Министерство образования и науки Российской Федерации

Суперкомпьютерный консорциум университетов России
Министерство образования и науки Пермского края
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Пермский национальный исследовательский
политехнический университет»

Высокопроизводительные параллельные вычисления на кластерных системах (HPC 2014)

Материалы
XIV Международной конференции

(г. Пермь, 10–12 ноября 2014 г.)

Издательство
Пермского национального исследовательского
политехнического университета
2014

УДК 004.382.2:51
ББК 32.973
В93

Сборник XIV Международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах», состоявшейся в Пермском национальном исследовательском политехническом университете 10–12 ноября 2014 г., содержит материалы докладов, посвященных высокопроизводительным параллельным вычислениям на кластерных системах применительно к науке, образованию и промышленности.

Редакционная коллегия:
В.П. Гергель – д-р техн. наук, профессор;
Н.А. Шевелев – д-р техн. наук, профессор (гл. редактор);
В.Я. Модорский – д-р техн. наук, профессор;
К.А. Баркалов – канд. физ.-мат. наук, ст. преподаватель.

Финансовая поддержка конференции

Российский фонд фундаментальных исследований
Министерство образования и науки Пермского края
Корпорация INTEL
Компания NVIDIA
Компания «НИАГАРА КОМПЬЮТЕРС»
Компания «ПЛМ-УРАЛ»
Компания «ТЕСИС»
Компания «ИВС-СЕТИ»
Компания «РСК»

Информационная поддержка конференции

Газета научного сообщества «Поиск»
Журнал «Суперкомпьютеры»
Информационно-аналитический центр Parallel.Ru

ISBN 978-5-398-01312-2
© ПНИПУ, 2014

10–12 ноября 2014 г. под эгидой Суперкомпьютерного консорциума университетов России на базе Пермского национального исследовательского политехнического университета при непосредственном участии Нижегородского государственного университета им. Н.И. Лобачевского и поддержке Российского фонда фундаментальных исследований, Министерства образования и науки Пермского края, корпораций Intel, NVIDIA, компаний «РСК», «Ниагара Компьютерс», «ИВС-СЕТИ», «ПЛМ-Урал», «Тесис», газеты научного сообщества «Поиск», журнала «Суперкомпьютеры», информационно-аналитического центра Parallel.Ru проведена XIV Международная конференция «Высокопроизводительные параллельные вычисления на кластерных системах».

Работа конференции была организована по следующим направлениям:

– 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;

– решение прикладных задач на кластерных системах.

В рамках конференции подробно рассмотрены вопросы организации высокопроизводительных вычислительных систем, научных исследований с использованием параллельных вычислений, основного и дополнительного образования разработчиков и пользователей суперЭВМ.

Проведение конференции поддержано грантом РФФИ № 14-07-20403-г.

Н.В. Аболмазова, А.Н. Коварцев

Самарский государственный аэрокосмический университет им. акад. С.П. Королева
(Национальный исследовательский университет)

К ВОПРОСУ О ВОЗМОЖНОСТИ ОЦЕНКИ НАДЕЖНОСТИ СЛОЖНЫХ ПРОГРАММНЫХ ПРОДУКТОВ

Оценивание надежности программных продуктов коренным образом отличается от методов исследования надежности технических систем. Непосредственный перенос классических методов теории надежности применительно к программным системам, как правило, не дает удовлетворительных результатов, что связано с «природой» исследуемых объектов.

Введение

Понятие отказа программы субъективно: одна и та же программа может устроить одного пользователя и не устроить другого. Более того, программные ошибки имеют разную степень воздействия на конечный результат: одни приводят к катастрофе, другие выводят на экран дисплея тексты с орфографическими ошибками. Понятие надежности применительно к программным средствам (ПС) является «сослагательным», и, говоря об их надежности, мы должны упоминать, какие типы ошибок имеются в виду. Понятие «надежность элементов технических устройств (ТУ)» обычно не связывают с типами отказов.

Основной постулат теории надежности ТУ – зависимость количества отказов элемента от времени эксплуатации – абсолютно бессмыленен применительно к понятию надежности программ.

В ПС ошибки возникают при определенных сочетаниях исходных данных. Бесчисленные запуски программы при одном или ограниченном количестве наборов исходных данных, где она работоспособна, в течение любого интервала времени не породят ни одного отказа. Более того, надежность программных модулей со временем может только увеличиваться за счет устранения выявляемых ошибок. Исходя из этого теоретически может возникнуть ситуация, когда при тестировании программного модуля ошибки уже не будут обнаруживаться.

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

Постановка задачи

Наиболее общее определение *надежности* программных продуктов предлагает Б. Мейер [1]. Он определяет надежность как *способность про-*

граммы давать разумные результаты во всех возможных окружениях, и в частности в аномальных условиях.

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

Однако количество сочетаний исходных данных программных модулей (в общем случае) настолько велико, что перебрать все сочетания для современной ЭВМ практически невозможно. Таким образом, проблему оценки надежности ПС можно сформулировать как комбинаторную проблему поиска алгоритмов частичного перебора пространства исходных данных модуля.

Для сравнительно простых программных модулей, имеющих несколько входных параметров, в настоящее время получены положительные результаты. Для некоторых модулей удается доказать их корректность [1]. В других случаях q можно определить экспериментально [2, 3].

Вместо дискретной меры – мощности – введем непрерывную меру, оценивающую объем многомерного тела. Тогда ненадежность модуля можно оценить по формуле $q = V(\Omega_E) / V(\Omega_{\text{In}})$.

Для современных сложных программных средств число исходных данных измеряется сотнями и тысячами переменных, поэтому прямые методы тестирования становятся неэффективными и практически нереализуемыми. Кажется естественным оценивать надежность сложных ПС, используя известные характеристики составляющих его компонентов так же, как это делается для технических систем. Однако надежность ПС не может быть вычислена непосредственно исходя из надежностей составляющих ее элементов (программных модулей).

Метод оценки надежности сложных программных продуктов

Предположим, что нам известны характеристики надежности всех модулей, из которых составлено ПС. Пусть q_i – ненадежность *i*-го модуля $A_i : X_i^{\text{In}} \to Y_i^{\text{Out}}$, вычисленная с учетом всевозможных применений модуля, т.е. на достаточно широкой области значений исходных данных Ω_{In}^i: $X_i^{\text{In}} = (x_{1i}^{\text{In}}, x_{2i}^{\text{In}}, ..., x_{n_i i}^{\text{In}}) \in \Omega_{\text{In}}^i$. Допустим, что логические ошибки в организации логики функционирования ПС отсутствуют (данная проблема требует отдельного рассмотрения [1]).

Пусть ПС имеет L маршрутов ее исполнения [4]. Каждый из маршрутов обозначим $M_j = i_{j_0} i_{j_1} ... i_{j_k}$, где i_{j_k} – номер программного модуля для ПС. 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 ПС можно оценить величиной $Q_{n,,} = \sum_{j=1}^{L} r_j Q_{M_j}$.

Представленная формула справедлива, если оценки ненадежности программных модулей неизменны при различных использованиях этих модулей, но это не так.

При выполнении программы за счет функциональных преобразований могут изменяться области изменения исходных данных комплектующих ее модулей.

Это способно увеличить характеристику ненадежности программных модулей за счет уменьшения знаменателя формулы $q_i = V(\Omega_E^i) / V(\Omega_{\text{In}}^i)$. Но еще большую проблему представляет изменение законов распределения исходных данных модулей за счет соответствующих функциональных преобразований первоначально равномерного распределения исходных данных ПС.

Появляется необходимость использования вероятностных мер при оценке ненадежности программных модулей.

Пусть некоторый модуль, имеющий единственный входной параметр x, «прошел» серию испытаний (из N испытаний), в которых не было обнаружено ни одной ошибки, причем $x \in [a,b]$. Пусть теперь при использовании модуля в ПС изменился диапазон значений входного параметра модуля $x \in [c,d] \subset [a,b]$ без изменения его закона распределения. 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 $\tilde{q}(x) = P\{\Delta \in [0,d]\} \int_x^{x+\Delta} K\lambda\, e^{-\lambda x} dx =$

$$= \frac{d}{b} \frac{e^{-\lambda x}}{1 - e^{-\lambda d}} (1 - e^{-\lambda \Delta}).$$

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

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

$$= \frac{d}{b}\frac{(1-e^{-\lambda\Delta})K}{\lambda d}(1-e^{-\lambda d}) = \frac{(1-e^{-\lambda\Delta})}{\lambda b}.$$

Как показывают расчеты [5], существенные изменения значений λ (степени неравномерности распределения) практически не сказываются на величине $q_{\text{ср}}$. Более того, $q_{\text{ср}} \approx q$. Однако данный результат необходимо еще обобщить на многомерный случай.

Список литературы

1. Мейер Б., Бодуэн К. Методы программирования: в 2 т. – М.: Мир, 1982. – Т. 2.

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

3. Коварцев А.Н. Автоматизация тестирования вычислительных модулей // Надежность и качество 2001: сб. докл. междунар. симпоз. – Пенза, 2001. – С. 285–288.

4. Липаев В.В. Надежность программных средств. – М.: СИНТЕГ, 1998. – 232 с.

5. Аболмазова Н.В., Коварцев А.Н. Проблема оценки надежности программных комплексов // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XIII Всерос. конф. / под ред. проф. В.П. Гергеля. – Н. Новгород: Изд-во Нижегор. гос. ун-та, 2013. – С. 4–8.

Н.Г. Аксак, А.Н. Феденко

Харьковский национальный университет радиоэлектроники

ПАРАЛЛЕЛИЗМ В СТАТИСТИЧЕСКИХ СИСТЕМАХ МАШИННОГО ПЕРЕВОДА

Рассматривается повышение быстродействия и масштабируемости нейроалгоритмов машинного перевода за счет их параллельной реализации на высокопроизводительных системах путем использования параллелизма данных.

Предлагаемая модель оценивания ускорения обучения и функционирования многослойной нейронной сети, совокупно учитывающая вре-

менные и аппаратные характеристики, позволяет повысить быстродействие параллельных нейровычислений при решении задач автоматического машинного перевода на высокопроизводительных системах с обменом сообщениями.

Введение

В настоящее время машинный перевод остается одной из проблем обработки естественного языка. Объем предназначенной для перевода информации увеличивается каждую секунду, растет спрос на быстрые и высококачественные системы машинного перевода в режиме реального времени. Так, например, в существующих системах процесс перевода 50 слов занимает в среднем 1 минуту. Это обусловлено сложными алгоритмами, заложенными в процессе перевода, а также наличием большого количества словарей в статистических моделях.

Повышение быстродействия и масштабируемости возможно за счет параллельной реализации алгоритмов машинного перевода на высокопроизводительных системах путем использования параллелизма данных.

Обзор статистических систем машинного перевода

Задачей машинного перевода является преобразование исходного текста с одного языка в наиболее вероятный по содержанию текст на другом языке.

