычисления первого собственного вектора и метод 5 для последовательного вычисления следующих векторов, можно найти все собственные числа и вектора симметричной матрицы

Метод 6.

Входные данные: $A \in \mathbb{R}^{n\times n}$ – симметричная матрица, не имеющая кратных собственных чисел, $\epsilon > 0$ – точность вычислений.

Выходные данные: $(\tilde{x}^i, \lambda_i), i = \overline{1,n}$ – пары собственных векторов и соответствующих им собственных значений.

Первой собственный вектор вычисляется методом 4 со стартовым приближением $x^0 = (1,0,\dots,0)^T$, далее методом 5 вычисляются остальные собственные вектора. Соответствующие им собственные числа вычисляются по формуле

$$\lambda_i = \rho(\tilde{x}^i), i = \overline{1,n}.$$

Программная реализация

В силу того, что в данном методе выполнение итераций осуществляется последовательно, целесообразно проводить распараллеливание вычислений, реализуемых в ходе выполнения итерации. Исследование методов 5 и 6 показывают, что основная часть машинного времени тратится на решение системы линейных уравнений. В данной работе решение систем линейных уравнений проводится двумя способами: методом Гаусса и при помощи QR-разложения методом Гивенса [3]. Распараллеливание метода Гаусса происходит по процедуре, указанной в [4]. В силу того, что матрицы вращений Гивенса имеют структурированную разреженную форму, распараллеливание произведения матриц в QR-разложении имеет специальную упрощенную форму. Обратный ход решения системы линейных уравнений при помощи QR-разложения проводится также как в методе Гаусса.

Реализация метода была написана на языке Python 2.7. Предварительный эксперимент проводился на ноутбуке Acer Aspire 5920g на базе двухъядерного процессора Intel Core 2 Duo T8300, 2.4 ГГц, 3Гб RAM под управлением операционной системы Linux Mint 11. В таблице ниже приведены результаты вычислений.

Таблица 1. Результаты численного эксперимента

Размер матрицы	Среднее количество итераций для метода 4 (для одного собственного вектора)	Среднее количество итераций для метода 4 (для всех собственных векторов)
30	6	217
50	7	376
100	7	767

При использовании параллельного варианта решения систем линейных уравнений скорость вычислений возросла на 28% при использовании QR-разложения и на 36% при использовании метода Гаусса.

Заключение

В силу того, что приведенный метод выводит собственные вектора и числа начиная с первых итераций, он может обеспечить ускорение в решении задач, для решения которых необходимы один или несколько собственных векторов.

Литература

1. Деммель Д.Ж. Вычислительная линейная алгебра. Москва: Мир. 2001.
2. Измайлов А.Ф., Солодов М.В. Численные методы оптимизации. Москва. Физматлит. 2003.
3. Hogben L. Handbook of Linear Algebra. Chapman & Hall/CRC. 2007.
4. Гергель В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. Нижний Новгород: Издательство Нижегородского госуниверситета. 2010.

ПРОБЛЕМА ПРОГРАММНОЙ РЕАЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ РАСЧЕТА СЛОЖНЫХ СТРОИТЕЛЬНЫХ ОБЪЕКТОВ В РАСПРЕДЕЛЕННЫХ КОМПЬЮТЕРНЫХ СРЕДАХ

В.Е. Хромых, А.Н. Супрун, Д.И. Кислицын

Нижегородский государственный архитектурно-строительный университет

В настоящее время развитие строительной отрасли в России и за рубежом характеризуется тенденциями к нарастанию сложности проектируемых объектов, повышению требований к надежности и устойчивости конструкций к прогрессирующему обрушению, сокращению сроков проектирования, повышению технико-экономических показателей строительных объектов. В связи с этим непрерывно повышаются требования и к системам автоматизированного проектирования, используемым в строительной отрасли. Для этой цели разрабатываются интегрированные системы, ориентированные на автоматизацию нескольких этапов подготовки проекта, создаются программные средства поддержки технологии корпоративного проектирования, совершенствуется дружественный интерфейс программных комплексов, применяются параллельные технологии выполнения расчетов.

В целях повышения производительности системы автоматизированного проектирования строительных объектов было предложено [1-4] условное разделение сооружения на проектные единицы, совместная работа которых как единой механической системы обеспечивается специальной программой «Решатель». Под проектной единицей в рассматриваемом здесь варианте автоматизированных систем с распараллеливанием вычислительных процессов понимается часть сооружения, которая:

- может быть выделена путем «разрезания» проектируемого объекта по стержням;
- должна быть геометрически неизменяемой системой, имеющей опорные связи с основанием.

Указанный метод, реализованный для конструкций из стержневых систем, может быть развит и на сплошные среды, а также на широкий класс краевых задач математической физики. Решение задачи может проводиться на кластере из N рабочих стан-ций, каждая из которых выполняет моделирование работы одной проектной единицы поэтапно в соответствии с алгоритмом, описанным в [1, 2].

Опыт построения автоматизированной системы управления расчётом сложных строительных объектов на базе метода разделения объекта на проектные единицы [2-4] показал, что для реализации системы необходимо решить следующие задачи:

1) разработка базы данных для централизованного хранения всех необходимых в процессе расчёта данных;
2) разработка удобного, понятного и информативного графического интерфейса для пользователя;
3) разграничение прав доступа для пользователей при работе с системой;
4) автоматизация процессов расчета проектной единицы, исключающая потребность в постоянном контакте системы с пользователем;
5) организация клиент-серверного взаимодействия;

6) формирование исходных данных на входном языке базового программного средства;
7) управление базовым программным средством;
8) формирование и решение систем уравнений;
9) организация сбора и анализа результатов расчёта;
10) графическое отображение структуры и параметров модели.

В настоящее время в ННГАСУ продолжается разработка программного комплекса АСУР «Решатель», позволяющая в соответствии с методом разделения объекта на проектные единицы (ПЕ) сделать доступным решения задач неограниченной сложности. Текущая версия позволяет многократно увеличить количество вычислительных узлов (по сравнению с предыдущей версией), что в свою очередь позволяет значительно повысить производительность распределенной вычислительной системы.

По сравнению с предыдущей реализацией АСУР «Решатель» к настоящему времени были внесены следующие изменения:

- при разделении на ПЕ проектные единицы могут связываться неограниченным числом стержневых связей;
- увеличена стабильность работы как сервера, так и системы в целом;
- расширен функционал для работы с системой проектировщика;
- проведена работа по повышению автоматизации системы;
- внедрена возможность использования в качестве базового программного средства новейшей версии коммерческого программного продукта «Лира»;
- расширена БД;
- доработан и интегрирован в систему модуль «Мнемосхема», графически отображающий структуру расположения и взаимосвязь ПЕ.

В качестве СУБД в разработанном программном комплексе используется свободно распространяемая версия Microsoft SQL Server 2008 Express.

Система АСУР «Решатель» построена на основе клиент-серверного взаимодействия при трехуровневой архитектуре в виде совокупности следующих компонент: сервера базы данных, клиентского приложения и сервера приложений, отвечающего за выполнение логики приложения.

Графический интерфейс системы позволяет работать в двух режимах: «Руководитель проекта» (ГИП) и «Проектировщик». Выбор режима происходит после прохождения авторизации автоматически. При успешной авторизации пользователь переходит к главному диалоговому окну системы (рис. 1).

На главном диалоговом окне вводится вся необходимая информация о проекте. А также реализуется доступ к дополнительным диалоговым окнам для ввода необходимой информации, управления расчетом, получения отчета и т.д.

В режиме «Руководитель проекта» ГИП должен иметь возможность ввода:

- данных об объекте (наименование объекта, шифр проекта);
- количества проектных единиц (ПЕ);
- фамилий проектировщиков, которые будут рассчитывать «свои» ПЕ;
- ссылок на чертежи (рисунки) и иную проектную документацию для проектировщиков по каждой ПЕ.

После ввода всех необходимых данных ГИП должен будет посредством кнопки «Передать на Сервер» отправить данные на сервер, которые будут сохранены в базу данных, после чего проект становится доступен для работы другим проектировщикам. Каждый проектировщик входит в систему под своей учетной записью и в соответствии с данными БД получает доступ только к «своему» проекту и «своей» ПЕ.

По нажатию кнопки «Запрос данных» на главное диалоговое окно подгружается введенная ГИПом информация по проекту.

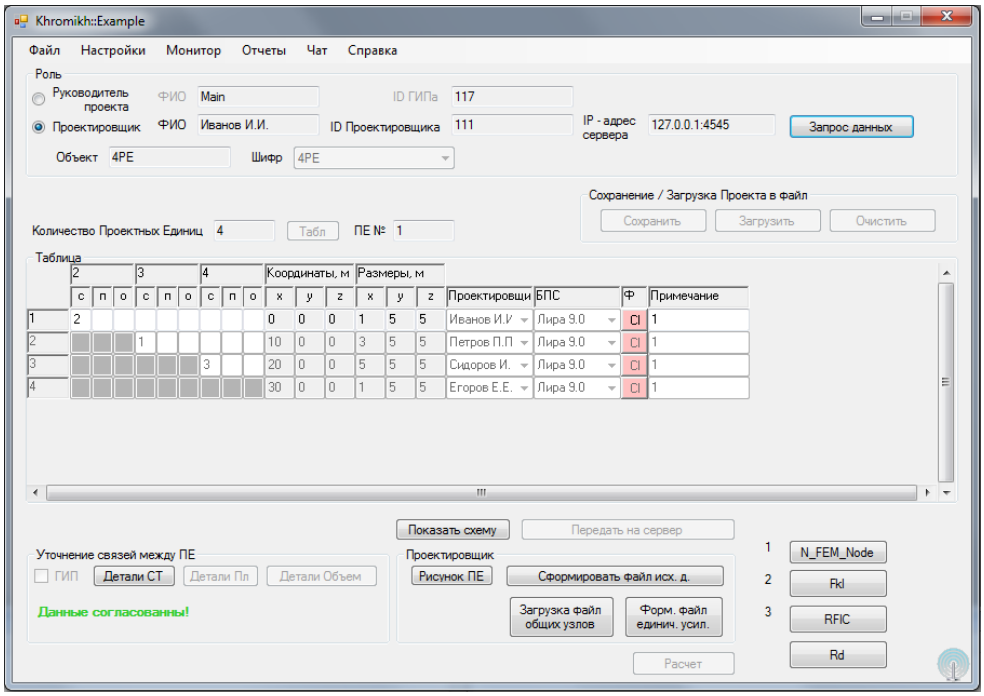


Рис. 1. Главное диалоговое окно системы «Решатель»

В режиме «Проектировщик» графический интерфейс системы расширился диалоговым окном, которое отображает ход выполнения проекта. По ходу работы с проектом в БД будут поступать данные от клиента о пройденных этапах, которые отображаются в соответствующем окне (рис. 2).

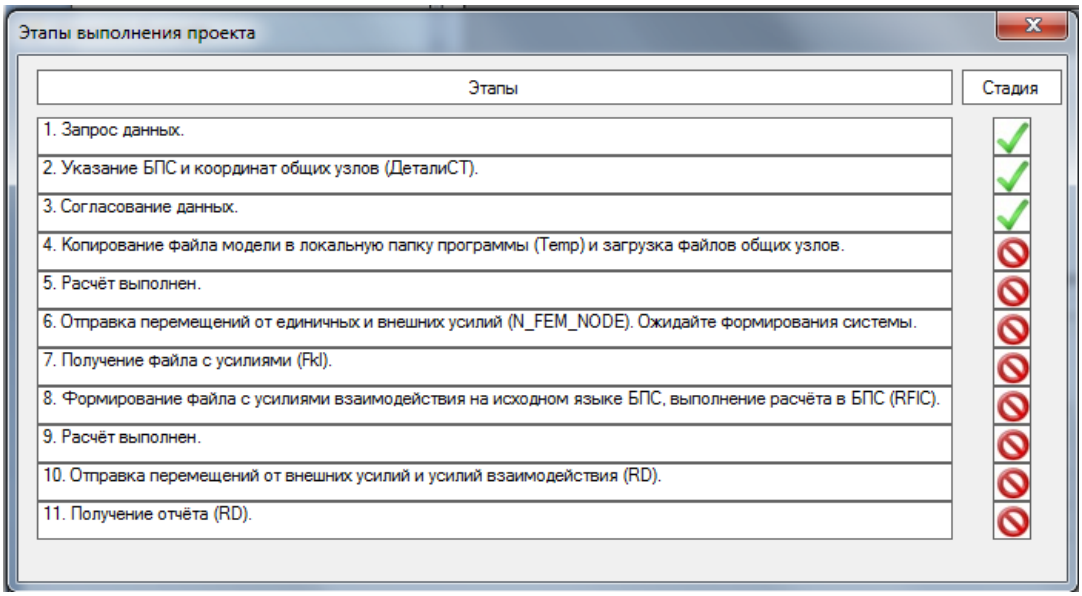


Рис. 2. Диалоговое окно «Этапы выполнения проекта»

Проектировщик в данной реализации системы АСУР «Решатель» уже может воспользоваться мнемосхемой – условной информационной моделью расположения ПЕ, выполненной как комплекс символов, изображающих элементы системы с их номерами и взаимными связями (рис. 3). Модуль запускается через главное диалоговое окно системы кнопкой «Показать схему».

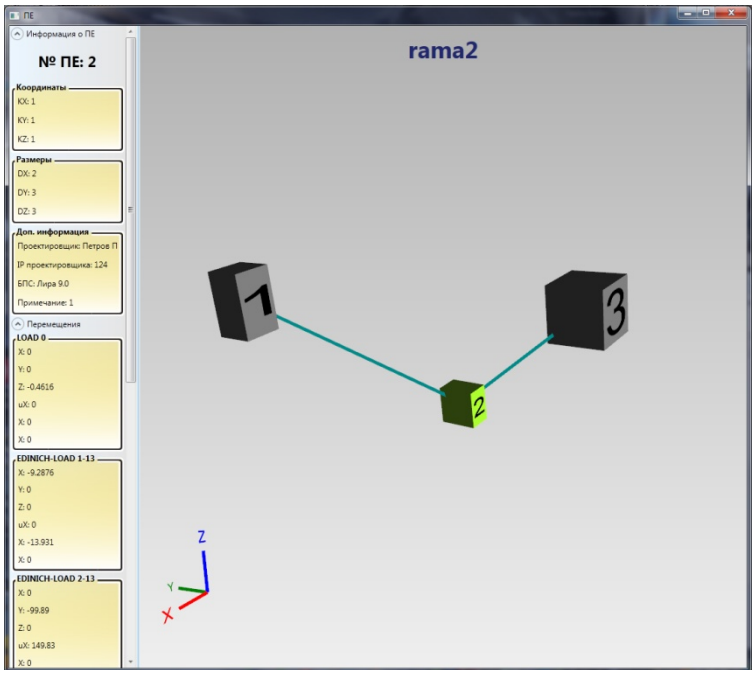


Рис. 3. Модуль «Мнемосхема»

В режиме «Проектировщик» в отдельном диалоговом окне «Детали Стержни» (рис. 4), открывающемся кнопкой «Детали СТ», расположенной на главном диалоговом окне, указывается базовое программное средство (БПС) для «своей» ПЕ и координаты общих узлов.

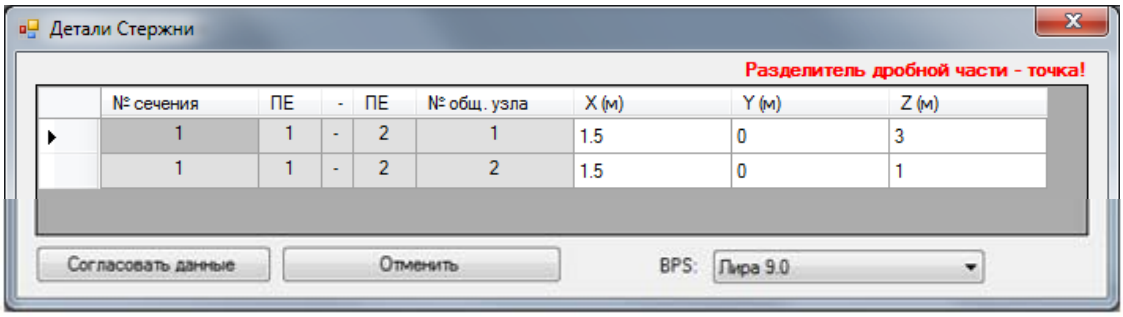


Рис. 4. Диалоговое окно «Детали Стержни»

После ввода данных, уточняющих взаимосвязи между ПЕ, необходимо согласовать эти данные кнопкой «Согласовать данные». Данные будут отправлены на сервер. После успешного согласования координат общих узлов на локальных компьютерах проектировщиков будет сформирован файл со списком общих узлов и координат на исходном языке БПС. Проектировщики на основе данного файла должны будут построить в БПС модели «своих» ПЕ, а затем кнопкой «Загрузка файлов общих узлов» через соответствующее диалоговое окно (рис. 5) загрузить сформированные проектировщиками в БПС файлы со списками номеров и координат общих узлов в системе БПС для передачи в БД на сервер.

После передачи файлов общих узлов на всех локальных компьютерах проектировщиков будет запущен расчет объекта по методу разделения на ПЕ от единичных загружений. После успешного расчета «своей» ПЕ от единичных загружений система автоматически отправляет перемещения на сервер и запрашивает посредством клиента файл с усилиями взаимодействия, используя который на локальном компьютере проектировщика будет сформирован файл с усилиями взаимодействия для «своей» ПЕ на исходном языке БПС и запущен на расчет от усилий взаимодействия. После успешного

расчета «своей» ПЕ от усилий взаимодействия перемещения будут автоматически отправлены системой посредством клиента на сервер. Как только перемещения от всех ПЕ из расчета от усилий взаимодействия будут отправлены, проектировщики в автоматическом режиме получат детальный отчет о проведенном проекте в специальном диалоговом окне (рис. 6). В предыдущей реализации системы АСУР «Решатель» пользователю необходимо было контролировать выполнение выше указанных этапов расчета и самому выполнять определенные действия для того, чтобы перейти к следующему этапу.

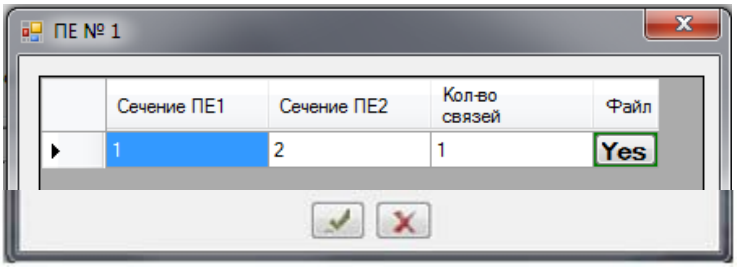


Рис. 5. Диалоговое окно «Загрузка файлов общих узлов»

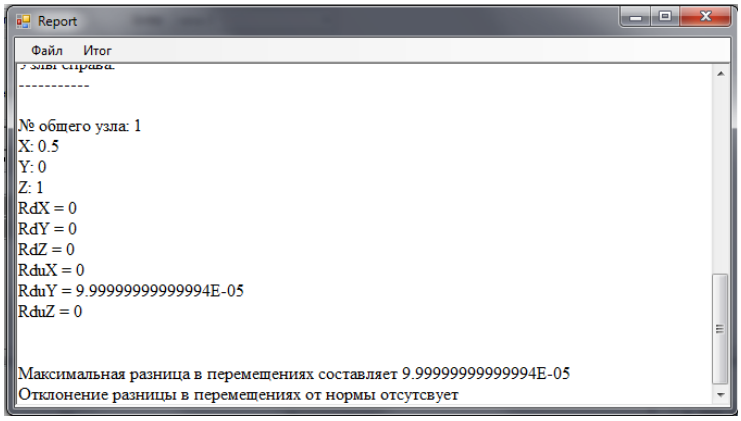


Рис. 6. Диалоговое окно «Отчет»

На основе имеющейся сейчас системе АСУР «Решатель» планируется проводить эксперименты с использованием БПС «Лира 9.4» с последующим анализом полученных данных и сравнением результатов при решении тех же задач, но уже с использованием предыдущей версии БПС – «Лира 9.0».

На указанном уровне развития интерфейса завершается очередной этап модернизации системы «Решатель».

Литература

1. Кислицын Д.И., Супрун А.Н. Программный модуль для расширения функциональных возможностей вычислительного комплекса «Лира» // International Journal for Computational Civil and Structural Engineering. Vol. 4, Issue 2. Moscow–New York. 2008.
2. 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. Новочеркасск: Юж.-Рос. гос. техн. ун-т. (НПИ), 2010.
3. Кислицын Д. И., Хромых В. Е. Численная реализация метода разделения конструкции на проектные единицы // Материалы 23-й Всероссийской научно-практической

конференции по графическим информационным технологиям и системам «КОГРАФ-2013» – Н. Новгород, 2013.
4. 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.

РЕАЛИЗАЦИЯ ЭЛЛИПТИЧЕСКОЙ КРИПТОСИСТЕМЫ С ПАРАЛЛЕЛИЗМОМ МАШИННЫХ ОПЕРАЦИЙ

Н.И. Червяков, М.Г. Бабенко, А.В. Гладков, М.А. Дерябин

Северо-Кавказский федеральный университет, Ставрополь

Рассматриваются проблемы реализации криптографических систем на массивно-параллельных системах. Проводится обзор реализации криптосистем на элиптических кривых. Рассматривается возможность применения системы остаточных классов для повышения эффективности реализации криптографических преобразований.

Введение. Постановка задачи

В современном обществе много внимания уделяется обеспечению безопасности информации. Одним из перспективных способов построения криптосистем с открытым ключом являются криптосистемы, построенные на точках эллиптической кривой. В настоящее время ряд государств, включая Российскую Федерацию, США, Украину и т.д. принимают стандарты по построению цифровых подписей с использованием эллиптических кривых. Также разработаны стандарты шифрования данных на эллиптической кривой IEEE 363 PI (стандартные спецификации для шифрования с открытым ключом), они включают в себя шифрование, цифровую подпись и криптографический ключ механизма согласования с использованием криптосистем на эллиптической кривой (ECC). Вследствие всего вышесказонного становится актуальной задача эффективной реализации арифметики с точками эллиптической кривой с параллелизмом машинных операций.

Основная часть

Модульная арифметика играет центральную роль в реализации ECC. Модульное сложение и умножение по модулю *n* работают как обычные сложение и умножение, за исключением того, что ответ на расчет сводится к его остатку от деления на p. Например, 2 • 4 = 3 (mod 5), потому что 8 имеет остаток 3, когда делится на 5.

Определение 1. Группа точек эллиптической кривой – это точка в бесконечности O и множество решений (x, y) уравнения вида

$$y^2 = x^3 + ax + b(\mathrm{mod}\, p),$$

где a, b – любые два числа из F_p, а p – простое число. Если решение (x, y) удовлетворяет указанному выше уравнению, то (x, y) является точкой на эллиптической кривой.

В аддитивной группе точек эллиптической кривой определена операция сложения $P_1, P_2, P_3 \in EC(F_p)$, где $P_3 = P_1 + P_2$ и координаты точки $P = (x_3, y_3)$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},$$

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1.$$

Множество точек на эллиптической кривой образует группу, которая используется в построении эллиптической кривой криптосистемы. ECC могут быть использованы для шифрования и цифровой подписи.

Криптографические ключи генерируются следующим образом

- Выбирается эллиптическая кривая, определенная уравнением $y^2 = x^3 + ax + b \pmod p$. Простое число p должно быть не менее 160 бит.
- Определяется точка P на эллиптической кривой.
- Выбирается целое x в интервале $(1, p)$.
- Вычисляется $Q = xP$.
- Открытый криптографический ключ, являющийся комбинацией эллиптической кривой, выбранной на ней точкой P, простого числа p и Q.
- Закрытый ключ *x*.

Безопасность ECC основана на проблеме решения задачи дискретного логарифмирования эллиптической кривой: возможность определять x при заданных P и Q. Эффективность этой криптосистемы зависит от быстроты вычисления kP для получения некоторого числа x и точки P на эллиптической кривой.

Для создания цифровой подписи сообщения, отправитель должен выполнить следующие действия:

- выбрать целое k в интервале $(1, p-1)$.
- вычислить kP.
- вычислить $k^{-1} \bmod p$.
- вычислить $s = k^{-1}(hex(M) + xr) \bmod p$,

где $hex(M)$ – алгоритм хеширования текста сообщения M, x – закрытый ключ отправителя $r = x_1 \bmod p$.

Цифровой подписью для сообщения m является пара целых чисел (*r*, *s*).

Цифровая подпись отправителя может быть проверена с помощью получателя следующим образом:

- Получить подлинную копию открытого ключа отправителя.
- Убедиться, что r и s – целые числа из интервала $(1, p)$.
- Вычислить $w = s^{-1} \bmod p$.
- Вычислить $hex(M)$.
- Вычислить $u_1 = hex(M)w \bmod p$,

где $hex(M)$ – SHA, вычисляемый по сообщению M.

- Вычислить u_2 = rw mod n.
- Вычислить $u_1 P + u_2 Q = (x_0, y_0) \bmod p$.
- Цифровая подпись отправителя подтверждается, если $r_1 = x_0 \bmod p$.

Трудность решения задачи дискретного логарифмирования эллиптической кривой позволяет достичь более высокого уровня безопасности, при этом имея пропорционально меньший размер ключа шифрования. В свою очередь, чем меньше размер ключа, тем больше вычислительная эффективность. Хотя ECC могут быть реализованы в любом применении криптографии с открытым ключом (например, банкоматы, карточки с балансом телефона, медицинские хранения записей или электронные деньги), использование ECC особенно выгодно в тех случаях, когда возможности хранения, обработки, пропускная способность ограничены. Примерами таких приложений являются беспроводная связь и смарт-карты.

Устройства беспроводной связи и смарт-карты имеют ограниченные возможности хранения, обработки и пропускной способности. Короткий размер ключа ECC снижает как необходимый объем дискового пространства, так и объем данных, которые должны быть переданы. Это также увеличивает эффективность операций с открытым ключом и скорость вычислений. Кроме того, так как эти приложения требуют более высокий уровень безопасности (с более длинными ключами), ECC может продолжать обеспечивать уровень безопасности, эквивалентный другим криптосистемам с открытым ключом, с помощью соразмерно коротких ключей и использования меньшего количества дополнительных системных ресурсов.

В десятичной арифметике логарифмы часто используются для умножения и деления. В СОК используется аналогичный метод, называемый индексными вычислениями [1]. Использование индексного преобразования над полем Галуа $GF(p)$ сводит операции умножения и деления к операциям сложения и вычитания соответственно. Операция умножения является модульной и поэтому может быть реализована в СОК как сложение. С точки зрения аппаратной реализации, сложение в СОК реализуется легче, чем умножение [1, 2]. Деление, напротив, является немодульной операцией в СОК, поэтому использование индексных преобразований над $GF(p)$ значительно улучшит время выполнения операции и снизит аппаратурные затраты.

При построении умножителей, работающих в модулярной арифметике, можно воспользоваться индексным или «дискретно-логарифмическим» представлением операндов и заменить операцию модулярного умножения операцией модулярного сложения индексов или дискретных логарифмов. Индексное представление модулярного числа основывается на понятии первообразного «корня по простому модулю». Таким корнем является целое число, возведение которого в степень 1,2,…, p–1 дает неповторяющиеся вычеты по модулю [1].

Индексный метод позволяет получить очень простую реализацию с помощью метода, аналогичного методу вычисления логарифмов. Умножители по модулю p_i реализуются с помощью табличного поиска при малом p_i (5 бит и менее) и индексного сложения при больших p_i (6 бит и более) [3]. Поэтому в тех случаях, когда диапазон обрабатываемых данных лежит в этих пределах, целесообразно использовать умножители и делители на основе индексного исчисления. Однако индексное счисление может быть использовано только в том случае, когда модуль p_i – простое число.

На практике встречаются также случаи, когда модуль p_i - произвольное число, поэтому появляется необходимость разработать эффективные методы умножения, пригодные при любом модуле, независимо от того, является ли последний простым числом или нет. Так, например, если реализовать умножитель по модулю $p_i = 249$, который не является простым числом, так как $249 = 83 \cdot 3$, то непосредственное использование индексного счисления невозможно. Так как числа 83 и 3 являются простыми, то возможно индексное счисление по разложенным модулям.

При умножении (делении) по модулю p_i сначала определяются индексы чисел, затем эти индексы складываются (вычитаются) по модулю p_i –1 и по обратной таблице определяется окончательный результат [4].