Существует два принципиально разных подхода к построению алгоритмов машинного перевода: основанный на правилах и статистический.

Системы машинного перевода, основанные на правилах, предполагают использование лингвистической информации об исходном и выходном языках, в то время как статистическая технология перевода основана на статистическом вычислении вероятности совпадений. Для работы этой системы необходимы огромные базы параллельных текстов, где попарно хранятся словосочетания (фразы из 2–3 слов) и их переводы, так называемые N-граммы. В процессе перевода также используется механизм анализа, но не лингвистический, а статистический. Система подбирает вариант перевода, основываясь на частоте совпадений, т.е. в конечном итоге будет подставлен вариант, имеющий наиболее высокий процент совпадений.

Для перевода текста используют модель языка и модель перевода. Модель языка дает оценку вероятности фразам выходного языка, а модель перевода оценивает вероятность исходной фразы при условии фразы на выходном языке. В системах статистического перевода используются варианты N-граммной модели языка. Согласно этой модели правильность выбора того или иного слова зависит только от предшествующих (N–1)-слов.

Специфический алгоритм CKY [1, 2], или алгоритм Кока – Янгера – Касами, применяется на этапе структурного анализа в процессе машинного перевода. В работе [3] представлен трехступенчатый CKY-алгоритм. В качестве исходного текста выбрано предложение на английском языке: Method improves efficiency. Процесс перевода показан в виде треугольной таблицы.

С учетом предварительной сегментации исходного предложения при первом шаге осуществляется поиск выходных фраз в модели перевода (рис. 1). Нижние ячейки таблицы постепенно заполняются полученными переводами и связываются вероятностями в зависимости от модели перевода. Например, ячейка, соответствующая сегменту Method improves исходного предложения, может иметь запись «Метод повышает: 0,1667 %» или «Способ улучшает: 0,05 %» в зависимости от размера модели перевода. Ячейки 1 и 2 хранят переводы для сегментов Method improves efficiency и improves efficiency соответственно.

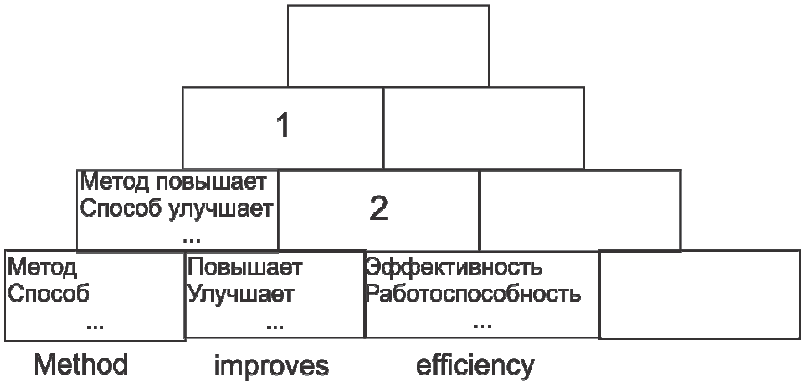


Рис. 1. Шаг 1: поиск выходных фраз

Второй шаг (рис. 2) предполагает объединение полученных на предыдущем шаге переводов и постепенное заполнение таблицы с помощью модели языка в направлении снизу вверх. Для получения перевода полной фразы Method improves efficiency необходимо поочередно сочетать записи различных ячеек. Для каждой полученной комбинации рассчитывается вероятность посредством N-граммной модели языка.

После полного заполнения таблицы из ее вершины выбирается наиболее вероятный перевод целого предложения (рис. 3). Затем он сравнивается с различными комбинациями по всем ячейкам таблицы в направлении сверху вниз. Процесс продолжается, пока не будет найдена пара с одинаковыми вероятностями.

Недостатком данного алгоритма являются большие затраты времени, необходимые для выполнения второго шага. Решить эту задачу возможно с применением стратегии распараллеливания.

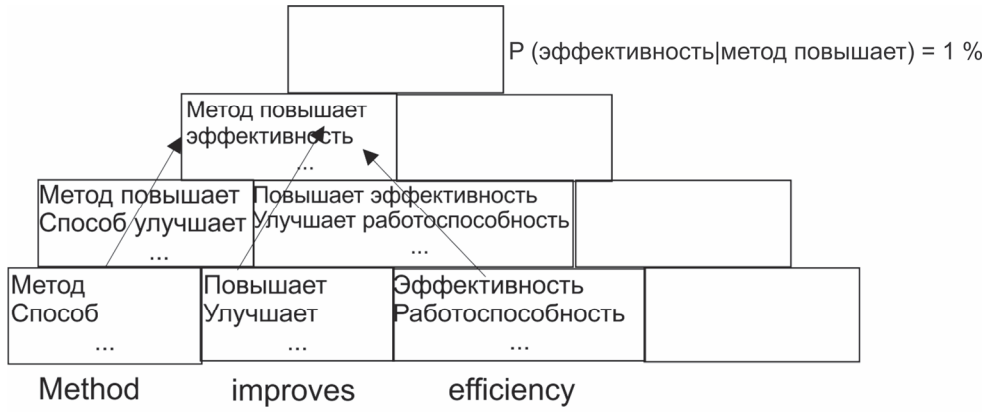


Рис. 2. Шаг 2: объединение выходных фраз

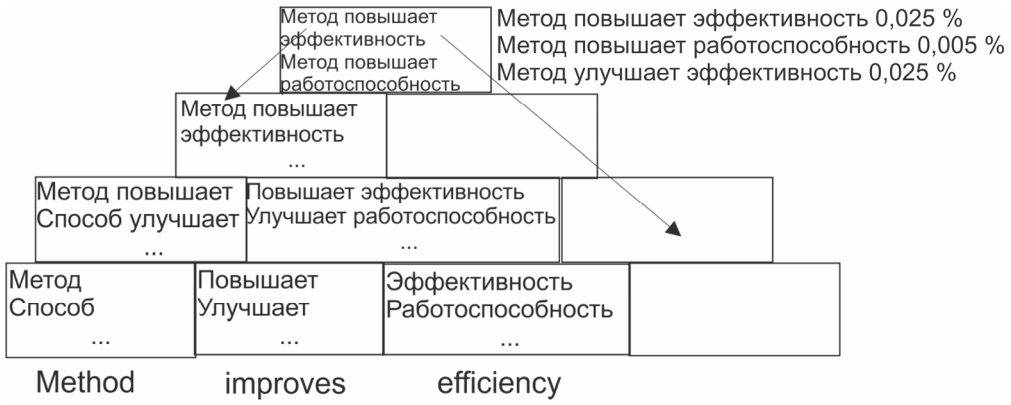


Рис. 3. Шаг 3: выделение наиболее
вероятного перевода целого предложения

При вычислении таблицы каждая комбинация, порождаемая на втором шаге, является независимой и может иметь множество сдвигов вправо и влево внутри себя. Это позволило использовать SIMD-архитектуры, предназначенные для реализации высокоскоростных систем обработки информации в режиме реального времени. Для расчетов была использована технология CUDA, реализованная на 1.30GHz NVIDIA GTX280 процессоре, а также на 1.40GHz NVIDIA GTX480 Fermi, и технология OpenMP с 4 потоками, реализованная на 2.67GHz Intel Nehalem CPU. Полученное ускорение при автоматическом переводе зависело от длины предложений, большее ускорение получено при использовании технологии OpenMP.

Для улучшения качества перевода можно использовать разные теории, которые могут быть положены в основу разработки программ-переводчиков. Остановимся на представлении непрерывной модели языка в виде нейронной сети (НС).

Так, например, для перевода с английского на сербскохорватский язык использована многослойная нейронная сеть [4].

Оценка непрерывного пространства нейросетевой языковой модели в статистической системе машинного перевода представлена в работе [5]. На

рис. 4 изображен многослойный персептрон, где входными данными являются слова $w_{i-n+1},...,w_{i-1}$ ($w_{i-3}, w_{i-2}, w_{i-1}$ являются предшественниками 4-граммной модели).

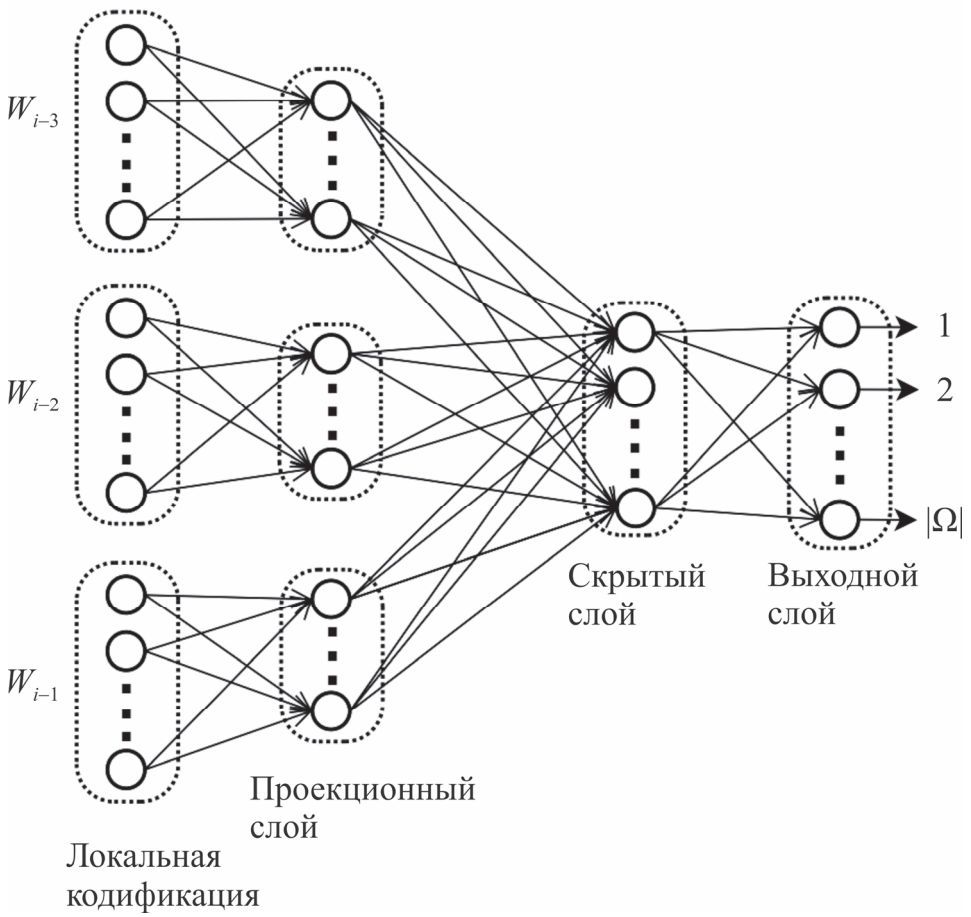


Рис. 4. Архитектура непрерывного пространства нейронной сети языковой модели

В работе [6] исследованы непрерывные модели пространства перевода, где вместо стандартных дискретных представлений вероятности перевода используется непрерывное представление единиц перевода. Для перевода большого набора данных применяется многослойная нейронная сеть (МНС) со структурированным выходным слоем.