Если модуль не является простым, то необходимо разложить его на меньшие модули, провести вычисления с помощью обычных методов, а затем вновь перейти к исходному модулю. Метод индексного счисления эффективен при обработке данных 6–10 бит, и когда модуль представляет собой простое число [3].

Проблема деления в СОК в общем виде привлекает внимание многих исследователей при разработке высокопроизводительных многомодульных арифметико-логических

устройств (АЛУ). Цифровые системы, построенные на основе арифметики СОК, могут сыграть важную роль в высокоскоростных системах обработки данных в режиме реального времени, с поддержкой параллельной обработки целочисленных данных [5]. Операции сложения, вычитания и умножения, называемые модулярными операциями, могут быть реализованы очень быстро, без распространения переносов [6]. Немодульные операции деления, сравнения чисел, определения знака, определения переполнения диапазона остаются сравнительно медленными [7]. Любое улучшение скорости этих медленных алгоритмов значительно улучшит производительность многомодульных АЛУ.

Некоторые алгоритмы деления в СОК в общем случае были разработаны ранее. Эти алгоритмы могут быть разделены на две категории: с использованием умножения и с использованием вычитания [7]. Большинство алгоритмов на основе умножения предварительно вычисляют обратную величину делителя, после чего эта величина умножается на делимое. Алгоритмы на основе вычитания используют вычитание кратных делителя из делимого до тех пор, пока полученный результат не будет меньше делителя. Некоторые известные алгоритмы деления в СОК на основе умножения изложены в [8]. Все эти алгоритмы используют преобразование в обобщенную позиционную систему счисления (ОПСС) для нахождения обратной величины делителя и сравнения чисел. Кроме того эти алгоритмы являются медленными, так как требуют выполнения большого количества арифметических действий. Некоторые алгоритмы на основе вычитания представлены в [8]. Эти алгоритмы не требуют вычислений в ОПСС, однако используют некоторые другие немодульные операции [4, 5]. Алгоритм с вычитанием, представленный в [9], кажется наиболее привлекательным для применения на практике, поскольку использует эффективный метод проверки четности для сравнения и обнаружения переполнения диапазона. Большинство существующих алгоритмов обладают различными недостатками, что делает их менее пригодными для решения задачи деления в СОК.

Литература

1. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах – М.: Советское радио, 1968. – 440 с.
2. Амербаев В.М., Стемпковский А.Л., Широ Г.Э. Модулярный быстродействующий согласованный фильтр // 50 лет модулярной арифметике. Юбилейная международная научно-техническая конференция. Сборник трудов. – М., 2000. – 775 с.
3. Содерстренд М.А., Верниа К. Недорогой быстродействующий умножитель по модулю и его применение в арифметических устройствах на основе системы счисления в остаточных классах // ТИИЭР, 1980. №4. Т.68.
4. Hung C., Parhami B. Fast RNS division algorithms for fixed divisors with application to RSA encryption // Information Processing Letters 51. 1994. P.163-169.
5. Hung C., Parhami B. An approximate sign detection method for residue numbers and its application to RNS division // Computers Math. Applic. 1994. Vol. 27. №4. P. 23-35.
6. Julien G.A. Residue number scaling and other operations using ROM arrays // IEEE Trans. on Comput. 1978. Vol. 27. №4. P. 325-336.
7. Claudio E.D., Piazza F., Orlandi G. Fast combinatorial RNS processors for DSP applications // IEEE Trans. Comput. 1995. Vol.44. №5. P. 624-633.
8. Червяков Н.И., Лавриненко И.Н., Лавриненко С.В., Мезенцева О.С. Методы и алгоритмы округления, масштабирования и деления чисел в модулярной арифметике

// 50 лет модулярной арифметике. Юбилейная международная научно-техническая конференция. Сборник трудов. М., 2000. С. 291-310.
9. Radhakrishnan D., Yuan Y. Novel approaches to the design of VLSI RNS multipliers // IEEE Trans. Circuits and System. 1992. Vol.39. №1. P.52-57.

РЕАЛИЗАЦИЯ ЕС-ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ТОЧКАХ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Н.И. Червяков, М.Г. Бабенко, Е.С. Кияшко, К.С. Шульженко

Северо-Кавказский федеральный университет, Ставрополь

Рассматриваются особенности построения генератора псевдослучайных чисел на основе ЕС-последовательностей. Приводится разработка нейронной сети для генерации псевдослучайных чисел с улучшеными статистическими характеристиками.

В повседневной жизни мы часто используем числа, случайным образом выбранные из некоторого множества. Для принятия решения мы подбрасываем монету. При компьютерном моделировании случайные числа позволяют сделать эти модели максимально похожими на реальные. Нашли свое применение случайные числа и в криптографии.

С появлением ЭВМ начались исследования эффективных способов получения случайных чисел. Числа, построенные с использованием некоторого алгоритма, только кажутся случайными. Они подчиняются некоторому правилу и корректнее называть их «псевдослучайными».

Основная сложность генерации псевдослучайных чисел на компьютере заключается в том, что он может находиться только в конечном, хотя и большом, числе состояний. Следовательно, любой генератор псевдослучайных чисел (ГПЧ) по определению периодичен, а всё периодическое неслучайно. Таким образом, задача ГПЧ – построение неотличимой от случайной последовательности с таким периодом разумной длины, чтобы она соответствовала различным критериям случайности.

В [1] говорится, что b-ичная последовательность $\{x_i\}$ является k-распределенной, если $P(x_n x_{n+1}...x_{n+k-1} = a_1 a_2 ... a_k) \approx \frac{1}{b^k}$ для всех b-ичных чисел $a_1 a_2 ... a_k$.

На основе понятия k-распределенности вводится понятие случайной последовательности [1].

Определение 1. b-ичная последовательность длины N случайна, если она k-распределена для всех положительных целых чисел $k \le \log_b N$.

В ходе исследований нами был реализован ГПЧ на точках эллиптической кривой. Данный выбор обусловлен удобством вычисления элементов таких последовательностей и их хорошими статистическими свойствами [2].

Пусть задано простое число $p > 3$. Тогда эллиптической кривой в форме Вейерштрасса [3], определенной над конечным простым полем F_p, называется множество пар чисел $(x, y), x, y \in F_p$, удовлетворяющих тождеству [5]

$$y^2 = x^3 + ax + b \pmod p, \qquad (1)$$

где $a, b \in F_p$ и $4a^3 + 27b^2$ несравнимо с 0 по модулю p.

Пары чисел (x, y), удовлетворяющие выражению (1), называются точками эллиптической кривой.

Введем операцию сложения двух произвольных точек $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$. Суммой $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$ будем называть точку $P_3(x_3, y_3)$, координаты которой определяются сравнениями

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 (\mathrm{mod}\, p) \\ y_3 = \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, p) \end{cases}, \tag{2}$$

где параметр λ определяется из сравнений

$$\begin{cases} \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, p), \ \textit{при} \quad x_1 \neq x_2; \\ \lambda = \dfrac{3x_1^2 - a}{2y_1} (\mathrm{mod}\, p), \ \textit{при} \quad x_1 = x_2 \quad \textit{и} \quad y_1 = y_2 \neq 0. \end{cases} \tag{3}$$

Для деления воспользуемся теоремой [4].

Малая теорема Ферма. Для всякого простого числа p и натурального числа a, такого, что $НОД(a, p) = 1$, имеет место равенство $a^{p-1} = 1 (\mathrm{mod}\, p)$.

В качестве экспериментальных данных нами были использованы 3 эллиптические кривые. Коэффициенты данных кривых, а также простые числа представлены в табл. 1. Кривая I была взята из [5], остальные подобраны в вычислительном эксперименте.

Таблица 1. Параметры исследуемых эллиптических кривых

	Значение
Кривая I	
a	7
b	43308876546767276905765904595650931995942111794451039583252968842033849580414
p	57896044618658097711785492504343953926634992332820282019728792003956564821041
Кривая II	
a	57797173260300591862887039728922683231990117698099726395553302688959615241517
b	75536999230121086036200401454318870573057067937024436778923171878238702848887
p	74071195152409579744075542254950126325253398248741892913923221064376398898449
Кривая III	
a	61030749271824800554991559575506208894416059169660377593696256977721982938 73
b	33290497613296215899503662933070548906818669667800999865470615866701901212386
p	67372339006199060004899298784615830852533077490386635296533159255966007376509

На основе данных кривых нами были сгенерированы последовательности из 100, 200,…,1000 точек. Над данными последовательностями был произведен статистический анализ на основе определения 1 и понятия k – распределенности.

Для выполнения условия случайности последовательности k должно удовлетворять неравенству $k \leq \log_b N$. Однако тестирование $2^{14} - 2^{17}$ подпоследовательностей – задача ресурсоемкая. Поэтому для анализа нами были выбраны подпоследовательности, состоящие только из 0 или только из 1. Обозначим их соответственно $A0$ и $A1$. В качестве критерия k – распределенности нами было взято выражение

$$\frac{1}{2^{k+1}} < P(x_n x_{n+1} \ldots x_{n+k-1} = a_1 a_2 \ldots a_k) \approx \frac{1}{2^k} < \frac{1}{2^{k-1}}. \tag{4}$$

Значения, для которых выполнялось данное условие, предоставлены в табл. 2.

Таблица 2. Значения k, удовлетворяющие (4) для $A0$ и $A1$

Количество точек	Номер кривой	Длина последовательности	Значения k для $A0$	Значения k для $A1$
100	*I*	22323	1-3	1-5
	II	22464	1-3	1-6
	III	22323	1-3	1-7
200	*I*	44712	1-3	1-6,15
	II	44853	1-3	1-6
	III	44755	1-3	1-8,14-15
300	*I*	67010	1-3	1-6,14-16
	II	67243	1-3	1-6
	III	67131	1-3	1-7,15-16
400	*I*	89353	1-3	1-6,15-16
	II	89566	1-3	1-6
	III	89705	1-3	1-7
500	*I*	111670	1-3	1-6,15-16
	II	111930	1-3	1-6
	III	112086	1-3	1-7
600	*I*	133995	1-3	1-7,15
	II	134381	1-3	1-6
	III	134590	1-3	1-7,17
700	*I*	156456	1-3	1-7
	II	156744	1-3	1-6
	III	156982	1-3	1-7,17
800	*I*	178727	1-3	1-7
	II	179073	1-3	1-6
	III	179474	1-3	1-7,17
900	*I*	201035	1-3	1-7
	II	201388	1-3	1-6
	III	201892	1-3	1-7,17
1000	*I*	223343	1-3	1-6
	II	223823	1-3	1-6
	III	224315	1-3	1-7,17

Исходя из данных результатов, можно сделать вывод, что ни одна кривая не удовлетворяет понятию случайности по определению 1. Однако видно, что кривые *I* и *III* показали результаты лучшие, чем *II*.

Применим метод из работы [6] для улучшения традиционных генераторов с помощью нейронных сетей. Ниже описан подход для создания устойчивых генераторов случайных чисел, которые будут использоваться в механизмах безопасности электронных систем торговли. В их основе будут лежать искусственные нейронные сети (ИНС) прямого распространения. Известно, что ИНС обладают очень интересными функциональными возможностями аппроксимации. Этот факт делает их мощным инструментом во многих научных дисциплинах. Например, у ИНС с прямой связью типа многослойного персептрона (MLP) есть теоретическая способность аппроксимировать произвольные нелинейные отображения так же, как их дифференциалы [9]. Предполагается, что такая аппроксимация ИНС является менее затратной, так как ей потребует меньше параметров, чем другим конкурентоспособным методикам (ортогональные полиномы, сплайны или прогрессия Фурье). Кроме того, так как ИНС – параллельные и распределенные

устройства обработки, то они могут быть осуществлены в параллельных аппаратных средствах. Следовательно, они могут использоваться для приложений в реальном времени.

Самым важным и интригующим свойством, которое делает ИНС полезными для приложений, является их способность обобщения. Она реализуется благодаря способности получать правильное решение, когда на входы поданы ранее не вводимые данные.

Вышеупомянутые способности приобретаются MLP путем обучения на заранее известных входных векторах. При этом должно выполняется условие, что не произошло переобучение [7]. С одной стороны, в случае переобучения [7] сеть очень хорошо знает обучающие выборки. С другой стороны, она неспособна выдать решение, когда поданы неизвестные входные данные. Такое случается, потому что сеть рисует соответствующую поверхность обучающих выборок более сложным способом. Это происходит в силу того, что она является поверхностью настройки и имеет более высокую степень, чем необходимо для отображения фактической выборки. Тогда мы не получаем аналитической формулы для того, чтобы описать ранее упомянутую сложную поверхность. В итоге ответ MLP становится непредсказуемым. Даже для неизвестных данных с маленькими расстояниями от обучающих выборок (похожие входные данные) выходы сети будут сильно отличаться от полученных учебных данных. Хотя в распознавании образов, управлении, прогнозе такая ситуация очень нежелательна, она может использоваться в случае генерации случайных чисел. В этой работе мы демонстрируем такое переобучение MLP, что сеть может эксплуатироваться как механизм для генерации устойчивых псевдослучайных последовательностей битов следующим образом.

1. Обучите MLP топологии, например 4-6-6-1, задаче кратности 4 так, чтобы переобучить, например, на 2500 эпохах, когда достаточно 500 эпох. Цель состоит в том, что переобучение должно произойти в процессе обучения MLP. Конечно, можно использовать другие подобные двоичные оценки, так же как и более сложную топологию MLP.

2. Проверьте обученную MLP, используя тестовые входные векторы. Их значения находятся в интервале [0,1]. Компоненты векторов произведены традиционным (псевдо) генератором случайных чисел, наподобие указанных в экспериментальном разделе работы.

3. Сформируйте комплексную функцию внутренних связей полученной MLP. Вычислите ее значение, когда тестовый входной вектор, сформированный заранее, подается на вход сети.

Таким образом, получается последовательность (псевдо) случайных чисел, качество которой количественно оценено с помощью статистических тестов. У комплексной функции внутренних представлений MLP, используемых в данной работе, есть следующая аналитическая формула

$$y = \mathrm{mod}\, f\left(1000 * \sum \frac{\left|\tanh\left(o_k\right) - \tanh\left(o_{k+1}\right)\right|}{\left|\tanh\left(o_{k+1}\right) - \tanh\left(o_{k+2}\right)\right|}\right),$$

где y – полученное случайное число и o_k, o_{k+1}, o_{k+2} – значения активации вычислительных элементов слоев MLP. mod$f$ – Unix-функция, которая извлекает дробную часть вещественного числа.