Модель оценивания ускорения обучения и функционирования МНС

Следует отметить, что реализация процессов обучения нейронных сетей с применением современных параллельных технологий позволяет значительно ускорить автоматический перевод текстов.

Моделирование параллельной процедуры позволяет установить количественную связь между взаимосвязанными временными, аппаратными

характеристиками и получаемым ускорением МНС, благодаря чему можно выбрать более эффективные и простые в реализации решения.

В основу построения модели оценивания ускорения обучения и функционирования МНС положено отношение тактового времени выполнения последовательного обучения к распределенному, адаптированному к многопроцессорным вычислительным системам. Для упрощения изложения материала при расчете тактового времени выполнения последовательного и распределенного обучения МНС введено предположение – каждая скалярная операция выполняется за одну условную единицу времени.

МНС задается архитектурой $n_1 - n_2 - ... - n_L$, где n_1 – количество нейронов во входном слое n_l, l – количество нейронов в скрытых слоях, $l = \overline{2, L-1}$; n_L – количество нейронов в выходном слое. Обучающая выборка $\{(X(1), D(1)), (X(2), D(2)), ..., (X(K), D(K))\}$ состоит из K примеров $X(K) = \left[x^1(k), .., x^{n_1}(k) \right]^T$ и соответствующих им значений целевого признака $D(k) = \left[d^1(k), d^2(k), ..., d^{n_L}(k) \right]^T$, $k = \overline{1, K}$, тестовая выборка содержит T примеров. Для обработки обучающей и текстовой выборок требуется U эпох обучения.

Для определения ускорения S построены модели оценивания ускорения распределенного обучения L-слойной нейронной сети [7, 8]:

$$S\left(U, K, T, L, P, h, n_1, ..., n_L, t^h, t_c\right) = \\ = \frac{Q\left(U, K, T, L, n_1, ..., n_L, t^h\right)}{Q_p\left(U, K, T, L, n_1, ..., n_L, h, t^h\right) + Q_c\left(U, K, T, L, P, t_c\right)},$$

и распределенного функционирования МНС:

$$S\left(L, P, h, n_1, ..., n_L, t^h, t_c\right) = \frac{Q\left(L, n_1, ..., n_L t^h\right)}{Q_p\left(L, n_1, ..., n_L, h, t^h\right) + Q_c\left(L, P, t_c\right)}.$$

Общее количество скалярных операций Q [7], необходимых для последовательного обучения МНС, определяется в соответствии с выражением

$$Q = U \times \left(K \times \left(\sum_{m=2}^{L} \left(2n_{m-1} + 3\right) n_m + \sum_{m=L-1}^{2} 2n_{m+1} + 7 \right) n_{m-1} n_m + \right. \\ \left. + 8 n_L n_{L-1} \right) + T \times \left(\sum_{m=2}^{L} \left(2n_{m-1} + 3\right) n_m \right).$$

Для определения количества скалярных непараллельных операций в параллельной и распределенной процедурах $Q_{\text{пос}}$ используется разница между общим количеством скалярных операций Q в последовательной

процедуре и количеством независимых скалярных операций в распределенной процедуре Q_p, полученная путем увеличения количества вычислителей, стремящаяся к некоторому предельному значению, которое целесообразно использовать в расчетах. Общее количество вызовов методов передач данных осуществляется посредством Q_c между h вычислителями.

Вычисление общего количества параллельных скалярных операций Q_p при параллелизме процедуры на уровне реализации функций нейронов осуществляется посредством выделения тактов обработки независимых операций, в качестве которых представлены функции вычисления выхода нейрона, ошибки реакции сети и настройка весовых коэффициентов, а скалярные операции для вычисления перечисленных функций выполняются последовательно на отдельных вычислителях [8].

Выводы

Модель оценивания ускорения обучения и функционирования МНС совокупно учитывает временные и аппаратные характеристики, что позволяет повысить быстродействие параллельных нейровычислений при решении задач автоматического машинного перевода на высокопроизводительных системах с обменом сообщениями.

Список литературы

1. Cocke J., Schwartz J.T. Programming languages and their compilers: Preliminary notes. Technical report / Courant Institute of Mathematical Sciences. – New York University, 1970.

2. Kasami T. An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report AFCRL-65-758 / Air Force Cambridge Research Lab. – Bedford, MA, 1965.

3. Chao-Yue Lai. Parallelizing a statical machine translator / U.C. Berkeley CS267. – Spring, 2010.

4. Koncar N., Guthrie G. A Natural Language Translation Neural Network // New Methods in Language Processing / ed. D.B. Jones, H.L. Somers. – London: UCL Press, 1997. – P. 219–228.

5. Neural Network Language models for Translation with Limited Data Proceeding / M. Khalilov, José A.R. Fonollosa, F. Zamora-Martínez, M.J. Castro-Bleda, S. España-Boquera // ICTAI '08 Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence. – Vol. 2. – P. 445–451.

6. Le Hai Son, Allauzen A., Francois Yvon. Continuous Space Translation Models with Neural Networks // Human Language Technologies:

Conference of the North American Chapter of the Association for Computational Linguistics, 3–8 June. – Montreal, Canada, 2012. – P. 39–48.

7. Аксак Н.Г., Тыхун А.Ю. Анализ моделей производительности реализации нейроалгоритма // Системи обробки інформації: зб. наук. пр. – Харків: ХУПС імені Івана Кожедуба, 2008. – Вип. 1 (68). – С. 2–5.

8. Аксак Н.Г., Тыхун А.Ю., Новосельцев И.В. Модели производительности синтеза нейроалгоритма на высокопроизводительные архитектуры // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2008): материалы 8-й Междунар. конф. – Казань: Изд-во КГТУ, 2008. – С. 199–202.

С.В. Аксёнов, К.А. Костин, Н.И. Герасимова

Национальный исследовательский Томский политехнический университет

ИСПОЛЬЗОВАНИЕ ФИЛЬТРОВ ЛАВСА ДЛЯ КЛАССИФИКАЦИИ ТЕКСТУР НА GPU

Анализ графических и видеоколлекций требует быстрого алгоритма сегментации кадра или изображения для выделения в последующем регионов, значимых для исследования. Фильтры Лавса показали себя как достаточно эффективное средство для представления текстурных характеристик, сочетающее эффективность и относительно невысокую затратность по ресурсам. Алгоритм вычисления характеристик предполагает независимость вычислений для разных фрагментов входного кадра или изображения. Предложена реализация не только параллельного алгоритма сегментации, но и пример классификатора, использующего рассчитанные локальные особенности фрагментов.

Введение

Поиск изображений и видео по содержимому является одной из актуальных задач ввиду накопления огромного количества неструктурированных массивов банков данных, содержащих такие файлы. Обычным подходом при коллекционировании такой информации служит задание тегов, характеризующих содержимое. Однако достаточно часто текстовое описание не отражает в полной мере загруженный файл, и пользователю приходится вручную просматривать большое количество графических изображений или кадров видео, чтобы найти нужное. Вариантом решения этой проблемы может быть предварительный анализ изображений перед тем, как они будут сохранены в коллекции, локализующий основные объекты и классифицирующий их с запоминанием тегов, связанных с распознанными фрагментами. В работе было предложено использовать индек-

сирование графики и видео с помощью выделения текстурных характеристик и классификации графической информации с помощью интеллектуальной системы, обрабатывающих рассчитанные свойства разных фрагментов кадра видео или изображений. Целью работы является увеличение производительности поисковых систем в банках графических и видеоданных с помощью нечеткого классификатора, анализирующего текстурные характеристики на базе параллельных вычислений на GPU.

Алгоритм расчета текстурных характеристик

Существует достаточно много подходов, позволяющих сегментировать изображение по разным свойствам его фрагментов [1], однако в своей работе мы использовали статистический метод, предложенный Лавсом, позволяющий оценить распределение простейших примитивов (точки, границы, изменение градиента и т.д.) внутри каждого фрагмента. Фактически фильтры Лавса являются двумерными матрицами, свертка с которыми для разных частей обрабатываемого изображения показывает наличие в них заданных характеристик. Их особенностью является возможность их генерации для определенного размера (что также происходит с помощью свертки базовых 1D векторов размерностью 3, 5, 7). Таким образом, предполагается наличие большого числа фильтров, каждый из которых представляет одно из свойств текстур, например изменение градиента по горизонтам и рябь по вертикали или волнистость по горизонтали и локализация точек по вертикали. На рис. 1 показаны примеры вычисления нескольких характеристик с помощью разных фильтров.

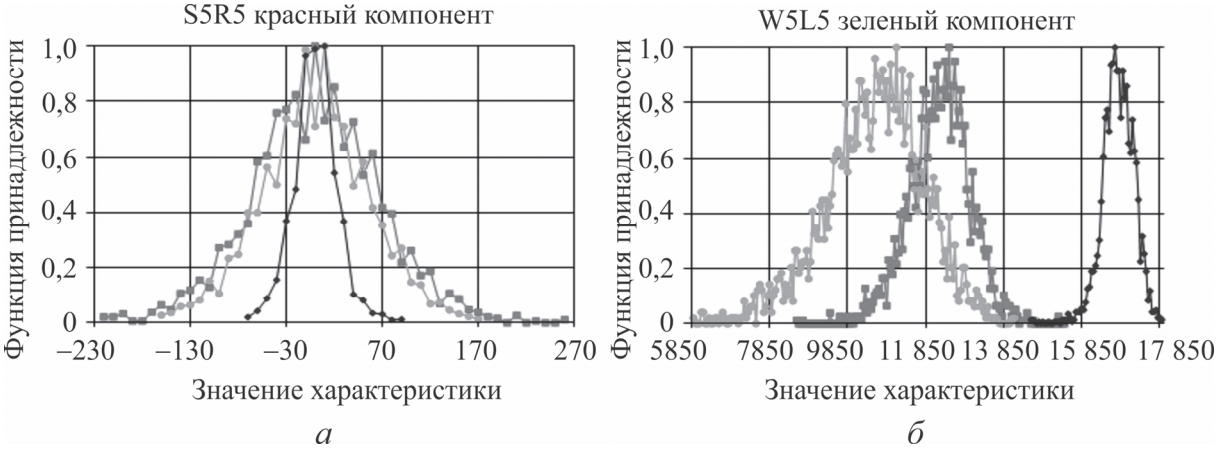
а *б* *в*

Рис. 1. Пример вычисления текстурных характристик Лавса: *а* – оригинального изображения с помощью фильтров; *б* – изменения уровня красной характеристики по вертикали и горизонтали; *в* – степени волнистости по вертикали и горизонтали зеленого компонента

Однако, как показали результаты [2], использование всего набора характеристик является излишним, так как практически для многих характеристик отфильтрованные значения свойств для разных текстур являются

очень близкими, что существенно ограничивает их практическое применение. В работе кадры видео и изображения исследовались фильтрами размером 5×5. На рис. 2 представлены графики распределения, которые в нашем подходе выражают функции принадлежности свойств, значений текстурных характеристик для двух разных текстур: первое свойство – наличие точек по горизонтали и волнистость по вертикали для красного компонента и второе свойство – волнистость по горизонтали и изменение уровня зеленого компонента по вертикали.