Применение нейронной сети позволяет генерировать последовательности на базе ЕС-последовательностей с улучшенными статистическими свойствами.

Литература

1. Кнут Д.Э. Искусство программирования. Том 2. – М.: Вильямс. 2002. 788 с.

2. Тараканов В.Е. Несколько замечаний об арифметических свойствах рекуррентных последовательностей на эллиптических кривых над конечным полем // Математические заметки. Том 82. Вып. 6, декабрь 2007.
3. Тараканов В.Е. Свойства делимости точек эллиптических кривых над конечным полем // Труды по дискретной математике. М.: Физматлит, 2001. Том 4. С. 243-258.
4. Болотов А.А., Гашков С.Б., Фролов А.Б., Часовских А.А. Алгоритмические основы эллиптической криптографии. М.: Издательство РГСУ, 2004. 499 с.
5. ГОСТ Р 34.10-2001.
6. Червяков Н.И. и др. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. М.: Физматлит. 2012. 280 с.
7. Patterson D.W. Artificial Neural Networks. Theory and Applications, Prentice Hall, 1996.
8. Червяков Н.И., Галушкин А.И., Евдокимов А.А., Лавриненко И.Н., Лавриненко А.В. Применение искусственных нейронных сетей и системы остаточных классов в криптографии М.: Физматлит, 2012. 280 с.
9. Hornik K., Stinchombe M., White H. Multilayer feedforward network are universal approximators // Neural Networks. 1989. 2. P. 359-366.

РЕШЕНИЕ ПРОБЛЕМЫ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ ОСТАТОЧНЫХ КЛАССОВ

Н.И. Червяков, М.Г. Бабенко, Е.С. Кияшко, К.С. Шульженко

Северо-Кавказский федеральный университет, Ставрополь

В статье проанализированы существующие методы дискретного логарифмирования, рассмотрено решение проблемы дискретного логарифмирования с использованием непозиционной системы счисления – системы остаточных классов.

В настоящее время существует широкий спектр важных для теории и практики математических задач, требующих вычислений с многоразрядными числами. Одной из таких задач является дискретное логарифмирование. Возникающая при этом проблема состоит в том, что обширные классы существующих методов и алгоритмов для решения этой задачи в рамках быстродействия обеспечиваемого современными вычислительными устройствами практически не могут быть реализованы.

Поиски новых путей повышения производительности вычислительных устройств привели исследователей к объективному выводу, что в этом направлении позиционная система счисления свои возможности исчерпала. Это объясняется тем, что при выполнении арифметических операций над числами, представленными в этой системе, возникает необходимость учета межразрядных переносов, что приводит к большим задержкам в случае обработки многоразрядных чисел. Поэтому для того чтобы значительно повысить производительность вычислительных устройств, необходимо применение систем счисления, лишенных подобного недостатка.

В свете сказанного актуален переход к непозиционным системам счисления, наиболее перспективной из которых является система остаточных классов. Эта система обладает множеством неоспоримых преимуществ в сравнении с позиционной системой, однако ее широкое применение сдерживается рядом причин: невозможность визуального сравнения чисел, трудность определения знака числа, выполнения операций масштабирования и деления произвольных чисел. Кроме того, если обрабатываются многоразрядные числа, то определенные трудности возникают при выполнении такой операции как модульное возведение в степень. Следовательно, для построения полноценной компьютерной арифметики на базе системы остаточных классов необходимо разработать эффективные алгоритмы выполнения этих операций

Для начала определим, что же такое дискретное логарифмирование, в чем состоит его проблема, и какие методы решения существуют на данный момент.

Дискретное логарифмирование (DLOG) –задача обращения функции g^x в некоторой конечной мультипликативной группе G.

Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны.

Для заданных g и a решение x уравнения $g^x = a$ называется дискретным логарифмом элемента a по основанию g. В случае, когда G является мультипликативной группой кольца вычетов по модулю m, решение называют также индексом числа a по осно-

ванию *g*. Индекс числа *a* по основанию *g* гарантированно существует, если *g* является первообразным корнем по модулю *m*.

Задача дискретного логарифмирования является одной из основных задач, на которых базируется криптография с открытым ключом. Классическими криптографическими схемами на её основе являются схема выработки общего ключа Диффи-Хеллмана, схема электронной подписи Эль-Гамаля, криптосистема Мэсси-Омуры для передачи сообщений (подробно эти криптографические схемы изложены в [1, 3]). Их криптостойкость основывается на предположительно высокой вычислительной сложности обращения показательной функции. Последняя вычисляется достаточно эффективно, в то время как даже самые современные алгоритмы вычисления дискретного логарифма имеют очень высокую сложность, которая сравнима со сложностью наиболее быстрых алгоритмов разложения чисел на множители.

Другая возможность эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями. Теоретически доказано, что с их помощью дискретный логарифм можно вычислить за полиномиальное время [2]. В любом случае, если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практическую непригодность криптосистем на его основе. Однако на данный момент, алгоритмов, способных вычислить дискретный логарифм за полиномиальное время, не существует.

Существует набор различных алгоритмов вычисления дискретного логарифма, с разной степенью сложности. Однако в данной работе будут затронуты только некоторые из алгоритмов, представляющие наибольший интерес.

Первый алгоритм, который будет рассмотрен в этой работе, – это P-метод Полларда для дискретного логарифмирования.

Этот алгоритм осуществляет случайный поиск дискретных логарифмов, как и метод «шаг ребенка – шаг великана», но он требует меньшего объема памяти для хранения данных, поэтому предпочтительней для практических целей [3]. В основе данного метода лежит та же идея, что и в основе p-метода для факторизации – строится псевдослучайная последовательность, в которой находятся совпадающие элементы при помощи метода Флойда [4], а затем на основании полученной величины вычисляется искомый дискретный логарифм.

Пусть требуется вычислить $\log_g a$ в конечной группе G порядка *n*.

Группа G разбивается на три непересекающихся подмножества примерно равной мощности: $G = S_1 U S_2 U S_3$, так чтобы $1 \notin S_2$. Причем разбиение должно быть построено таким образом, чтобы проверка, к какому подмножеству принадлежит данный элемент x, была простой.

Например, если $G = Z_p$, где p – простое число, то можно задать разбиение $S_1 = \left\{1,...,\left[\frac{p}{3}\right]\right\}, S_2 = \left\{\left[\frac{p}{3}\right],...,\left[\frac{2p}{3}\right]\right\}, S_3 = \left\{\left[\frac{2p}{3}\right],...,p-1\right\}$, или разбиение может быть таким: если $x \bmod 3 = 1$, то $x \in S_1$, если $x \bmod 3 = 2$, то $x \in S_2$, если $x \bmod 3 = 0$, то $x \in S_3$.

Далее на G задается последовательность $x_0, x_1, x_2, ...,$ где $x_0 = 1, x_{i+1}$ вычисляется по x_i посредством функции f для $i \geq 0$:

$$x_{i+1} = f(x_i) = \begin{cases} ax_i, если & x_i \in S_1; \\ x_i^2, если & x_i \in S_2; \\ gx_i, если & x_i \in S_3. \end{cases}$$

Вычисления проводятся в группе G, то есть если $G = Z_m$, то вычисления следует производить по модулю m.

Такая последовательность групповых элементов может быть представлена двумя последовательностями $u_0, u_1, u_2, \ldots$ и $v_0, v_1, v_2, \ldots$ такими, что $x_i = g^{u_i} a^{v_i}, u_0 = v_0 = 0,$

$$u_{i+1} = \begin{cases} u_i, \text{если } x_i \in S_1 \\ 2u_i, \text{если } x_i \in S_2 \\ u_i + 1, \text{если } x_i \in S_3 \end{cases}, \quad v_{i+1} = \begin{cases} v_i + 1, \text{если } x_i \in S_1 \\ 2v_i, \text{если } x_i \in S_2 \\ v_i, \text{если } x_i \in S_3 \end{cases}$$

Вычисления в последовательностях u и v производятся по модулю n.

В силу того, что группа G – конечная, при помощи метода Флойда можно найти такие x_i и x_{2i}, что $x_i = x_{2i}$. Тогда $g^{u_i} a^{v_i} = g^{u_{2i}} a^{v_{2i}} \Rightarrow a^{(v_i - v_{2i})} = g^{(u_{2i} - u_i)}$. Логарифмируя по основанию g обе части данного уравнения, получим:

$$(v_i - v_{2i}) \log_g a \equiv (u_{2i} - u_i) (\mathrm{mod}\ \ n).$$

Решая это сравнение, получим искомый логарифм. (Заметим, что если $G = Z^*_m$, то $n = \varphi(m)$).

Сложность данного метода составляет $O(\sqrt{n})$, где n – порядок группы G.

Замечание. Метод может дать отказ в том случае, когда $v_i = v_{2i}$. Тогда следует назначить случайные значения от 0 до n–1 переменным u_0, v_0, вычислить $x_0 = g^{u_0} a^{v_0}$ и повторить все шаги алгоритма.

Замечание. В том случае, когда имеется достаточно места для хранения данных в процессе вычислений (например, когда G невелико или вычисления производятся вручную), можно обойтись без метода Флойда. Тогда следует хранить все члены последовательностей x, u и v до того, как $x_i = x_j$ и дискретный логарифм находится из сравнения $(v_i - v_j) \log_g a \equiv (u_j - u_i) (\mathrm{mod}\ n)$.

Следующий алгоритм нахождения дискретного логарифма, который вызывает интерес, это метод Полига-Хеллмана.

Этот алгоритм использует следующий подход: пусть G – группа порядка n, и $n = p^{\alpha_1}{}_1 p^{\alpha_2}{}_2 \ldots p^{\alpha_k}{}_k$ – каноническое разложение на простые сомножители. Если $x = \log_g a \mod n$, то, вычислив $x_i = \log_g a \mod p^{\alpha_i}{}_i$, для $1 \le i \le k$, можно восстановить x, используя китайскую теорему об остатках.

Для того чтобы вычислить x_i, вычисляют коэффициенты $l_0, l_1, \ldots, l_{\alpha_k - 1}$ в p_i-чном представлении числа $x_i : x_i = l_0 + l_1 p_i + \ldots + l_{\alpha_k - 1} p^{\alpha_k - 1}{}_i$, где $0 \le l_j \le p_i - 1$.

Алгоритм Полига-Хеллмана:

Вход: g – порождающий элемент циклической группы порядка n, $a \in G$.

Ш.1. Найти каноническое разложение $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$.

Ш.2. Для i от 1 до k выполнить следующие действия:

1. Задать $q = p_i, \alpha = \alpha_i$.

2. Задать $\gamma = 1, l_{-1} = 0$.

3. Вычислить $\overline{g} = g^{\frac{n}{q}}$.

4. Для j от 1 до $\alpha - 1$ выполнить:

4.1. Вычислить $\gamma = \gamma g^{l_{i-1} q^{j-1}}, \overline{a} = (a\gamma^{-1})^{\frac{n}{q^{j+1}}}$

4.2. Вычислить $l_i = \log_{\bar{g}} \bar{a}$ (например, используя алгоритм «шаг младенца – шаг великана» или прямой поиск).

5. Вычислить $x_i = l_0 + l_1 q + ... + l_{\alpha-1} q^{\alpha-1}$.

Ш.3. Используя китайскую теорему об остатках, решить систему сравнений $x \equiv x_i (\mathrm{mod}\ \ p^{\alpha_i}{}_i)$.

Выход $x = \log_g a \bmod n$.

Замечание. Все вычисления производятся в группе *G* кроме случаев, когда оговорено иное.

Замечание. Поскольку порядок элемента $\bar{a} = (\bar{g})^{l_j}$ (в чем нетрудно убедиться, подставив вместо $\bar{a}$ его выражение из 4.2 и учитывая, что порядок $\bar{a}$ есть *q*), то $l_i = \log_{\bar{g}} \bar{a}$.

Заметим, что вычисление логарифма прямым поиском на этапе 4.2. происходит сравнительно быстро, так как приходится перебирать не более *q* значений.

Данный метод эффективен в случаях, когда *n* является большим числом, а все его простые сомножители – малыми числами.

Если на каждом этапе произвести преобразование чисел в СОК, то при моделировании криптосистемы на процессоре Intel Core i5 с использованием алгоритма Полига-Хеллмана на языке C++ время решения задачи дискретного логарифмирования над полем $2^{61} - 1$ уменьшается в среднем в 6 раз. Это возможно благодаря тому, что в этом алгоритме используются, преимущественно, операции умножения, выполнение которых в СОК происходит значительно быстрее, и как следствие, возрастает скорость выполнения всей операции дискретного логарифмирования.

Сложность данного алгоритма составляет $O(\sum_{i=1}^{k} \alpha_i (\log n + \sqrt{p_i}))$ умножений в группе при условии, что разложение *n* известно.

На основании приведенных методов решения проблемы дискретного логарифмирования, нетрудно убедиться, что при решении этих задач используются громоздкие и трудоемкие операции над числами. В работах [5, 6] изложены примеры того, как использование чисел в непозиционной системе счисления (в данном случае в системе остаточных классов) позволяет ускорить вычисления дискретного логарифма. Применение СОК на основных этапах позволит существенно быстрее осуществлять получение результата без существенного изменения алгоритмов [7-9].

Литература

1. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации. М.: Горячая Линия - Телеком, 2005. – 230 с.
2. Квантовые компьютеры и квантовые вычисления URL=http://cs.mipt.ru/docs/comp/rus/develop/other/quantum_comp/ дата доступа 30.09.2013
3. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография. М.: АНО НПО «Профессионал», 2005. – 480 с.
4. Левитин А.В. Алгоритмы: введение в разработку и анализ. М.: Издательский дом «Вильямс», 2006 – 576 с.
5. Червяков Н.И., Галушкин А.И., Евдокимов А.А., Лавриненко А.В., Лавриненко И.Н. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. М.: Физматлит. 280 с.
6. Omondi A., Premkumar B. Residue number systems: theory and implementation. Imperial College Press, 2007 – 256 p.

7. Червяков Н.И., Тынчеров К.Т., Велигоша А.В. Высокоскоростная цифровая обработка сигналов с использованием непозиционной арифметики // Радиотехника. 1997. № 10. С. 23-29.
8. Червяков Н.И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка, применение. 2006. № 10. С. 24-36.
9. Червяков Н.И. Ускоренный алгоритм определения позиционных характеристик и его нейросетевая реализация // Нейрокомпьютеры: разработка, применение. 2001. № 10. С. 19-21.