Использование первой характеристики как компонента классифицируемого вектора выглядит малопривлекательно, так как значения этого свойства для разных классов достаточно близки, в то же время для первой характеристики наблюдается более существенная разница для разных текстур.



а *б*

Рис. 2. Графики распределения текстурных характеристик: *а* – фильтра наличия точек по горизонтали и волнистость по вертикали красного компонента; *б* – фильтра волнистости по горизонтали и изменение уровня зеленого компонента по вертикали:

На основе многих тестов для разных изображений и кадров видео было получено, что для фильтров размеромы 5×5 наиболее существенными оказались 4 характеристики для каждого из 3 цветовых компонентов:

– изменение значения компонента по горизонтали и вертикали;

– изменение значения компонента по горизонтали и наличие ряби по вертикали;

– наличие волнистости по горизонтали и изменение характеристики по вертикали;

– волнистость по горизонтали и волнистость по вертикали.

Таким образом, было выявлено 12 характеристик, которые, по нашему мнению, для фильтра 5×5 позволяют описать классы текстур.

Классификатор текстур

В качестве классификатора было предложено использование нечеткой системы типа синглтона, в которой процедура классификации описывается с помощью нечетких правил: ЕСЛИ $x_1 = A_{1j}$ и $x_2 = A_{2j}$ и … $x_k = A_{kj}$, ТО объект принадлежит к классу *j*, где x_i – значение текстурной характеристики; A_{ij} – нечеткая область определения *i*-й переменной для класса *j*.

В нашем случае в качестве нечетких множеств, определяющих степень принадлежности к классу текстуры, будут выступать множества с усредненными функциями распределения текстурной характеристики для нескольких образцов текстур, принадлежащих одному классу.

Особенностью классификатора будет то, что существует возможность добавления новых классов текстур без нарушения уже существующих знаний, выраженных правилами. На рис. 3 представлен вид предлагаемого нечеткого классификатора.

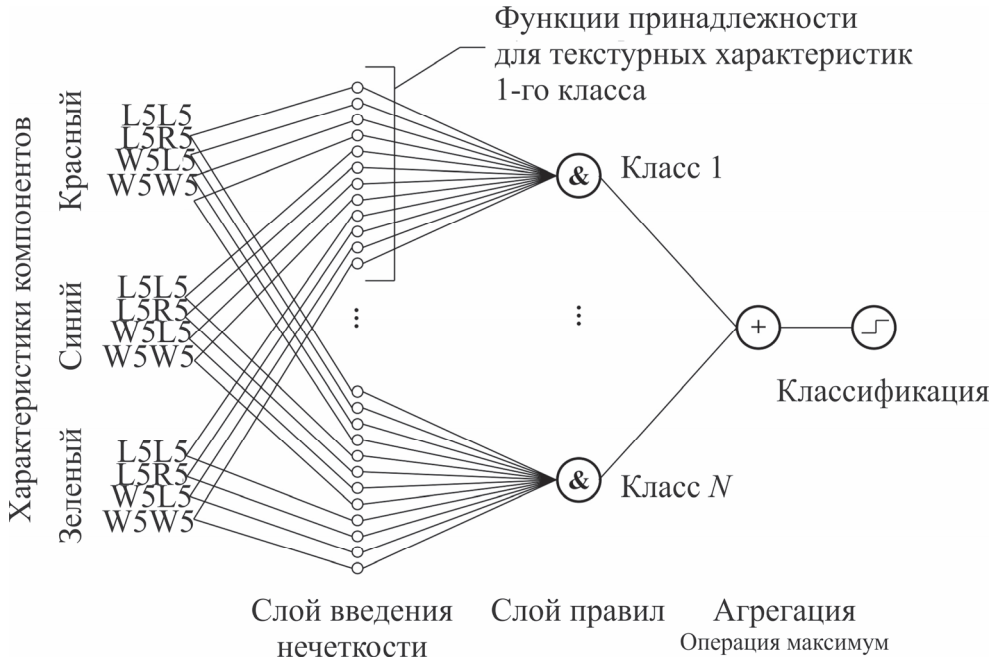


Рис. 3. Предлагаемый классификатор, основанный
на анализе текстурных харктеристик Лавса

Логический вывод выполняется на уровне правил с помощью операции МИНИМУМ для значений истинностей предпосылок. Для агрегации результатов правил используется операция МАКСИМУМ, которая находит правило с наибольшей степенью уверенности. На заключительном этапе эта степень сравнивается с пороговым значением, и при превышении над последним делается вывод о классификации фрагмента.

Достоинствами предлагаемой модели являются: независимость вычислений для разных фрагментов изображения или кадра, отсутствие рекурсий и прямой ход вычислений, что крайне эффективно для многопоточных вычислений на GPU.

Параллельная реализация классификатора с помощью GPU и ее результаты

Алгоритм классификации текстур на GPU представляется следующими шагами. В разделяемой памяти GPU хранятся матрицы L5L5, L5R5, W5L5, W5W5. При вызове ядра передаются в глобальную память цветовые оттенки входного кадра и параметры нечеткой системы: функции принадлежности антецедентов, выраженные в виде отнормализованных гистограмм. Ниже представлен синтаксис ядра CUDA, значения аргументов для функции даны в таблице.

__global__ void GPUFuzzyClassifier(const int *colorMatrix, int imageWidth, const float *histograms, const int *borders, const float *histogramStartValue, const float *widthOfBar, const int *rules, int *classID)

Аргументы ядра

№ п/п	Название переменной	Назначение
1	colorMatrix	Вектор цветовых оттенков, хранящий значения красных, зеленых и синих компонентов пикселей для всех пикселей обрабатываемого изображения
2	imageWidth	Ширина изображения, необходима, так как цветовые оттенки представлены в виде 1D массива
3	histograms	Нормализованные гистограммы существенных характеристик, использующиеся в антецедентах классификатора
4	borders	Индекс начала гистограммы функции принадлежности антецедента, необходим, так как все гистограммы передаются в одном массиве
5	histogramStartValue	Значение границы крайнего левого столбца гистограммы характеристики
6	widthOfBar	Ширина столбцов гистограмм, необходима из-за разного разброса
7	rules	Индексы функций принадлежности, которые формируют правило для оценки каждого класса текстуры
8	classID	Результат обработки: класс текстуры для каждой области 5×5 изображения

Алгоритм функционирования ядра состоит из следующих шагов:

Шаг. 1. Расчет значений текстурных характеристик L5L5, L5R5, W5L5, W5W5 для красного, зеленого и синего цветовых оттенков окрестности 5×5 пикселя, соотвутствующего потоку.

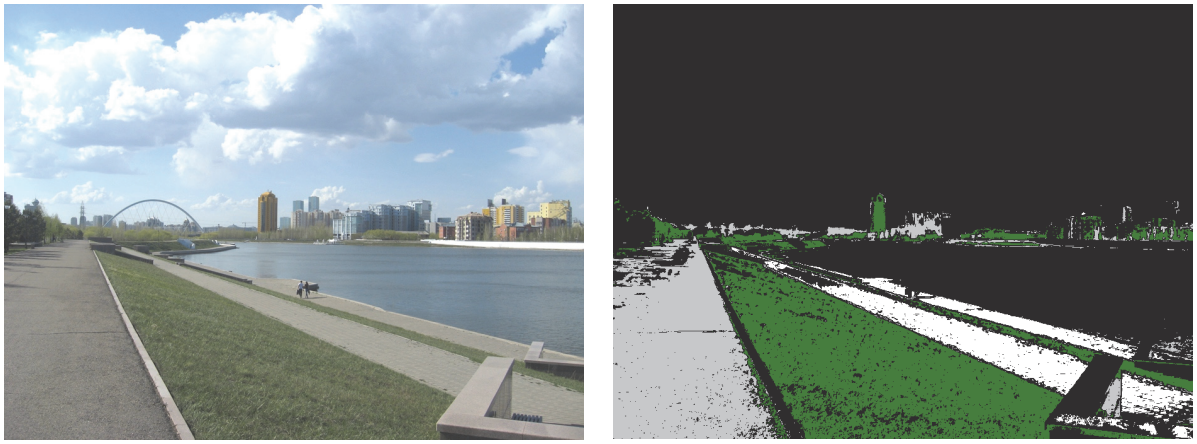
Шаг. 2. Активация нечеткого классификатора. Фаззификация. Для каждого из 12 значений характеристик определяется столбец гистограммы, в которую попала характеристика для каждого правила.

Шаг. 3. Расчет степени уверенности для каждого правила.

Шаг. 4. Нахождение правила с самой высокой степенью уверенности и сравнение с пороговым значением.

Шаг. 5. Если пороговое значение меньше, то результат классификации – тип, ассоциированный с правилом с максимальной степенью уверенности, в противном случае база текстур не содержит образцов, похожих на исследуемый тип окрестности.

Тестирование реализованного алгоритма проводилось на GPU NVidia GeForce 8600 GT, в состав которой входит 4 мультипроцессора и возможно сгенерировать сетку из 65 535×65 535 блоков, причем последние могут включать до 512 нитей. На рис. 4 показан пример классификации текстур для изображения. Обнаруженные области текстур травы, асфальта и тротуарной плитки окрашены на результирующем изображении разными оттенками. Выполнение ядра классификации происходило для 20 разных типов текстур в течение 65 мс, аналогичный алгоритм классификации, основанный на многопоточных вычислениях на CPU для Intel Core2 6320 с частотами ядер 1,87 Ггц, запущенный с 8 потоками OpenMP, выполнялся в течение 10 с.



Оригинал Результат классификации

Рис. 4. Результаты классификации

Как показывают результаты тестов с визуальной оценкой качества распознавания на нескольких наборах тестов, предлагаемый подход реали-

зации классификации текстур на основе использования вычислений на графических картах имеет перспективы практического применения.

Исследование поддержано грантом Российского фонда фундаментальных исследований №14-07-31090 МОЛ_А.

Список литературы

1. Красильников Н.Н. Цифровая обработка 2D- и 3D-изображений: учеб. пособие. – СПб.: БХВ-Петербург, 2011. – 608 с.

2. Lukac R., Plataniotis K.N. Color Image Processing: Methods and Applications. – Boca Raton: Talor and Francis, 2007. – 600 p.

3. Petrou M., Sevilla P.G. Image Processing: Dealing with Texture. Taflove A. Computational Electrodynamics: The Finite-Difference Time-Domain Method. – London: Artech House, 1995. – 599 p.

А.С. Антонов, А.М. Теплов

Научно-исследовательский вычислительный центр
Московского государственного университета им. М.В. Ломоносова

О ПРАКТИЧЕСКОЙ СЛОЖНОСТИ ПОНЯТИЯ МАСШТАБИРУЕМОСТИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Рассмотрено понятие масштабируемости параллельных программ, и показана связь этого понятия с анализом множества факторов, влияющих на работу программы и параметров ее запуска. Обоснован подход к анализу масштабируемости с использованием нескольких параметров запуска и различных динамических характеристик для описания свойств и качества работы параллельных программ.

Введение

Исследование свойств параллельных программ направлено на получение данных о работе программы на вычислительной системе, об эффективности использования предоставленных вычислительных ресурсов и о других особенностях выполнения реализации алгоритма на вычислительной системе. На общую эффективность работы программы влияет большое количество факторов, и реальные программы обычно выполняются с производительностью, значительно меньшей пиковой [1].

Одним из важнейших свойств параллельной программы является ее масштабируемость. В литературе можно найти достаточно большое количество определений этого свойства [2, 3], которые в большинстве своем

указывают на тот факт, что для программы при изменении «масштабов» ее работы ряд факторов, снижающих эффективность, начинает оказывать большее или меньшее влияние. Результатом этого становится изменение таких показателей качества работы параллельной программы, как ускорение и производительность, характеризующая способность использовать дополнительные вычислительные ресурсы системы. Характерной чертой каждой параллельной программы является вид зависимости таких показателей качества, как интенсивность изменения производительности и характер изменения эффективности вычислений с ростом масштаба вычислительной системы.

Свойства понятия «масштабируемость»

Многие из определений масштабируемости указывают на то, что программа должна использовать ресурсы эффективно, однако рост накладных расходов часто приводит к падению эффективности. Таким образом, сохранение эффективности работы параллельной программы является важной составляющей понятия масштабируемости параллельной программы.

Часто свойство масштабируемости параллельной программы изображают графически в виде кривой, которая показывает зависимость ускорения или производительности программы от числа использованных процессоров. Отмечается влияние на характер зависимости других факторов, потому оговаривается, при каких параметрах получена зависимость. Влияние размера задачи привело к появлению понятий сильной масштабируемости, слабой и масштабируемости вширь [4].

Следует отметить, что понятия сильной, слабой масштабируемости и масштабируемости вширь уточняют понятие масштабируемости программы и характеризуют зависимость эффективности работы программы не только от объема вычислительных ресурсов, но и от размера задачи применительно к одной и той же параллельной программе. Влияние на эту зависимость оказывает значительно большее число факторов, и для понимания свойств исследуемой программы важно понимать, каким образом эти факторы оказывают взаимное влияние на качество ее работы.

Примером такого взаимного влияния нескольких факторов на эффективность работы параллельной программы и проявления в масштабируемости может служить явление сверхлинейного ускорения, вызванного удачным размещением данных программы в кеш-памяти при выполнении вычислений. Факт наличия такого явления говорит о свойстве алгоритма, при котором данные между вычислителями распределяются таким образом, что вычисления могут производиться в основном в рамках кеш-

памяти. Однако для одной и той же программы такая декомпозиция данных для разных размеров задачи наступит при разном числе вычислительных ресурсов. Таким образом, можно построить зависимость производительности или эффективности работы программы уже от двух параметров. Тогда сверхлинейное ускорение будет проявляться при наличии какого-то соотношения между размером задачи и объемом вычислительных ресурсов. На это соотношение оказывает влияние и объем кеш-памяти разного уровня. Можно найти и другие факторы, влияющие на вид зависимости, но сам факт его наличия определяет реализованный алгоритм.

Рассмотренный пример показывает, что такой вид зависимости характеризует свойство масштабируемости программы и является функцией значения производительности от нескольких параметров. При этом зависимость от этих параметров характеризует одновременно сильную, слабую масштабируемость и масштабируемость вширь.

На графике (рис. 1) показана зависимость производительности параллельной реализации скалярного произведения векторов от числа процессов и длины векторов. Видна область значений числа процессов и размера задачи, на которой проявляется сверхлинейное ускорение, связанное с помещением данных в кеш-память. Эта область повторяется при сохранении постоянного соотношения между размером задачи и числом процессоров.

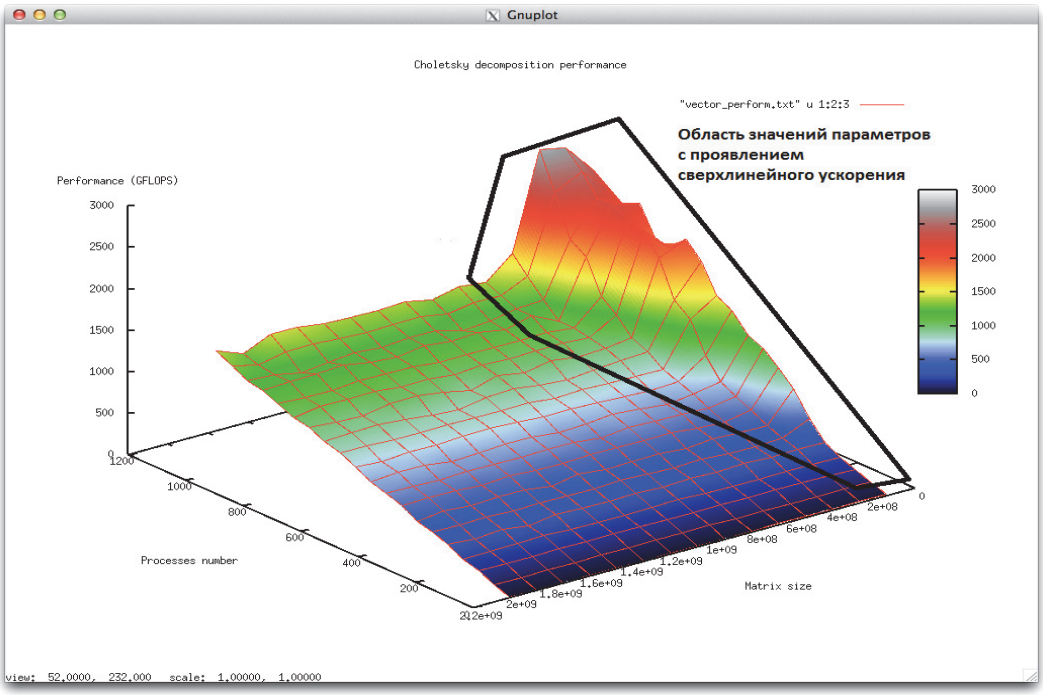


Рис. 1. Зависимость производительности скалярного произведения векторов от числа процессов и длины вектора. Обозначена область значений параметров, при которых проявляется сверхлинейное ускорение

При рассмотрении такой зависимости производительности от нескольких параметров становится понятно, что для изучения свойств программы недостаточно классического рассмотрения одномерной зависимости характеристик выполнения программы от количества процессоров, а необходимо исследовать зависимость этих характеристик от всех значений параметров, которые меняются при исследовании. Таким образом, при рассмотрении зависимости производительности той же программы еще и от объема кеш-памяти мы получили бы зависимость характеристик уже от трех параметров, а в общем случае таких параметров может быть значительно больше.

Параметром запуска параллельной программы будем называть любой параметр, оказывающий влияние на работу параллельной программы, не влияющий на реализацию алгоритма работы программы (например: размер задачи – a, число процессов – b, размер блока – c, число процессов на узел – d и т.п.).

Параметры запуска по типу можно классифицировать следующим образом:

– ***аппаратные*** – относящиеся к параметрам аппаратуры;

– ***системные*** – относящиеся к операционной системе, используемым библиотекам и системным приложениям и их настройкам;

– ***программные*** – влияющие на логику работы программы или вычислительную сложность задачи, различные параметры, предусмотренные разработчиком.

Важным при исследовании характеристик выполнения параллельной программы является и выбор рассматриваемого показателя качества работы программы. Во многих определениях масштабируемости важной характеристикой работы параллельной программы называют изменение эффективности ее работы на вычислительной системе. При рассмотрении эффективности работы программы как отношения достигнутой производительности к пиковой, если рассматривать свойство масштабируемости как зависимость от нескольких параметров производительности, эффективность работы будет частной производной производительности по числу процессов. Зависимость эффективности работы программы от тех же параметров позволит понять те области значений параметров, на которых работа программы остается на приемлемом уровне эффективности, а также те, на которых является недостаточной. Эти области значений легко получить, отсекая все значения, в которых эффективность ниже приемлемого уровня.

Для определения причин наличия особенностей масштабируемости параллельных программ можно использовать не только такие интегральные характеристики, как производительность, ускорение, эффективность.

Для определения причин снижения производительности можно рассматривать, как меняются характеристики более низкого уровня. Например, можно рассматривать изменение активности использования коммуникационной сети или жесткого диска при изменении параметров запуска.

Динамическими характеристиками параллельной программы будем называть такие характеристики ее работы, которые характеризуют качество работы параллельной программы и могут быть измерены или вычислены только при наличии данных о реальном ее выполнении. К динамическим характеристикам можно отнести такие характеристики, как время выполнения, ускорение, производительность, эффективность, средние и моментальные характеристики работы аппаратуры при выполнении данной параллельной программы.

Динамические характеристики можно разделить на несколько групп.

К первой можно отнести ***измеряемые характеристики***. Примерами таких характеристик могут быть время выполнения, количество кеш-промахов и т.п.

Ко второй можно отнести ***вычисляемые характеристики***, значения которых можно получить, исходя из данных измеряемых характеристик. Такие характеристики используются для лучшего понимания качества работы программы и могут использовать несколько измеряемых характеристик для лучшего представления данных. Примером может быть ускорение, моментальная и средняя эффективность и т.п.

Исходя из того, что для понимания свойств параллельной программы важно рассматривать все доступные изменяемые параметры запуска и для понимания причин найденных особенностей масштабируемости нужно рассматривать все доступные характеристики, нужно рассматривать масштабируемость как более общее свойство параллельной программы, включающее в себя как различные параметры запуска, так и различные динамические характеристики.

Масштабируемостью будем называть свойство параллельной программы, характеризующее зависимость изменения динамических характеристик работы этой программы от изменения параметров ее запуска.

Использование для анализа свойств параллельной программы масштабируемости в такой формулировке позволяет охарактеризовать качество ее выполнения, а также получить представление о том, как меняется качество использования вычислительных ресурсов при изменении различных параметров запуска программы, и сделать выводы о факторах, влияющих на эффективность ее работы.