ИССЛЕДОВАНИЯ АЛГОРИТМА ГПЧ НА ОСНОВЕ БИЛИНЕЙНОГО СПАРИВАНИЯ НА ТОЧКАХ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ

Н.И. Червяков, М.Г. Бабенко, А.С. Назаров, И.С. Крисина

Северо-Кавказский федеральный университет, Ставрополь

Представлен алгоритм ГПЧ на основе билинейного спаривания на точках эллиптической кривой. Проводится иследование данного алгоритма при реализации в нейросетевом базисе.

Современные информационные системы требуют особых подходов к сохранению секрета. Одним из важных криптографических примитивов является генератор псевдослучайных чисел (ГПЧ). В настоящее время особую актуальность приобретают ГПЧ, построенные на точках эллиптической кривой. Это объясняется тем, что они удовлетворяют двум критериям – криптографической и статистической безопасности. Первый критерий получается вследствие того, что эллиптическая кривая позволяет:

1) строить криптографические протоколы с меньшей длиной ключа,

2) сохранять при этом уровень безопасности разработанного шифра.

Вследствие всего вышесказанного особую актуальность приобретает научная задача: разработка нового алгоритма построения ГПЧ, сочетающего в себе высокое быстродействие, хорошие статистические и криптографические свойства формируемой выходной последовательности.

В генерации и использовании случайных чисел нуждаются:

- механизмы целостности (ISO 8731-2 [7]);
- криптографические механизмы обмена ключами (протокол [8] Diffie-Hellman);
- схемы построения цифровых подписей (схема ElGamal);
- схема цифровой подписи (DSS) [9].

Кроме того, случайные числа используются для генерации псевдонимов, трафика и общения приложений. Это обеспечивает защиту от атак анализа трафика. Такой подход позволяет вычислить устойчивые и эффективные потоковые шифры [9].

Желательны случайные последовательности битов хорошего качества, то есть хорошего поведения в статистическом смысле и смысле непредсказуемости. Иначе, для криптоанализа существовала бы возможность вычислить последующие биты, учитывая сегмент этой битовой последовательности и располагая разумными компьютерными ресурсами [10]. За прошлые два десятилетия была проведена значительная работа по созданию и анализу генераторов псевдослучайных чисел или последовательностей битов [10].

Уровень случайности последовательности может быть определен с помощью статистических тестов. Они эмулируют вычисления, встречаемые на практике. Эти тесты позволяют проверить то, что при исследовании свойства последовательности совпадают с предсказанными, если каждый бит (или число) был взят в соответствии с равномерным распределением вероятности [11]. При разработке безопасные генераторы ключа должны удовлетворять определенным критериям, таким как длительный период, идеальное распределение k-кортежа, большая линейная сложность, хаотичность, рас-

пространение и нелинейность [11]. Большинство из них содержится в предложенной методике тестирования.

Теоретическая статистика предоставляет некоторые количественные меры случайности. Существует буквально бесконечное число критериев проверки случайности последовательности [6].

Тест на случайное блуждание оказывается одним из наиболее мощных и чувствительных тестов на наличие корреляций в ГПЧ. В частности, тест с использованием идеи случайного блуждания был одним из тестов, выявивших корреляции в ГПЧ типа «сдвиговой регистр» [1]. Другой тест на случайное блуждание позволил объяснить природу этих корреляций [3].

Существует несколько вариаций теста на случайное блуждание в различном количестве измерений [2]. Была рассмотрена одномерная модель направленного случайного блуждания [3].

Алгоритм теста.

Шаг 1.

1.1. Однонаправленное блуждание начинается в некотором узле одномерной решетки при дискретных значениях времени *i* с вероятностью $1-\mu$.

1.2. Однонаправленное блуждание останавливается с вероятностью μ. Переход на шаг 2.

Шаг 2. В случае 1.2 переход на шаг 1. Начало нового блуждания.

Вероятность блуждания длинной n равна $P(n)=\mu^{n-1}(1-\mu)$. Средняя длина блуждания равна $\langle n\rangle=\dfrac{1}{1-\mu}$. Следует отметить, что рассмотренная модель представляет собой эффективный метод Вольфа для одномерной модели Изинга [3]. Это можно увидеть из того, что средний размер кластера в методе Вольфа равен средней длине блуждания для $\mu=\tanh\left(\dfrac{J}{k_B T}\right)$, где J – константа связи спинов.

Для построения схемы разделения секрета введем операцию спаривания на эллиптической кривой, аналогичную рассмотренной в работе [4]. Для любой точки $P\in E\left(F_{p^k}\right)$ выполняется равенство $(n+1)P=O$, где O – точка в бесконечности. Обозначим $E\left(F_{p^k}\right)$ множество точек эллиптической кривой, которая задана уравнением $y^2=x^3+ax+b$, над полями F_{p^k}. Для построения спаривания важно знать структуру группы точек эллиптической кривой. Ниже приведена теорема, описывающая структуру группы, когда эллиптическая кривая задана над полем F_{p^k}, где p – простое число, $k\geq 1$ и порядок группы точек эллиптической кривой выражается формулой $E\left(F_{p^k}\right)=p^k+1-t.$

Теорема 1 [5]. Если $t^2=p^k, 2p^k, 3p^k$, то группа – циклическая. Если $t^2=4p^k$, то группа изоморфна $Z_{\sqrt{p^k}-1}\oplus Z_{\sqrt{p^k}-1}$ в случае $t=2\sqrt{p^k}$, или изоморфна $Z_{\sqrt{p^k}+1}\oplus Z_{\sqrt{p^k}+1}$ в случае $t=-2\sqrt{p^k}$.

Если $t=0, p^k\neq 3(\bmod 4)$, то группа циклическая. Если $t=0, p^k\equiv 3(\bmod 4)$, то группа или циклическая или изоморфна $Z_{\frac{p_i^k+1}{2}}\oplus Z_2$.

Из теоремы 1 следует, что в случае, если $t^2 = 4p^k$ и r – четное число, группа точек $E\left(F_{p^k}\right)$ представляется в виде прямой суммы $Z_{\sqrt{p^k}-1} \oplus Z_{\sqrt{p^k}-1}$ или $Z_{\sqrt{p^k}+1} \oplus Z_{\sqrt{p^k}+1}$. Обозначим эти группы $E[l]$, где $l = \sqrt{p^k} - 1$ или $l = \sqrt{p^k} + 1$.

Так как $E[l]$ представляется в виде прямой суммы циклических групп, то можно зафиксировать некоторую образующую пару точек G и H таким образом, чтобы любую точку в $E[l]$ можно представить с их помощью. Рассмотрим точки $P = a_1 G + b_1 H$ и $Q = a_2 G + b_2 H$, принадлежащие $E[l]$, где $a_1, a_2, b_1, b_2 \in [0, l-1]$. Для некоторых зафиксированных целых $\alpha, \beta \in [0, l-1]$ определим функцию следующим образом:

$$L_{\alpha,\beta} : E[l] \times E[l] \to E[l] \text{ и } L_{\alpha,\beta}(P,Q) = (a_1, b_1 - a_2, b_2)(\alpha G + \beta H)$$

исключая тривиальный случай, когда α и β одновременно равны нулю.

Пусть G_1, G_2 и G_3 – три абелевы группы. Билинейное спаривание является отображением $e: G_1 \times G_2 \to G_3$ среди этих трех групп. Это отображение должно удовлетворять свойству билинейности: для $\alpha, \beta \in G_1, \gamma, \delta \in G_2$ справедливо $e(\alpha + \beta, \gamma) = e(\alpha, \gamma) + e(\beta, \gamma), e(\alpha, \gamma + \delta) = e(\alpha, \gamma) + e(\alpha, \delta)$.

Следующая теорема показывает, что функция $L_{\alpha,\beta}$ задает билинейное спаривание.

Теорема 2 [4]. Функция $L_{\alpha,\beta}$ обладает следующими свойствами:

1. Тождественность.

Для всех $P \in E[l]$, $L_{\alpha,\beta}(P,P) = O$.

2. Билинейность.

Для всех $P, Q, R \in E[l]$

$$L_{\alpha,\beta}(P+Q,R) = L_{\alpha,\beta}(P,R) + L_{\alpha,\beta}(Q,R)$$

$$\text{и } L_{\alpha,\beta}(P,Q+R) = L_{\alpha,\beta}(P,Q) + L_{\alpha,\beta}(P,R).$$

3. Ассиметричность.

Для всех $P, Q \in E[l]$

$$L_{\alpha,\beta}(P,Q) = -L_{\alpha,\beta}(P,Q).$$

4. Невырожденность.

Для всех $P \in E[l]$

$$L_{\alpha,\beta}(P,O) = O.$$

Кроме того, если $L_{\alpha,\beta}(P,Q) = O$ для всех $Q \in E[l]$, тогда $P = 0$.

Функция $L_{\alpha,\beta}$ называется спариванием, поскольку она отображает $E[l] \times E[l] \to E[l]$ (аналогично традиционным спариваниям Вейля и Тейта).

На базе выше изложенного спаривания предложим алгоритм построения ГПЧ.

Алгоритм ГПЧ.

Вход: p -простое число, $a, b, \alpha, a_1, b_1 \in F_p$, где α - квадратичный невычет в F_p^*

Выход: ГПЧ

1. Для $i = 1$ до $n-1$ выполнять: $(a_{i+1}, b_{i+1}) = (a_1, \alpha b_1)(a_i, b_i)$.

2. Вывод $a_n G + b_n H_n$.

3. Конец.

Условием для того, чтобы этот алгоритм мог построить последовательность максимального периода p^2-1, является $ord\{a_1 x+b_1\}=p^2-1$ в поле $F_{p^2}^*=\{ax+b \mid x^2=\alpha,\alpha,a,b\in F_p^*\}$, где α - квадратичный невычет в F_p^*.

Рассмотренный выше алгоритм построения ГПЧ был реализован в среде программирования Microsoft Visual Studio 2010 на языке программирования C++. Затем над построенным ГПЧ был проведен тест на наличие корреляций «Критерий направленного случайного блуждания». Алгоритм теста был описан выше.

Было произведено 20 χ^2-проверок по критерию направленного случайного блуждания с $\mu=\frac{1}{2}$. Был применен «Алгоритм Р» случайного блуждания:

1. При $r<\mu$ совершается шаг вправо.
2. При $r\geq\mu$ остановка (где r – случайное число, порожденное генератором).

Таким образом, $\mu=\frac{1}{2}$ означает, что в проверках участвует один бит генератора. В результате проведения теста корреляции направленного блуждания были получены результаты для генератора, построенного на основе билинейного спаривания точек эллиптической кривой. Распределение выходных величин незначительно отличается от χ^2-распределения. Каждый из двадцати тестов пройден в том смысле, что получены вполне приемлемые результаты построенного генератора псевдослучайных чисел.

Из всего вышесказанного можно сделать вывод, что предложенный ГПЧ обладает большим периодом. Проведенные тесты показали, что построенный ГПЧ обладает хорошими статистическими свойствами.

В работе [12] представлены новые методики, основанные на использовании искусственных нейросетевых архитектур для улучшения традиционных генераторов, таких как ANSI X.9, базовых для DES и IDEE. Также в этой работе предлагается методика тестирования с использованием нейронных сетей, разработанная для оценки свойства непредсказуемости. Этот тест непредсказуемости, наряду с обычно используемыми статистическими тестами и тестами нелинейности, предложен как методология оценки устойчивости генераторов псевдослучайных чисел. Посредством этой методологии оценены классические и предложенные генераторы. Показано, что предложенные генераторы ведут себя значительно лучше, чем традиционные, в частности в терминах непредсказуемости [13].

По результатам применения методики тестирования данного ГПЧ показано, что разработанный генератор псевдослучайных чисел обладает хорошими статистическими свойствами.

Литература

1. Vattulainen I., Ala-Nissila T., Kankaala K // Phys. Rev. Lett. 73, 2513, 1997.
2. Binder K., Heermann D.W. Monte Carlo simulation in statistical physics. Springer-Verlag, Berlin, 1992.
3. Shchur L.N., Heringa J.R., Blöte H.W // Physica A241, 1997. P. 579.
4. Lee H.-S. A self-pairing map and its applications to cryptography // Applied Mathematics and Computation. 2004. Vol. 151. P. 671–678.
5. Болотов А. А., Гашков С. Б., Фролов А. Б., Часовских А. А. Алгоритмические основы эллиптической криптографии. М.: Изд-во МЭИ, 2004. 499 с.
6. Кнут Д. Искусство программирования. Том 2. Получисленные алгоритмы. – М.: Издательский дом «Вильямс», 2001. 832 с.

7. ISO 8731-2, «Approved Algorithms for Message Authentication, Part 2: Message Authenticator Algorithm (MAA)».
8. Diffie W., Hellman M.E. New directions in cryptography // IEEE Transactions on Information Theory. 1976. Vol. 22, No 6. P. 644-654.
9. Schneier B. Applied Cryptography. J. Willey & Sons, second edition, 1996.
10. Zeng K., Yang C.-H., Wei D.-Y., Rao T.R.N. Pseudo random bit generators in stream cipher cryptography // IEEE Computer, 1991. P. 8-17.
11. Simmons G. J. (editor). Contemporary Cryptology. The Science of Information Integrity // IEEE Press, 1992.
12. Червяков Н.И. и др. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. М.: Физматлит, 2012. 280 с.
13. Karras D.A., Zorkadis V. Improving pseudorandom bit sequence generation and evaluation for secure internet communications using neural network techniques // IEEE, 2003. P. 1367–1372.

ИССЛЕДОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ АЛГОРИТМОВ МОДУЛЯРНОГО ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЯ

Н.И. Червяков, Е.С. Карнаухова

Северо-Кавказский федеральный университет, Ставрополь

Предлагается реализация алгоритма вейвлет-преобразования на основе использования приближенного метода. Применение алгоритма позволяет сократить время выполнения операции в кодировании и декодировании изображения в 1,31 – 2,72 раз соответственно по сравнению с алгоритмами, работающими на традиционной арифметики.

Введение

Построение эффективных методов и алгоритмов вейвлет-анализа для устройств распознавания и классификации образов, работающих в реальном времени. В последнее время такие устройства всё чаще реализуют с использованием ПЛИС-технологий. Однако сами вендоры ПЛИС-схем признают, что арифметические возможности их устройств являются очень слабыми. Умножитель общего назначения бит (MAC-модуль) ПЛИС проигрывает во много раз хорошо спроектированным проблемно-ориентированным заказным интегральным схемам. Вдобавок, недостатки ПЛИС растут в геометрической прогрессии с увеличением точности вычислений. Эти причины стимулируют разработчиков прикладных решений синтезировать альтернативные вычислительные структуры. Одной из популярных технологий создания эффективных схем фильтрации при априорно известных коэффициентах фильтрации является так называемая битовая арифметика в дополнительном коде или БАДК. БАДК позволяет реализовать алгоритм фильтрации с помощью выборок из LUT-таблиц, выходы которых будут накапливаться сдвиговым сумматором. Проблемы организации необходимого диапазона предлагается решать с помощью модулярных вычислений в системе остаточных классов. СОК позволяет проводить вычисления с помощью нескольких параллельных и не взаимодействующих друг с другом малоразрядных каналов. Эта возможность, наряду с остальными её преимуществами, делает СОК потенциально привлекательным для реализации устройств цифровой обработки сигналов.

Основная часть

Для вейвлет-преобразования функции $f(x)$ необходимо вычислить серию коэффициентов $\{a_n, d_n, d_{n-1}, \ldots, d_1\}$, где a_n – аппроксимация функции, d_i – детализирующие коэффициенты функции, $i = 1, \ldots, n$. Каждый коэффициент находится интегрированием (1, 2):

$$a_{J-N,k} = \left(f, \varphi_{J-N,k}\right) = \int_R f(x)\overline{\varphi_{J-N,k}(x)}dx; \quad (1)$$

$$d_{J-m,k} = \left(f, \psi_{J-m,k}\right) = \int_R f(x)\overline{\psi_{J-m,k}(x)}dx, \quad m = 1, 2, \ldots, N. \quad (2)$$

Возникает проблема вычисления большого количества интегралов с необходимой точностью. Следует также учитывать, что при высоком уровне разрешения J носители функций $\varphi_{J,k}(x)$ и $\psi_{J,k}(x)$ становятся малыми порядка $J/2$.

Быстрое вейвлет-преобразование, предложенное Малла, позволяет решить эту проблему. Алгоритм Малла даёт возможность вычислять коэффициенты вейвлет-разложения без интегрирования, используя алгебраические операции на основе свёртки:

$$\begin{aligned} a_n^{(i)} &= \sum_{k=0}^{N-1} g_k a_{2n-k}^{(i-1)} \quad i = 1,2,\ldots,J; \\ d_n^{(i)} &= \sum_{k=0}^{N-1} h_k a_{2n-k}^{(i-1)} \quad a_n^{(0)} \equiv x_n, \end{aligned} \tag{3}$$

где $a_n^{(i)}$ и $d_n^{(i)}$ являются аппроксимирующими и детализирующими коэффициентами i-го уровня, а g_k и h_k $(k = 0,1,\ldots,N-1)$ – коэффициенты низкочастотного и высокочастотного анализирующих фильтров соответственно; x_n – исходный сигнал; N – порядок фильтра.

Эти равенства обеспечивают быстрые алгоритмы вычисления вейвлет-коэффициентов (каскадные алгоритмы, алгоритмы Малла). Термин «быстрые» означает не только, что в (3) используются более быстрые алгебраические процедуры, но и то, что при каждом преобразовании общее число новых коэффициентов не увеличивается в два раза, а остаётся прежним.

Алгоритмически выражение (3) можно представить так:

```
Начало {Алгоритм Малла}
 Для (i от 1 до J)
 Для (n от 1 до 2^(J-i))
```

$a_n^{(i)} = \sum_{k=0}^{N-1} g_k a_{2n-k}^{(i-1)}$ проход низкочастотным фильтром

$d_n^{(i)} = \sum_{k=0}^{N-1} h_k a_{2n-k}^{(i-1)}$ проход высокочастотным фильтром

```
Конец {Алгоритм Малла}
```

Схема разложения сигнала по алгоритму Малла приведена на рис. 1.

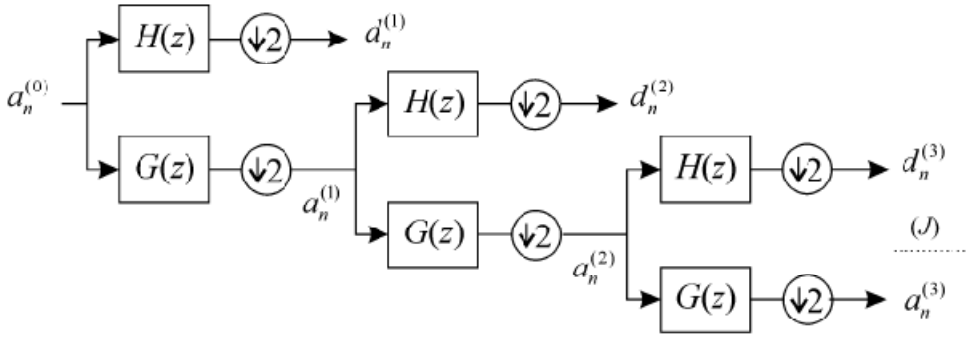


Рис. 1. Последовательность получения вейвлет-коэффициентов третьей октавы; $H(z)$ и $G(z)$ соответственно, высокочастотные и низкочастотные анализирующие фильтры в z-представлении

Для реализации алгоритма Малла необходимы арифметические операции сложения и умножения чисел, что может быть эффективно реализовано в системе остаточных классов. Однако для распознавания образов очень важным является выполнения операции сравнения чисел, которые можно выполнить с использованием приближенного метода в системе остаточных классов [1].

Суть приближенного метода сравнения модулярных чисел основана на использовании относительных величин анализируемых чисел к полному диапазону, определяемому китайской теоремой об остатках [1], которая связывает позиционное число A с его представлением в остатках $(\alpha_1,\alpha_2,\ldots,\alpha_n)$, где α_i – наименьшие неотрицательные вычеты числа, относительно модулей системы остаточных классов $p_1,p_2,\ldots,p_n$ следующим выражением:

$$A = \left| \sum_{i=1}^{n} \frac{P}{p_i} \left| P_i^{-1} \right|_{p_i} \alpha_i \right|_P , \qquad (4)$$

где $P = \prod_{i=1}^{n} p_i$, p_i – модули СОК, $\left|P_i^{-1}\right|$ – мультипликативная инверсия P_i относительно p_i, и $P_i = \dfrac{P}{p_i} = p_1 p_2 \ldots, p_{i-1}\, p_{+1}\, p_n$.

Если левую и правую части выражения (2) разделить на величину P, соответствующую диапазону чисел, то получим приближенное значение

$$\left|\frac{A}{P}\right|_1 = \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \alpha_i \right|_1 \approx \left| \sum_{i=1}^{n} k_i \alpha_i \right|_1 , \qquad (5)$$

где $k_i = \dfrac{\left|P_i^{-1}\right|_{p_i}}{p_i}$ – константы выбранной системы, а α_i – разряды числа, представленного в СОК, при этом значение каждой суммы будет в интервале [0,1). Конечный результат суммы определяется после суммирования и отбрасывания целой части числа с сохранением дробной части суммы. Дробная часть $F(A) = \left|\frac{A}{P}\right|_1 \in [0,1)$ содержит информацию как о величине числа, так и о его знаке. Если $\left|\frac{A}{P}\right|_1 \in \left[0, \frac{1}{2}\right)$, то число A положительное и $F(A)$ равна величине A, разделенной на P. В противном случае A – отрицательное число и $1–F(A)$ показывает относительную величину числа A по отношению к P. Исследования показали, что функция $F(A)$ может быть использована при разработке алгоритмов вычисления основных проблемных операций в системе остаточных классов [2]. Целая часть числа представляет собой ранг числа, то есть такую непозиционную характеристику, которая показывает, сколько раз диапазон системы P был превзойден при переходе от представления чисел в системе остаточных классов к его позиционному представлению. При необходимости определение ранга числа может производиться непосредственно в процессе выполнения операции суммирования констант k_i. Дробная часть может быть записана также как $A\bmod 1$, потому что $A = \lfloor A \rfloor + A\bmod 1$ [4]. Количество разрядов дробной части числа определяется максимально возможной разностью между соседними числами. При необходимости точного сравнения необходимо вычислить значение (3), которое является эквивалентом преобразования из СОК в позиционную систему счисления. Для решения задач основных процедур принятия решения достаточно знать приблизительно значения чисел $\left|\frac{A}{P}\right|_1$ и $\left|\frac{B}{P}\right|_1$ по отношению к динамическому диапазону [0,1), которое выполняется достаточно просто, но при этом верно определяется соотношениями $A=B$, $A>B$ или $A<B$.

Итак, приближенный метод вычисления ПХ может описываться следующей последовательностью действий [3]:

1. Вычисление констант СОК $k_i = \dfrac{\left|P_i^{-1}\right|_{p_i}}{p_i}$ с требуемой точностью.
2. Вычисление приближенных значений $\alpha_i k_i$, где k_i – константы найденные в п. 1, $1 \le \alpha_i \le p_i\text{-}1$.
3. Вычисление приближенного значения ПХ $\left|\sum_{i=1}^{n} k_i \alpha_i\right|_1$ в интервале [0,1).

Заключение

Разработанные модели, методы и алгоритмы реализуются в среде C++ и проводится сравнительная оценка временных показателей при использовании различных систем счисления. Для моделирования алгоритмов вейвлет-преобразования на основе использования приближенного метода были выбрана СОК с основаниями {2,3,5,7}.

Предложенный подход к реализации алгоритмов вейвлет-преобразований и распознавания образа позволяет расширить область применения модулярной арифметики в области цифровой обработки сигнала для задач распознавания изображений. Полученные новые результаты эффективного выполнения немодульных процедур являются развитием теории параллельных вычислений, основ разработки и проектирования высокопроизводительных и надежных вычислительных систем, функционирующих в системе остаточных классов. Применение предложенного алгоритма позволяет сократить время выполнения операции в кодированиии и декодировании изображения в 1,31 – 2,72 раз соответственно по сравнению с алгоритмами, работающими на традиционной арифметики.

Литература

1. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. – М.: Советское радио, 1968. – 440 с.
2. Hung C.Y., Parhami B. An Approximate Sign Detection Method for Residue Numbers and its Application to RNS Division // Computers Math. Applic. – 1994. – Vol. 27. – № 4. – P. 23-35.
3. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникационные технологии. – 2011. – №4. – С. 4-12.

О ЗАДАЧЕ КЛАССИФИКАЦИИ РАКОВЫХ ЗАБОЛЕВАНИЙ НА ОСНОВАНИИ ДАННЫХ О МЕТИЛИРОВАНИИ ДНК

И.Б. Крылов[1], М.В. Иванченко[1], А.А. Заикин[1,2]

[1]Нижегородский государственный университет им. Н.И. Лобачевского
[2]Университетский колледж Лондона

Кратко описаны подходы к изучению данных о метилировании ДНК и его связи с развитием онкологических заболеваний. Также изложена суть поиска наибо-лее значимых генов, шаблоны метилирования которых в значительной степени ме-няются при появлении ракового заболевания. Решается задача классификации, полученная точность составляет более 95 процентов.

Введение

Сегодня очень важной задачей является создание новых механизмов ранней диагностики онкологических заболеваний. Ведь именно ранняя диагностика и своевременно начатое лечение заболевания в разы увеличивают шансы на выздоровление. В настоящее время диагноз ставится преимущественно на поздних стадиях, что приводит к плохому исходу.

Выявление новых факторов риска рака для разработки чувствительных диагностических методов невозможно без фундаментальных исследований в области системной биологии, биоинформатики и методов анализа данных. Однако по-настоящему комплексных исследований, которые включали бы все эти направления, пока не ведется. Уникальность нашего подхода состоит в разработке методов, основанных на системном исследовании данной внутригенной и межгенной структуры метилирования ДНК с наивысшим на сегодняшний день разрешением в 485 000 точек на геном.

1. Поиск наиболее значимых генов и мер метилирования

В имеющихся данных содержится информация по 13 типам рака, данные взяты из базы данных с открытым доступом The Cancer Genome Atlas [1]. Будем рас-сматривать каждый тип рака в отдельности и решать задачу бинарной классификации, т.е. соотносить набор данных, содержащий значение меры метилирования для каждого гена, с классами «болен» и «здоров».

Решение данной задачи для каждого из 13 типов рака по отдельности является первым этапом исследований. Результатом решения данной задачи будет, во-первых, классификатор, способный давать ответ вопрос: «Болен ли этот человек раком конкретного типа?». Во-вторых, мы можем определить, шаблоны метилирования каких генов значительным образом различны, если пациенты принадлежат к разным группам «болен» или «здоров».

Будем решать задачу, рассматривая каждый ген в отдельности. Рассмотрим путь решения пошагово:

1. Исходные данные делим на обучающую и тестовую выборки.
2. Для каждого гена:

a. Применяем метод оптимизации для настройки параметров логистической регрессии (стохастический градиентный спуск), максимизируя функцию правдоподобия. Для усадки параметров пользуемся регуляризацией.
b. Строим ROC-кривую (англ. receiver operating characteristic, операционная характеристика приёмника) [2].
c. Высчитываем AUC (англ. area under ROC curve, площадь под ROC-кривой) [3].

3. Сортируем список генов по убыванию их AUC.
4. Устанавливаем минимальный проходной порог для генов AUCmin = 0.90 и исключаем из списка те гены, которые имеют AUC меньше, чем AUCmin.

В результате выполнения такой последовательности действий для каждого типа рака, мы получаем индивидуальный список генов для каждого вида онкологии, шаблоны метилирования которых значительным образом различны (в соответствии с мерой метилирования) в случаях наличия и отсутствия заболевания.

2. Решение задачи классификации

Более сложной задачей является задача отнесения конкретного набора данных об уровне внутригенного метилирования к классу «здоров» или «тип рака 1…13», в данном случае нас интересует, какое конкретное заболевание из списка имеет пациент. Для решения данной задачи используем модель «софтмакс»-регрессии [4], позволяющий оценить вероятность наличия того или иного вида заболевания. Для параметрической оптимизации используется метод стохастического градиентного спуска.

Заключение

В результате работы был выявлен ряд наиболее эффективных мер из рассматриваемого набора и список генов, на метилирование которых стоило бы обратить особое внимание при решение задачи классификации. Был построен классификатор, который не только может говорить о наличии ракового заболевания, но и идентифицировать его тип с высокой точностью, более чем 95%.