Рассмотрение изменения большого числа характеристик и нескольких параметров требует наличия данных о значительно большем объеме данных о выполнении программы и проведения значительно большего числа экспериментов. Это вызвано необходимостью получения данных о работе исследуемой программы на каждой конфигурации параметров запуска из рассматриваемых границ значений параметров.

Для такой формулировки понятия масштабируемости более классические понятия сильной, слабой масштабируемости и масштабируемости вширь будут частным случаем фиксированных параметров запуска программы. Для сильной масштабируемости фиксирован размер задачи, для слабой – фиксировано соотношение размера задачи к числу процессов, а для масштабируемости вширь фиксировано число процессов.

Преимущества предложенного подхода

Такой подход к понятию масштабируемости позволяет рассматривать изменение влияния на нее различных факторов, и потому не только находить значения параметров запуска, при которых работа программы имеет высокую производительность и достаточную эффективность работы, но и делать выводы о том, какие параметры вычислительной системы не позволяют увеличивать эффективность выполнения. В свою очередь, данные о наличии таких проблем и их причинах позволяют более подробно исследовать свойства программы.

При анализе собранных данных основное внимание следует уделять корреляции между различными динамическими характеристиками и связи между ними.

Примером корреляций таких динамических характеристик, рассматривая которые можно сделать выводы о причинах снижения производительности программы, может служить зависимость производительности блочного перемножения матриц от числа используемых процессоров и размера задачи [5].

На рис. 2 показана зависимость производительности блочного перемножения матриц от размеров задачи и числа процессов. Также на рисунке обозначены результаты замеров числа кеш-промахов в секунду, полученные системой мониторинга состояния вычислительной системы в момент работы приложения. В области с высокой производительностью число кеш-промахов значительно меньше, и эта область значений с высокой производительностью соответствует матрице меньшего размера. Наличие такой корреляции зависимости роста производительности и падения числа кеш-промахов второго уровня при малом размере задачи указывает на то, что данные лучше расположены в кеш-памяти, что приводит к увеличению эффективности.

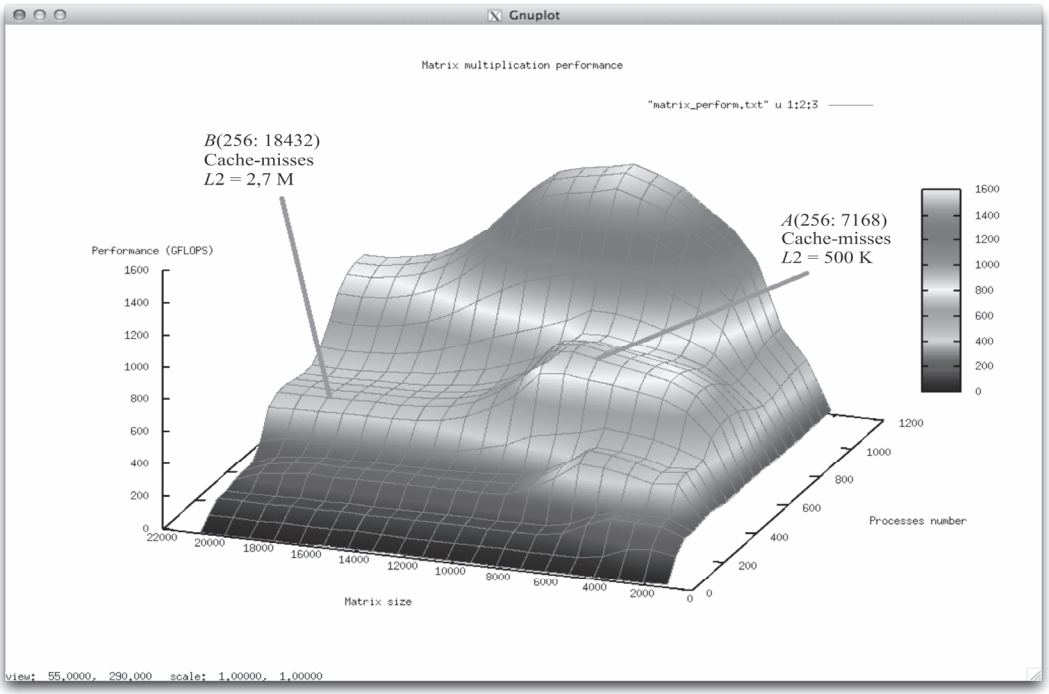


Рис. 2. Зависимость производительности блочного перемножения матриц от числа процессов и размера задачи. В точках *A* и *B* экспериментов при 256 процессах и матрицах размером 7168 и 18 432 число кеш-промахов второго уровня 500 тыс/с и 2,7 млн/с

Зная такого рода корреляции при анализе масштабируемости параллельной программы, можно извлекать больше данных об особенностях программы и ее свойствах.

Выводы

Обобщение понятия масштабируемости параллельной программы требует значительно большего объема данных о работе программы, однако позволяет более подробно и полно раскрыть важные свойства работы программы, недоступные при классическом подходе. К таким свойствам можно отнести взаимное влияние параметров запуска программы на общую производительность и эффективность работы, влияние на отдельные динамические характеристики параметров запуска параллельной программы. Такой подход дает возможность подобрать наиболее подходящие параметры запуска параллельной программы для оптимизации использования вычислительных ресурсов при увеличении производительности работы.

Работа выполняется при поддержке гранта РФФИ 13-07-00790 а.

Список литературы

1. Воеводин В., Воеводин В.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.

2. Бухановский А.В., Иванов С.В. Проектирование прикладного математического обеспечения параллельной обработки данных // Технологии параллельного программирования: учеб. материалы Зимней шк.-практикума. – 2006.

3. Коньшин В.Н. Параллельная реализация программного комплекса FlowVision // САПР и графика. – М.: КомпьютерПресс, 2006. – № 12 (122). – С. 1–4.

4. Антонов А.С., Теплов А.М. Исследование масштабируемости программ с использованием инструментов анализа параллельных приложений на примере модели атмосферы NH3D // Вестник Юж.-Урал. гос. ун-та. Вычислительная математика и информатика. – 2013. – Т. 2, № 1.

5. Антонов А.С., Теплов А.М. Использование данных системного мониторинга для определения факторов, уменьшающих (изменяющих или снижающих) масштабируемость приложения // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: тр. Всерос. науч. конф., 22–27 сентября 2014 г., г. Новороссийск. – М.: Изд-во МГУ, 2014. – С. 87–96.

С.С. Артемьев, В.Л. Лукинов

[1]Институт вычислительной математики и математической геофизики
Сибирского отделения РАН, Новосибирск
[2]Новосибирский государственный университет

ОСОБЕННОСТИ СТОХАСТИЧЕСКИХ ВЫЧИСЛЕНИЙ ДЛЯ СДУ И КРАЕВЫХ ЗАДАЧ НА СУПЕРКОМПЬЮТЕРАХ

Представлены особенности стохастических вычислений, связанные с решением стохастических дифференциальных уравнений и краевых задач. Выделены классы задач, решение которых требует применения расчетов на суперкомпьютерах. На примере обобщенной схемы Эйлера для решения систем СДУ со случайной структурой исследована катастрофическая потеря точности решения, требующая высокопроизводительных вычислений. Рассматриваются преимущества и недостатки параллельной реализации методов «блуждание по решетке» и «блуждание по сферам» для оценок решения и ковариации решения краевых задач эллиптического типа. Показываются зависимости точности разработанных методов от числа шагов и объемов выборки. Приводятся результаты численных экспериментов, проведенных в Сибирском суперкомпьютерном центре.

Введение

В связи с революционным характером развития технологий параллельного программирования возрастает интерес к параллельным численным методам решения прикладных задач, в частности к статистическому моделированию (или методу Монте-Карло). Отличительными особенностями алгоритмов статистического моделирования являются относительная простота их построения и высокая трудоемкость, особенно при требованиях высокой точности оценок. В связи с этим ставится проблема расширения сферы практического применения и повышения эффективности алгоритмов метода Монте-Карло для решения задач в различных областях науки. В данной работе рассматриваются следующие способы повышения эффективности алгоритмов метода Монте-Карло:

– способы уменьшения дисперсии полученных оценок решения (ценностное моделирование, весовые методы, условное осреднение);

– организация параллельных вычислений (динамический параллелизм, балансировки для статического параллелизма, выбор между балансировкой по пространственной сетке или по траекториям);

– использование машинных оптимизаций (векторные вычисления, работа с кешем, конвейерные вычисления с помощью средств компилятора).

Стохастические дифференциальные уравнения

Имеются различные определения стохастических дифференциальных уравнений (СДУ), например [1]

$$\frac{du}{dt} = F(t, x, u(t, x), Du(t, x), D2u(t, x)) + G(t, x, u(t, x), Du(t, x))W(t, x).$$

Можно рассматривать $W(t, x)$ либо как «белый шум» и по времени, и по пространству, либо как коррелированный шум по пространству. Например, переход от детерминированных уравнений Навье – Стокса (УНС) в гидродинамике и аэродинамике к СДУ происходит следующим образом. В классической задаче о плоском изотермическом течении вязкой жидкости в прямоугольной полости (каверне) модель течения жидкости задается системой УНС

$$\frac{\partial V_x}{\partial t} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 V_x}{\partial x^2} + \frac{\partial^2 V_x}{\partial y^2}\right) - \left(V_x \frac{\partial V_x}{\partial x} + V_y \frac{\partial V_x}{\partial y}\right) + F_x(t, x, y);$$

$$\frac{\partial V_y}{\partial t} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 V_y}{\partial x^2} + \frac{\partial^2 V_y}{\partial y^2}\right) - \left(V_x \frac{\partial V_y}{\partial x} + V_y \frac{\partial V_y}{\partial y}\right) + F_y(t, x, y).$$

Внешние возбуждения $F(t, x, y)$ могут рассматриваться как случайные. После дискретизации производных по пространственным переменным придем к большой системе (тысячи уравнений) обыкновенных СДУ.