Литература

1. The Cancer Genome Atlas homepage. NCI and the NHGRI. Retrieved 2009-04-28.
2. Signal detection theory and psychophysics. – New York, NY: John Wiley and Sons Inc., 1966.
3. Lobo, Jorge M.; Jiménez-Valverde, Alberto; and Real, Raimundo (2008), AUC: a misleading measure of the performance of predictive distribution models // Global Ecology and Biogeography. 17: 145–151
4. Greene, William H., Econometric Analysis. Fifth edition. Prentice Hall, 1993. P. 720-723.

АЛГОРИТМЫ УСКОРЕНИЯ ПРОЦЕССА ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ В ПАРАЛЛЕЛЬНОЙ СИСТЕМЕ ПОСТОБРАБОТКИ SCIENTIFICVIEW

А.Л. Потехин, В.В. Ломтев, И.В. Логинов

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

Введение

В рамках проекта «Развитие суперкомпьютеров и грид-технологий» в ИТМФ РФЯЦ-ВНИИЭФ был разработан пакет программ, предназначенный для 3D имитационного моделирования задач из области автомобиле- и авиастроения, атомной энергетики, нефтегазодобычи, космоса, экологии. Кроме собственно программ для моделирования таких процессов, как аэрогидромеханика, тепломассоперенос, прочность и других, в пакет вошли программы, обеспечивающие препостпроцессинг расчетных данных. В качестве основной программы для параллельной постобработки и визуализации данных была разработана и развивается в настоящий момент система ScientificView.

В качестве входных для ScientificView могут выступать данные, заданные на структурированных и неструктурированных сетках, состоящих из нескольких млрд. ячеек и сохраненных в форматы коммерческих систем математического моделирова-ния d3Plot, nGeom, Cgns, VTK, а также в ряд собственных форматов.

Для проведения обработки результатов моделирования пользователю ScientificView доступны:

- процедуры графической обработки данных (более 30 алгоритмов: различные вырезы, сечения, построение полей, поверхностей, а также узкоспециализированные алгоритмы, например алгоритмы для обработки моделирования таких задач, как гармонический анализ и расчет турбомашин);
- процедуры числовой обработки данных (табличное представление, калькуляция величин, вычисление интегральных характеристик, построение графиков и зависимостей величин);
- средства создания видеоматериалов, различные сервисные возможности.

Кроме разработки методов фильтрации данных особое внимание уделяется производительности модуля формирования изображения, т.е. той части системы, с которой непосредственно взаимодействует пользователь. Описанию алгоритмов, направленных на ускорение процесса формирования изображения, и посвящена данная работа.

1. Распространенные подходы для решения задачи ускорения процесса формирования изображения

На практике в компьютерной графике наиболее широко применяются следующие подходы:

1. Классические (списки отображения, стрипование, VBO, шейдеры и т.д.). В целом подходы направлены на ускорение процесса передачи информации на видеокарту в наиболее благоприятном для нее виде и получение таким образом максимальной производительности.

2. Cull-алгоритмы. Предназначены для выявления и полного либо частичного блокирования передачи на GPU тех фрагментов сцены, которые не видны зрителю в данном ракурсе просмотра (находятся «сзади» камеры, закрыты другими объектами и т.д.).

3. Level Of Details – создание упрощенных клонов объектов сцены, содержащих меньшее число многоугольников. Механизмы принятия решения об использовании упрощенных моделей в ряде случаев.

4. Распараллеливание алгоритмов формирования кадра непосредственно на GPU или посредством MPI/OpenMPI.

2. Алгоритмы, реализованные в ScientificView на предыдущих этапах развития

В ScientificView используется ряд способов классической оптимизации, но поскольку их производительность ограничена ресурсами GPU, а значит, не позволяет принципиально эффективно справляться со сценой любой, сколько угодно большой сложности, разработан ряд алгоритмов из Cull- и LOD-групп.

Ранее в системе был создан алгоритм (далее алгоритм A1), который для замкнутых трехмерных объектов определяет набор полигонов с нормалью, направленной от зрителя и блокирует их отображение. В подавляющей массе случаев использование алгоритма позволяет ускорить процесс формирования кадра в 2 раза.

При проведении пользователем манипуляций с отображаемой сценой (повороты, сдвиги, масштабирование) важно максимально быстро получить итоговый результат, при этом качеством изображения можно временно пренебречь. В ScientificView для всех отображаемых объектов строится упрощенная модель с заданным пользователем уровнем детализации (всего доступно 4 уровня детализации – от «очень грубо», до «очень подробно»). Эта модель применяется системой при манипуляциях пользователя для более быстрого формирования изображения (далее этот алгоритм будем называть A2).

Однако как только пользователь перестает манипулировать со сценой, время формирования кадра снова становится достаточно высоким. Поэтому ScientificView на основе алгоритма «обратной связи» OpenGL определяет пиксельные размеры отображаемых объектов и с учетом ряда критериев принимает решение об использовании упрощенной модели и в некоторых случаях «статического» формирования изображения (далее этот алгоритм будет называться A3).

Ниже приведен график изменения времени формирования кадра при применении разного набора алгоритмов для одной из характерных задач среднего уровня сложности – около 5 млн. полигонов (рис. 1).

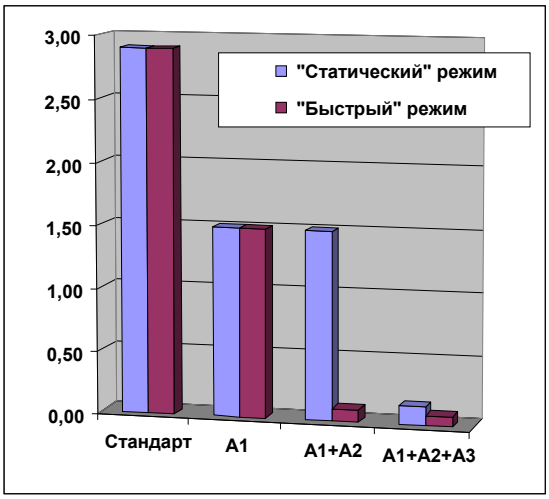


Рис.1. Время формирования кадра (в секундах) при манипуляциях со сценой («быстрый» режим) и при формировании полноценной картинки («статический» режим)

3. Распараллеливание алгоритмов обработки

С ростом числа ячеек в сетках обрабатываемых задач стало ясно, что без распараллеливания алгоритмов добиться высокой скорости обработки в ScientificView достаточно сложно. СуперЭВМ с распределенной памятью, как правило, обладают существенно большим ресурсом по сравнению с суперЭВМ на общей памяти, поэтому было принято решение о распараллеливании алгоритмов обработки с использованием библиотеки MPI. Использовалась схема распараллеливания на уровне данных, т.е. все процессорные ядра сервера суперЭВМ, за исключением одного выделенного, занимаются однотипной работой по обработке своей части данных. Это выделенное ядро является управляющим и занимается взаимодействием всего сервера с клиентской частью. За клиентской частью, запускаемой на ПЭВМ пользователя, остается задача отображения интерфейса и готовых растровых изображений, принятых с сервера.

Время формирования изображения в таком режиме состоит из следующих слагаемых:

- максимальное время формирования своей части итогового изображения «рабочим» процессорным ядром сервера (время рендеринга);
- время приема всех частей изображения управляющим ядром сервера;
- время сборки итогового изображения из частей;
- время передачи готового кадра с сервера на клиент.

Здесь «полезным» можно считать только время рендеринга, все остальные временные затраты являются накладными расходами, часть из которых не зависит от сложности задачи (например, время передачи с сервера на клиент). В первых параллельных версиях ScientificView время полезной работы на сложных задачах составляло около 15%.

В ходе работ по оптимизации рендер-подсистемы было реализовано 5 дополнительных алгоритмов:

1. Введена возможность подготовки и хранения нескольких уровней детализации для всех объектов. Модифицированный алгоритм A3 теперь может выбирать наиболее подходящий из них, что еще несколько ускоряет формирование изображения.
2. При передаче изображения с рабочих ядер на управляющее определяется минимальная прямоугольная пиксельная зона, в которую осуществляется вывод конкретным рабочим ядром. Только эта зона передается на управляющее ядро для дальнейшей обработки. Реализация алгоритма позволила снизить время передачи частей и сборки итогового изображения.
3. Введена иерархия при сборе итогового кадра: рабочие ядра передают свою пиксельную зону не непосредственно на управляющее ядро, а одному из небольшого числа выделенных рабочих ядер. Эти выделенные ядра проводят предсборку кадра и полученные результаты передают управляющему ядру. При большом числе ядер снижено на 40% время передачи частей и сборки итогового изображения.
4. Для более быстрой передачи кадра с сервера на клиент проводились эксперименты со сжатием информации. Лучший результат показало сжатие с использованием алгоритма jpeg. К сожалению, для высокоскоростных сетей (100 мбит и выше) существенного ускорения достигнуто не было, хотя для сетей с пропускной способностью 10 Мбит и менее получено ускорение 5 раз и выше без существенной потери качества изображения.
5. В «быстром» режиме та же задача была успешно решена путем реализации алгоритма пиксельного огрубления. Его суть заключается в том, что на сервере перед передачей кадра разрешение изображения уменьшается в 2, 4, 8 раз по каждому из направлений, а после приема его на клиенте растягивается на весь экран. Таким об-

разом, при манипуляциях с отображаемой сценой время передачи кадра с сервера на клиент за счет частичной потери качества изображения уменьшается в 4-64 раза.

4. Результаты применения алгоритмов в режиме параллельного рендеринга

Совместное применение описанных алгоритмов сегодня увеличивает полезную работу рендер-модуля ScientificView до 50%, что ускоряет формирование итогового изображения относительно базового варианта в 3-4 раза. Это позволяет даже для сложных задач удержать время формирования кадра в статическом режиме на уровне 1 сек, а при манипуляциях с отображаемой сценой обеспечить скорость 5 кадров в секунду. В то же время работа с тестами, моделирующими перспективные через 2-3 года задачи, показывает, что текущих возможностей будет недостаточно, а значит, работа в данном направлении будет продолжена.

Литература

1. Потехин А.Л., Логинов И.В., Козачек Ю.В., Никитин В.А., Кузнецов М.Ю., Деманова А.К., Попова Н.В., Фирсов С.А. ScientificView – система параллельной постобработки результатов, полученных при численном моделировании физических процессов // Сб. трудов XVIII Международной конференции по компьютерной графике и зрению «Графикон-2008». М. 2008. С. 192-198.

2. Потехин А.Л. Методы быстрого формирования изображения в параллельной сис-теме постобработки результатов научных вычислений ScientificView // Сб. трудов XX Международной конференции по компьютерной графике и зрению «Графикон-2010». СПб., 2010. С. 54-61.

3. Потехин А.Л., Никитин В.А., Логинов И.В., Кузнецов М.Г., Лопаткин А.И., Жирнов В.В., Черенков П.В., Ломтев А.В., Козачек Ю.В., Ломтев В.В. Пакет программ ЛОГОС. Новые возможности графической постобработки результатов моделирова-ния инженерных задач в параллельной системе постобработки ScientificView // Сб. трудов конференции «Супервычисления и математическое моделирование». Саров, 2012. С. 481-487.

НЕСТРУКТУРИРОВАННАЯ ПРИЗМАТИЧЕСКАЯ ДИСКРЕТИЗАЦИЯ СЛОЖНЫХ ГЕОЛОГИЧЕСКИХ СТРУКТУР В ПАРАЛЛЕЛЬНОМ РЕЖИМЕ

М.Л. Сидоров, В.А. Пронин

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров

В работе представлены описание и особенности реализации метода призматической дискретизации сложных геологических структур. Метод является двухэтапным.

На первом этапе производится генерация двумерной неструктурированной сетки в параллельном режиме на основе предложенного авторами фронтального метода сфер [1], который синтезирован из методов упаковки сфер [2], откусывания сферами [3] и подвижного фронта [4]. Двумерный генератор предназначен для создания планарных сеток с числом точек более миллиона. В частности, в параллельном режиме была получена планарная сетка с числом точек более 10 миллионов с использованием 289 процессоров. Время генерации сетки порядка 19 секунд.

После генерации двумерной неструктурированной сетки выполняется второй этап: построение трехмерной неструктурированной призматической сетки в параллельном режиме на основе обобщенного метода «угловой точки» с возможностью адаптации сетки под различные типы объектов: скважины сложной траектории, геологические разломы, пласты и т.п. Объемная сетка может иметь различное число ячеек вдоль оси аппликат в отличие от стандартного метода «угловой точки».

Целью работы является разработка сеточного генератора, покрывающего и превосходящего возможности зарубежных коммерческих аналогов для обеспечения конкурентоспособности отечественного программного комплекса НИМФА [5].

Максимальная сетка, построенная разработанным генератором, содержала 1.1 миллиарда ячеек и была построена с использованием 21600 процессоров за 4 минуты.

Литература

1. Сидоров М.Л., Пронин В.А. Параллельный генератор двумерной неструктурированной сетки на основе фронтального метода сфер // XIV Международный семинар «Супервычисления и математическое моделирование». Саров, 5-10 октября 2012.
2. Shimada K., Yamakawa A., Itoh T. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles // In 6th International Meshing roundtable. P. 375-390, 1997.
3. Li X.Y., Teng S.H., Ungor A. Bitting spheres in 3d. Submitted to the 8th International Meshing Roundtable.
4. Ito Y., Shih A.M., Soni, B.K. Reliable Isotropic Tetrahedral Mesh Generation based on an Advancing front method // Proceedings of the 13th International meshing roundtable, Williamsburg, VA. P. 95-105, 2004.
5. Доработка, верификация и адаптация программного комплекса НИМФА с целью его широкого внедрения в практику гидрогеологических расчетов по оценке воздействия ЯРОО Госкорпорации Росатом на подземные, поверхностные воды и грунты / Разведка и охрана недр. Власов С.Е. (Госкорпорация Росатом), Шагалиев

Р.М., Дерюгин Ю.Н., Горев И.В. (РФЯЦ-ВНИИЭФ), Глинский М.Л., Куваев А.А., (ФГУГП «Гидроспецгеология»).

РАЗРАБОТКА НОВОГО РЕШАТЕЛЯ РАЗРЕЖЕННЫХ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

С.А. Лебедев, Е.А. Козинов

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача решения систем линейных уравнений с симметричной разреженной положительно определенной матрицей методом Холецкого. Описывается новая последовательная реализация численной фазы разложения Холецкого, построенная на основе мультифронтального метода. Приводятся результаты вычислительных экспериментов, показывающие сопоставимость выполненной реализации с рядом известных библиотек. Формулируются планы по распараллеливанию для систем с общей памятью.

Введение

Решение разреженных систем линейных уравнений лежит в основе многих вычислительных задач компьютерной алгебры и моделирования физических процессов. Типичный пример – решение уравнений в частных производных методами конечных разностей или конечных элементов.

На сегодняшний день в мире разработано большое количество специализированного программного обеспечения для решения больших разреженных СЛАУ – так называемые «решатели» СЛАУ. В данной работе идет речь о разработке прямого решателя разреженных СЛАУ. Среди известных прямых решателей – MKL PARDISO, SuperLU, MUMPS, CHOLMOD и многие другие. Постоянно обновляемый обзор прямых решателей от авторов SuperLU можно найти по следующей ссылке: http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf.

В работе приведены текущие результаты, дано их сравнение с результатами некоторых известных библиотек, а также определены пути дальнейшего развития.

Постановка задачи

Пусть дана система линейных уравнений:

$$Ax = b. \tag{1}$$

Здесь A – разреженная симметричная положительно определенная матрица, b – плотный вектор, x – вектор неизвестных. Необходимо найти решение системы x.

Метод решения

Прямые методы решения задачи (1), как правило, основаны на применении разложения Холецкого к матрице A в виде:

$$A = U^T U, \tag{2}$$

где U – верхнетреугольная матрица. В этом случае решение системы сводится к последовательному решению двух треугольных систем:

$$U^T y = b, \tag{3}$$

$$Ux = y. \tag{4}$$

Особенностью процедуры разложения Холецкого для разреженной матрицы является то, что матрица обычно претерпевает заполнение, что на практике может привести

к неудовлетворительным требованиям по памяти. Степень заполненности матрицы можно уменьшить с помощью переупорядочивания ее строк и столбцов. Это соответствует нахождению матрицы перестановки P и переходу к эквивалентной системе (6):

$$\bar{A} = PAP^T, \tag{5}$$

$$\bar{A}(Px) = PB. \tag{6}$$

Таким образом, при решении разреженной системы с использованием метода Холецкого можно выделить следующие этапы:

1. Переупорядочивание – вычисление матрицы перестановки P и переход к системе (6);
2. Символическое разложение – построение портрета матрицы U, выделение памяти для хранения ненулевых элементов;
3. Численное разложение – вычисление значений матрицы U и размещение их в выделенной памяти;
4. Обратный ход – решение треугольных систем – уравнений (3), (4).

В данной работе идет речь о разработке численной фазы разложения Холецкого.

Во время численной фазы выполняется нахождение значений элементов верхнего треугольника. Это самая трудоемкая часть факторизации, т.к. для вычисления k-й строки фактора нужно знать значения в строках, содержащих ненулевой элемент в k-м столбце.

Мультифронтальный метод

Существует несколько методов численного разложения, и наибольшее распространение среди них получили ориентированный влево (leftlooking), ориентированный вправо (rightlooking) и мультифронтальный (multifrontal) методы [3]. В данной работе в качестве численного разложения использовался мултифронтальный метод. Проще всего мультифронтальный метод может быть описан в терминах дерева исключения.

Деревом исключения матрицы A называется дерево, множество вершин которого совпадает с множеством вершин графа матрицы (т.е. множеством строк), а множество ребер задается соотношением (7). Т.е. вершина i является потомком вершины j, если первый после диагонали ненулевой элемент в строке i расположен в столбце j:

$$e_{i,j} \in E \leftrightarrow j = \min_k\{a_{ik} \neq 0 \,\&\, i < k\}. \tag{7}$$

Основными структурами данных для мультифронтального метода являются множества матриц обновления (update matrix) и фронтальных матриц (frontal matrix).

Каждому узлу дерева ставится в соответствие матрица, которая называется фронтальной. Применяя операцию обновления 1 уровня (8), из фронтальной матрицы может быть получен соответствующий столбец фактора. Для того чтобы вычислить фронтальную матрицу, необходимо найти сумму матриц обновления всех детей узла в дереве. Матрица обновления в свою очередь получается из фронтальной матрицы после операции обновления 1 уровня.

$$A = \begin{pmatrix} d & v^t \\ v & C \end{pmatrix} = \begin{pmatrix} \sqrt{d} & 0 \\ \frac{v}{\sqrt{d}} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & C - \frac{vv^t}{d} \end{pmatrix} \begin{pmatrix} \sqrt{d} & \frac{v^t}{\sqrt{d}} \\ 0 & I \end{pmatrix}. \tag{8}$$

Более подробное описание мультифронтального метода можно найти в работе Лю [10].

Недостатком описанного подхода к разложению Холецкого является низкая производительность на матрицах больших размерностей из-за возникновения существенного количества кэш-промахов. Для решения этой проблемы применяется супернодальный (supernodal, «суперэлементный») подход. Он основан на использовании для алгоритма факторизации так называемых «супернодов» (supernode) – группы расположенных под-

ряд столбцов, имеющих одинаковую структуру заполненности ниже плотного треугольного блока. Использование супернодов позволяет производить факторизацию поблочно с применением оптимизированных матричных операций BLAS третьего уровня для плотных матриц. Подобный подход используется в MKL PARDISO, SuperLU, CHOLMOD и др.

Программная реализация

На основе описанной мультифронтальной схемы выполнена последовательная программная реализация решателя СЛАУ. Программная реализация выполнена на языке C. Для хранения матрицы разреженных матриц выбран формат CSR. Вычисления проводились в двойной точности.

При реализации численной части использовался супернодальный мультифронтальный метод с некоторыми модификациями [10], которые позволяют повысить производительность. Выделение супернодов выполняется после символической части на основе алгоритмов, приведенных в [1]. Реализация алгоритмов основана на работах [4, 6]. Для выполнения операции с плотными матрицами использовались функции из библиотеки Intel MKL [7].

Результаты экспериментов

Для анализа производительности программного комплекса был проведен ряд экспериментов на матрицах из коллекции [13] университета Флориды. В таблице 1 приведены характеристики выбранных матриц. Все они являются симметричными положительно определенными.

Таблица 1. Характеристики тестовых матриц

Матрица	Порядок	Число ненулевых элементов	Заполненность, %
pwtk	217 918	5 926 171	0,0125
msdoor	415 863	10 328 399	0,0060
parabolic_fem	525 825	2 100 225	0,0008
tmt_sym	726 713	2 903 837	0,0005
G3_circuit	1 585 478	4 623 152	0,0002
ecology2	999 999	2 997 995	0,0003

Параметры тестовой инфраструктуры приведены в таблице 2.

Таблица 2. Параметры тестового окружения

Процессор	2 четырехъядерных процессора Intel® Xeon E5520 (2.27 GHz)
Память	16 Gb
Операционная система	Windows
Среда разработки	Microsoft Visual Studio 2008
Компилятор	Intel® Parallel Studio XE 2013

В таблице 3 приведены результаты запусков последовательной версии решателя на тестовых матрицах, выделено время работы каждого этапа.

Таблица 3. Время работы последовательной реализации (в секундах)

Матрица	Число ненулевых элементов в факторе	Символическая часть	Численная часть
pwtk	49 426 260	1,06	5,39
msdoor	54 620 729	1,45	5,29
parabolic_fem	25 554 689	0,71	2,91
tmt_sym	30 095 144	0,91	3,81
G3_circuit	104 315 886	2,72	18,25
ecology2	36 283 143	1,13	5,07

Также был проведен ряд экспериментов на тех же тестовых матрицах с использованием следующих библиотек:

- MKL PARDISO: Intel® Math Kernel Library (в составе Intel® Parallel Studio XE 2013).
- MUMPS Version 4.10.0 [11].

Результаты экспериментов приведены в таблице 4. Для всех решателей использовались перестановки, полученные при использовании разработанного в ННГУ переупорядочивателя [14, 15].

MKL PARDISO при выполнении разложения Холецкого использует супернодальный метод, ориентированный влево, а MUMPS – супернодальный мультифронтальный метод.

Таблица 4. Сравнение работы численной фазы на тестовых матрицах

Матрица	Решатель авторов	MUMPS	MKL
pwtk	7,57	6,23	7,42
msdoor	8,28	6,91	8,21
parabolic_fem	4,42	4,93	4,34
tmt_sym	5,68	6,76	5,67
ecology2	7,35	9,22	7,01
G3_circuit	23,15	23,58	21,56

Как видно из таблицы 4, последовательная версия решателя показывает результаты, сравнимые с MUMPS и MKL. Т.к. наиболее трудоемким этапом вычислений является численная факторизация, время работы всего решателя в значительной мере определяется полученным числом ненулевых элементов в факторе и временем работы численной фазы, которое зависит от качества работы метода перестановки. Реализация авторов работает незначительно быстрее реализации из библиотеки MUMPS на 3 из 6 матриц (parabolic_fem, tmt_sym, ecology2), при этом отстает от Intel MKL в среднем всего лишь на 4%.

Заключение

Основным результатом работы является программная реализация мультифронтального метода. Реализованный алгоритм позволяет выполнять численное разложение разреженных положительно определенных матриц более эффективно по сравнению с предыдущим [14] и показывает сравнительное время работы с рядом широко распространенных прямых решателей.

Основным направлением дальнейшего развития является распараллеливание представленного алгоритма на системах с общей памятью (в том числе и ускорители Intel Xeon Phi), т.к. именно численная часть разложения занимает большую часть времени и требует значительных ресурсов памяти.

Вопрос об эффективной реализации параллельного алгоритма не решен до конца и представляет большой практический интерес [12, 2, 9]. Использование только высокопроизводительных библиотек, таких как BLAS, не дает приемлемых результатов из-за непоследовательного обращения к памяти и как следствие возникновения большого числа кэш-промахов, а также в связи с малыми размерами матриц, обрабатываемых функциями BLAS. Поэтому основной идеей распараллеливания является использование дерева исключения, при этом каждый узел дерева представляет собой отдельную вычислительную задачу, а узлы, не имеющие общих потомков, могут обрабатываться параллельно. При таком подходе можно рассматривать две формы балансировки – статическую и динамическую.

Одним из первых алгоритмов, использующих статическую балансировку, был алгоритм Гейста и Нг [5]. Основной идеей этого алгоритма является выделение некоторого уровня в дереве, т.е. множества вершин, таких, что для всех поддеревьев в корне с вершиной из этого множества объем вычислений примерно одинаков. В настоящее время алгоритмы, использующие статическую балансировку, так или иначе основаны на алгоритме Гейста–Нг.

При динамической балансировке используется схема мастер–рабочий, в которой мастер назначает узлы дерева рабочим в соответствие с их нагрузкой. Однако при таком простом подходе появляется большое число обращений потока в память другого потока, поэтому, как и при статической балансировке, потоку предпочтительнее назначать некоторое поддерево дерева исключения. Поиск наилучших структур данных и оптимального способа динамической балансировки продолжается до сих пор [8].

Исследование возможности создания эффективных реализаций рассмотренных алгоритмов для распараллеливания численной фазы на общую память является направлением дальнейшей работы.

Работа выполнена в лаборатории ННГУ-Intel «Информационные технологии». Авторы благодарят И.Б. Меерова, А.Ю. Пирову, А.В. Сысоева за полезные обсуждения и внимание к работе.

Литература

1. Ashcraft C., Grimes R. The influence of relaxed supernode partitions on the multifrontal method // ACM Trans. Math. Software. – 1989. – Vol. 15, No. 4. – P. 291–309.
2. Chowdhury I., L'Excellent J.-Y. Some experiments and issues to exploit multicore parallelism in a distributed-memory parallel sparse direct solver. Research report RR-4711, INRIA and LIP-ENS Lyon, Oct. 2010.
3. Davis T.A. Direct methods for sparse linear systems. – Philadelphia: SIAM, 2006. – 217 p.
4. Demmel J.W., Eisenstat S.C., Gilbert J.R., Li X.S., Liu J.W.H. A supernodal approach to sparse partial pivoting // SIAM J. Matrix Anal. Appl. – 1999. – Vol. 20, No. 3. – P. 720–755.
5. Geist A., Ng. E.G. Task scheduling for parallel sparse Cholesky factorization // Int J. Parallel Programming, 18:291(314). 1989.
6. Hogg J.D. Efficient sparse Cholesky factorization – URL.: http://www.maths.ed.ac.uk/~s0455378/EfficientCholesky.pdf.
7. Intel Math Kernel Library Reference Manual. URL: [http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].
8. L'Excellent J.-Y. Multifrontal method: Parallelism, Memory Usage and Numerical Aspects. Charge de recherché, Inria, Sept. 2012.
9. L'Excellent J.-Y., Sid-Lakhdar M. Introduction of shared-memory parallelism in a distributed-memory multifrontal solver. Research report №8227, Project-Team ROMA, Feb. 2013.
10. Liu J.W.H. The multifrontal method for sparse matrix solution: Theory and practice // SIAM Review. – 1992. Vol. 34, No. 1. – P. 82–109.
11. 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]
12. Posey S. GPU progress in sparse matrix solvers for applications in computational mechanics // European seminar on computing, Pilzen, Chech Republic, June, 2012.
13. The University of Florida Sparse Matrix Collection – URL.: www.cise.ufl.edu/research/sparse/matrices/.

14. Козинов Е.А., Лебедев И.Г., Лебедев С.А., Малова А.Ю., Мееров И.Б., Сысоев А.В., Филиппенко С.С. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей // Вестник Нижегородского университета им. Н.И. Лобачевского. – Н. Новгород: Изд-во ННГУ, 2012. – №5(2) – С. 376-384.
15. Пирова А.Ю. Разработка нового переупорядочивателя симметричных разреженных матриц // Материалы XIII всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах». Н.Новгород: Изд-во ННГУ, 2013. В печати.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ АВТОРОВ

СОДЕРЖАНИЕ

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

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ

Материалы XIII Всероссийской конференции
(Нижний Новгород, 14–16 ноября 2013 г.)

Под редакцией проф. В.П. Гергеля

Ответственный за выпуск К.А. Баркалов

Формат 60×84 1/8. Уч.-изд. л. 18,2. Тир. 150 экз.
Издательство Нижегородского госуниверситета им. Н.И. Лобачевского.
603950, Н. Новгород, пр. Гагарина, 23.