Простейшим и наименее трудоемким численным методом для решения задачи Коши для систем СДУ в смысле Ито вида

$$dy = f(y)dt + \sigma(y)dw(t), y(0) = y_0, 0 \le t \le T$$

является обобщенный явный метод Эйлера [2]

$$y_{n+1} = y_n + hf(y_n) + \sqrt{h}\sigma(y_n)\xi_{n+1}, n = 1, \ldots, N-1.$$

Рассмотрим трудоемкие характеристики решений СЧДУ, рассчитываемых методом Монте-Карло, когда моделируются миллионы траекторий решений СДУ. При численном анализе систем ОДУ с осциллирующими решениями принято наряду с графиком интегральной кривой исследовать фазовый портрет двух компонентов решения. В качестве аналогов интегральной кривой и фазового портрета для решений систем СДУ можно использовать частотную интегральную кривую (ЧИК) и частотный фазовый портрет (ЧФП) (рис. 1–3). Введем формальные определения ЧИК и ЧФП [2]. Сетка S_k и ЧИК W_k:

$$S_k = \left\{ (U_i, t_j) \left| \begin{array}{l} h_U = \dfrac{U_{N_U} - U_0}{N_U}, U_0 = A, U_{N_U} = B, U_i = U_0 + ih_U, i = 0, \ldots, (N_U - 1) \\ h_t = \dfrac{t_{N_t} - t_0}{N_t}, t_0 = C, t_{N_t} = D, t_j = t_0 + jh_t, j = 0, \ldots, (N_t - 1) \end{array} \right. \right\},$$

Здесь $[C, D] \subseteq [0, T]$, h_U и h_t – шаги сетки ЧИК по пространству и времени:

$$W_k = \left\{ (W_{ij}) \left| \begin{array}{l} W_{ij} - \text{количество пар } \left(y_l(t_k), t_j \right), \text{попавших в } (i, j)\text{-узел сетки } S_k, \\ \text{номер узла сетки } i = \left[\dfrac{y_l(t_k) - A}{h_U} \right], j = \left[\dfrac{t_j - C}{h_t} \right] \end{array} \right. \right\}.$$

Здесь [ ] – целая часть числа, $y_l(t_k)$ – значение l-го компонента решения СДУ в узле интегрирования t_k. Сетка S_p и ЧФП W_p:

$$S_p = \left\{ (U_i, V_j) \left| \begin{array}{l} h_U = \dfrac{U_{N_U} - U_0}{N_U}, U_0 = A, U_{N_U} = B, U_i = U_0 + ih_U, i = 0, \ldots, (N_U - 1) \\ h_V = \dfrac{t_{V_t} - V_0}{N_V}, V_0 = C, V_{N_V} = D, V_j = V_0 + jh_V, j = 0, \ldots, (N_V - 1) \end{array} \right. \right\}.$$

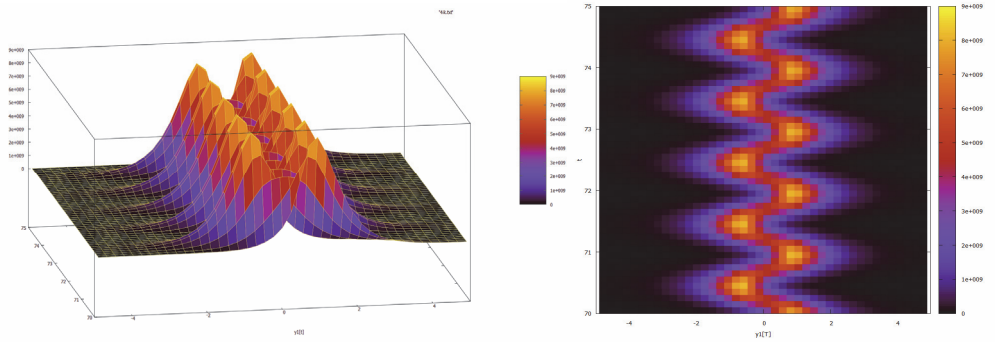


Рис. 1. ЧИК (сверху) и ЧИК(сбоку)

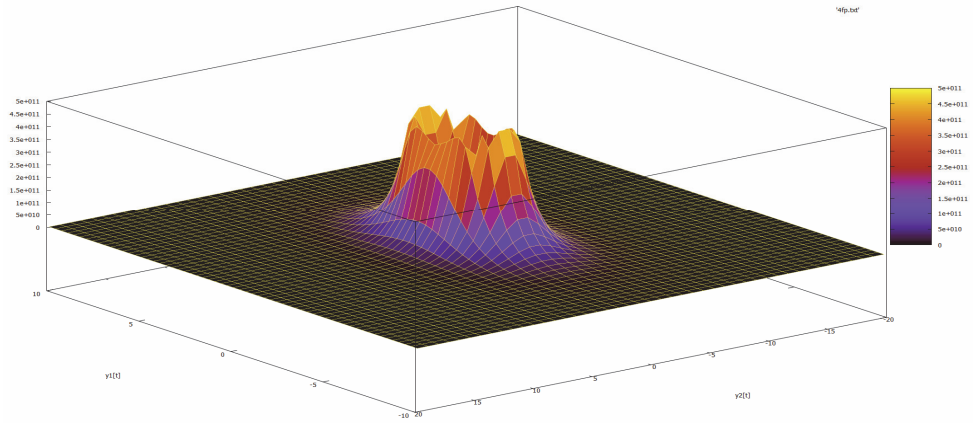


Рис. 2. ЧФП (сбоку)

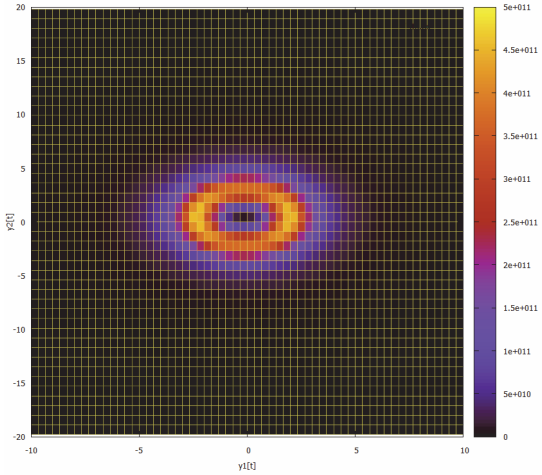


Рис. 3. ЧФП (сверху)

Здесь $A \le B, C \le D$ – произвольны, h_U и h_V – шаги сетки ЧФП по пространству.

$$W_p = \left\{ (W_{ij}) \middle| \begin{array}{l} W_{ij} - \text{количество пар } \left(y_l(t_k), y_m(t_j) \right), \text{ попавших в } (i,j)\text{-узел} \\ \text{сетки } S_p, \text{ номер узла сетки } i = \left[\dfrac{y_l(t_k) - A}{h_U} \right], j = \left[\dfrac{y_m(t_j) - C}{h_V} \right] \end{array} \right\}.$$

Здесь $y_l(t_k), y_m(t_j)$ – значения l-го и m-го компонентов решения СДУ.

Рассмотрим случай «шумящих» коэффициентов в линейном колебательном контуре, который задается СДУ:

$$\begin{aligned} dy_1 &= y_2 dt, \\ dy_2 &= -\left(\omega^2 y_1 + \lambda y_2\right) dt - \sigma_1 y_2 d\omega_1(t) - \sigma_2 y_1 d\omega_2(t). \end{aligned} \tag{1}$$

Для расчета ЧФП и ЧИК задачи (1) на сетке размером 300×300 моделировалось 106 траекторий решения системы СДУ, при этом использовалось 96 ядер кластера ССКЦ НКС-30Т+ GPU.

Краевые задачи

Примером краевой задачи, для решения которой применяются различные статистические алгоритмы, является метаэллиптическое уравнение в области D с условиями Дирихле на границе Γ:

$$\left(\Delta + c\right)^{p+1} u = -g, \left(\Delta + c\right)^k u\Big|_\Gamma = \varphi_k, k = 0, \ldots, p. \tag{2}$$

В алгоритмах «блуждания по сферам» оценка решения в заданной точке является функционалом на приближеннных траекториях диффузионного процесса, соответствующего оператору Лапласа [3, 5]. Преимуществом такого подхода является малая трудоемкость для выпуклых областей D, недостатком – требование параметра c быть константой. С помощью данного подхода разработан эффективный алгоритм оценки ковариации решения бигармонических уравнений [4], описывающих колебание тонких пластин под действием случайных нагрузок g, φ_0 и φ_1:

$$\Delta^2 u + cu = -g, u\Big|_\Gamma = \varphi_0, \Delta u\Big|_\Gamma = \varphi_1. \tag{3}$$

Если требуется получить глобальную оценку решения во всей области, используется замена дифференциального оператора конечно-разностным. С этой целью обычно строится равномерная сетка на области D, далее с помощью алгоритмов «блуждания по решетке» находятся значения в узлах сетки, затем с помощью линейного восполнения получается глобальное решение в любой точке области. Уменьшение трудоемкости для данного алгоритма получено с помощью выбора начального распределения.

Параллельные программы

Математическое ожидание оценок решения в n-м узле сетки для представленных методов находится с помощью следующей схемы:

$$m_n = \frac{1}{M}\sum_{m=1}^{M} y_n^m, n = 1,\dots N. \qquad (4)$$

Здесь y_n^m – значение m-й траектории в n-м узле сетки. При очень больших M и N требуются большие затраты машинного времени, поэтому распараллеливание вычислений необходимо. Для решения СДУ моделируются независимые траектории с одинаковой длиной по узлам временной сетки. Это допускает эффективную организацию их параллельного вычисления на многопроцессорном кластере. Схема распараллеливания в данном случае проста: различные процессоры вычислительной системы осуществляют полностью независимое решение, т.е. вычисляют последовательности, полученные на основе разных (в каждом процессоре) псевдослучайных чисел. Таким образом, моделируется последовательность, построенная на выборке независимых по процессорам одинаково распределенных случайных величин. При решении краевых задач (2) и (3) длины независимых траекторий в узлах пространственной сетки различны. Таким образом, вычислительная нагрузка для разных процессоров становится разной. С помощью организации динамической нагрузки удалось повысить эффективность работы алгоритма нахождения решения металлиптического уравнения на 10 %. Этот результат объясняется выравниванием нагрузки при значительном (больше 10^6) числе траекторий.

Следующая техника применяется для предотвращения потерь точности при округлении величин. Для простоты рассмотрим однородный кластер, распространение на случай гетерогенных кластеров очевидно. Пусть K – количество процессоров, M_k – количество траекторий на k-м процессоре. Очевидно, что для достижения высокой точности при достаточно мелком разбиении сетки величина M_k на каждом процессоре будет большой. Разобьем M_k траекторий на группы по B траекторий так, что $M_k = B \cdot M_l$. Для уменьшения потери точности в рассматриваемых задачах размеры B и M_l брались приблизительно одинаковыми. Формула (4) принимает вид

$$m_n = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{b=1}^{B}\frac{1}{M_l}\sum_{m=1}^{M_l} y_n^{m,k}.$$

Для оценки точности расчетов рассматриваемых алгоритмов метода Монте-Карло используется следующая формула оценки второго момента γ:

$$\hat{\gamma} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{M_l}\sum_{m=1}^{M_l}\left(y_n^{m,k}\right)^2.$$

При исследовани численного решения СДУ (1) и решении краевых задач (2), (3) алгоритмами «блуждания по решетке» необходим большой объем памяти для хранения результатов вычислений в узлах сетки. Особенно данное ограничение существенно при проведении расчетов с использованием графических карт. Одним из способов уменьшения потребления памяти является хранение результатов расчетов в узлах более крупной сетки. Другим, более сложным в огранизации, является алгоритм, основанный на распараллеливании по подобластям сетки (данным). Векторизация в большинстве случаев идеально подходит для статистических алгоритмов и позволяет добиться увеличения производительности на 25–40 % на каждом процессоре. Эксперименты со схемами использования MPI процессов и OpenMP потоков показали, что для рассматриваемых алгоритмов лучше всего MPI процессы использовать только для пересылки конечных результатов, а всю вычислительную нагрузку на процессор или графическую карту распределять с помощью OpenMP потоков.

Работа подготовлена при поддержке грантов РФФИ (№ 14-01-31451 и № 14-01-00340) и Новосибирским государственным университетом.

Список литературы

1. Артемьев С.С., Корнеев В.Д., Якунин М.А. Численное решение стохастических дифференциальных уравнений со случайной структурой на суперкомпьютерах // Сиб. журн. вычисл. матем. – 2013. – № 16 (4). – С. 303–311.

2. Артемьев С.С., Иванов А.А., Корнеев В.Д. Численный анализ стохастических осцилляторов на суперкомпьютерах // Сиб. журн. вычисл. матем. – 2012. – С. 31–43.

3. Lukinov V.L., Mikhailov G.A. The probabilistic representation and Monte Carlo methods for the first boundary value problem for a polyharmonic equation // Rus. J. Num. Anal. and Math. Modell. – Vol. 19, № 5. – P. 434–449.

4. Lukinov V.L. Simulation of the deflection of thin plates under the action of random loads // International Journal on Advances in Systems and Measurements. – 2012. – Vol. 5, no. 4, 5. – P. 103–106.

5. Mikhailov G.A. Parametric Estimates by the Monte Carlo method. – Utrecht, Nederlands, 1999.

Д.Р. Ахмеджанов, С.А. Лебедев

Нижегородский государственный университет им. Н.И. Лобачевского

ОБ ОДНОМ ПОДХОДЕ К РЕАЛИЗАЦИИ СИМВОЛЬНОЙ ФАЗЫ РАЗЛОЖЕНИЯ ХОЛЕЦКОГО ДЛЯ РАЗРЕЖЕННЫХ МАТРИЦ

Рассмотрен подход к осуществлению одного из этапов прямого решения СЛАУ с разреженной симметричной положительно определенной матрицей – символьной фазы разложения Холецкого. Описана поэтапная схема алгоритма. Программная реализация представленного метода выполнена в контексте прямого решателя разреженных СЛАУ, разрабатываемого на ВМК ННГУ. Приведены результаты экспериментов, показано превосходство предложенной реализации по сравнению с ранее разработанным аналогом.

Введение

Решение систем линейных алгебраических уравнений вида $Ax = b$ с разреженной симметричной положительно определенной матрицей A лежит в основе многих вычислительных задач компьютерной алгебры [1–3]. 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 (для симметричных положительно определенных матриц – разложение методом Холецкого) с последующим решением треугольных систем [4]. В этом случае решение системы выполняется в несколько этапов:

1. *Переупорядочение матрицы*: перестановка строк и столбцов матрицы для уменьшения размера фактора Холецкого.

2. *Символьная фаза*: анализ структуры исходной матрицы и построение портрета фактора.

3. *Численная фаза*: заполнение полученного портрета численными значениями.

4. *Решение треугольных систем*.

Таким образом, одним из существенных этапов решения является символьная фаза, которая определяет позиции ненулевых элементов в результирующей треугольной матрице разложения, а также строит вспомогательные структуры данных [5]. Несмотря на то, что основной вклад в общее время работы решателя вносит, как правило, численная фаза, оптимизация по скорости символьной фазы также представляет интерес. Исследо-

ваниями в этой области занимались J.W.H. Liu, T. Davis, J.R. Gilbert, E.G. Ng, B.W. Peyton и др.

В частности, J.W.H. Liu в 1990 г. разработал не только один из самых эффективных алгоритмов поиска дерева исключений и нашел ему множество применений, но и работал над алгоритмами подсчета ненулевых элементов в результирующем факторе. Позже, в 1994 г., J.R. Gilbert, E.G. Ng и B.W. Peyton его улучшили [6]. Они также вели исследования в области применения супернодов к символьной части [5].

В работе предлагается способ построения и реализации символьной фазы разложения Холецкого, основанный на комбинации ранее опубликованных алгоритмов с некоторыми модификациями. Приведены текущие результаты, дано сравнение с аналогом, разработанным ранее. Результаты используются на кафедре математического обеспечения ЭВМ факультета ВМК ННГУ при создании прямого решателя для больших разреженных СЛАУ[6,7].

Постановка задачи

Дана разреженная симметричная положительно определенная матрица A. Необходимо найти портрет (расположение ненулевых элементов) верхней треугольной матрицы U такой, что $U^T U = A$.

Метод решения

Будем использовать следующую схему вычислений:

1. Построение дерева исключения T(A).

Деревом исключения $T(A)$, соответствующим матрице A, называется дерево, множество вершин которого совпадает с множеством вершин графа матрицы (т.е. множеством строк), а множество ребер $E(T)$ задается соотношением

$$e_{i,j} \in E(T) \leftrightarrow i = \min_k \{u_{j,k} \neq 0, k > j\},$$

где $u_{j,k}$ – элемент фактора матрицы [8].

Построение дерева исключения выполняется в соответствии с алгоритмом, описанным в работе [3]. Алгоритм работает за время $O(n)$.

2. Подсчет количества ненулевых элементов в факторе U.

Для того чтобы определить портрет результирующей матрицы, необходимо подсчитать количество ненулевых элементов в ее строках или столбцах. В данной работе используется «столбцовый» вариант, позволяющий упростить дальнейшие вычисления. Вычислительная сложность

алгоритма составляет $O(n\alpha(n))$ [9], где $\alpha(n)$ – обратная функция Аккермана. Данная функция возрастает крайне медленно, что позволяет говорить об этом алгоритме как о работающем за почти линейное время.

3. Построение портрета фактора U.

Для построения портрета используется процедура, описанная в работе [7], но в соответствии со столбцовым вариантом этапа 2. Данная модификация позволяет сразу получить упорядоченный портрет фактора без дополнительного транспонирования.

Программная реализация

В рамках данной работы производилась программная реализация новой модификации символьной части с применением алгоритма подсчета ненулевых элементов результирующей матрицы в столбцах. Реализация выполнена на языке программирования C с использованием пакета инструментов IntelParallelStudioXE 2013 SP1.

Результаты вычислительных экспериментов

Для настоящей программной реализиции был выполнен ряд вычислительных экспериментов для выяснения времени работы символьной части разложения Холецкого матриц из коллекции университета Флориды [1]. Характеристики тестовых матриц и параметры тестового окружения приведены в табл. 1, 2.

В табл. 3 приведены результаты экспериментов по сравнению с предыдущей версией символьной фазы [7].

Таблица 1

Характеристики тестовых матриц

Название матрицы	Порядок	Число ненулевых элементов в верхнем треугольнике	Заполненность, %
pwtk	217 918	5 926 171	0,0250
msdoor	415 863	10 328 399	0,0119
parabolic_fem	525 825	2 100 225	0,0015
tmt_sym	726 713	2 903 835	0,0011
boneS10	914 898	28 191 660	0,0067
Emilia_923	923 136	20 964 171	0,0049
audikw_1	943 695	39 297 771	0,0088
ecology2	999 999	2 997 995	0,0006
Hook_1498	1 498 023	31 207 734	0,0028
Flan_1565	1 564 794	59 485 419	0,0049
G3_circuit	1 585 478	4 623 152	0,0004

Таблица 2

Параметры тестового окружения.
Расчеты выполнены на суперкомпьютере ННГУ «Лобачевский»[1]

№ п/п	Процессор	Intel Sandy Bridge E5-2660 2.2 GHz (8 ядер)
1	Память	64GB
2	Операционная система	Linux (CentOS 6.4)
3	Компилятор	Intel C++ Composer XE 2013

Таблица 3

Результаты вычислительных экспериментов.
Лучшие времена выделены

Название матрицы	Порядок	Время работы предыдущей версии, с	Время работы разработанной версии, с
pwtk	217 918	0,73	0,78
msdoor	415 863	1,23	0,86
parabolic_fem	525 825	0,41	0,4
tmt_sym	726 713	0,5	0,49
boneS10	914 898	4,8	4,83
Emilia_923	923 136	71,9	35,89
audikw_1	943 695	51,6	26,4
ecology2	999 999	0,6	0,6
Hook_1498	1 498 023	43,2	29,7
Flan_1565	1 564 794	38,25	29,3
G3_circuit	1 585 478	1,65	1,56

Эксперименты показали, что для большинства матриц удалось получить ощутимый выигрыш во времени работы символьной фазы (30–50 %).

Заключение

Разработанная на данный момент символьная фаза разложения Холецкого для разреженных матриц показывает превосходство по сравнению с ранее разработанной версией. Целью дальнейших исследований является продолжение работы над представленным алгоритмом. Предполагается использовать супернодальный подход [5], что позволит получить выигрыш во времени за счет меньшего количества кеш-промахов, а также распараллелить код для систем с общей памятью.

Работа выполнена в лаборатории «Информационные технологии» ВМК ННГУ при поддержке гранта РФФИ №14-01-3145514.

[1] Суперкомпьютер ННГУ «Лобачевский». URL: http://hpc-education.unn.ru/ru/ресурсы.

Авторы благодарят Иосифа Борисовича Меерова, Анну Юрьевну Пирову, Александра Владимировича Сысоева, Евгения Александровича Козинова и Алексея Владимировича Линёваза за полезные обсуждения и внимание к работе.

Список литературы

1. Davis T., Hu Y. The University of Florida sparse matrix collection // ACM Trans. Math. Software. – 2011. – Vol. 38, no. 1, article 1.

2. Писсанецки С. Технология разреженных матриц. – М.: Мир, 1988. – 410 с.

3. Davis T.A. Direct methods for sparse linear systems. – Philadelphia: SIAM, 2006. – 217 p.

4. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. – М.: Мир, 1984. – 333 с.

5. Ng E.G., Peyton B.W. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors. Mathematical Sciences Section / OakRidgeNationalLaboratory. – 1991.

6. Gilbert John R., Esmond G.Ng, Peyton Barry W. An efficient algorithm to compute row and column counts for sparse Cholesky factorization. – 1994. – Vol. 15, iss. 4.

7. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей / Е.А. Козинов [и др.] // Вестник Нижегор. ун-та им. Н.И. Лобачевского. – Н. Новгород: Изд-во ННГУ, 2012. – № 5 (2). – С. 376–384.

8. Liu J. W-H., Esmond G.Ng, Peyton B.W. On finding supernodes for sparse matrix computations // SIAM J. Matrix Anal and Appl. – 1993. – Vol. 14, iss. 1.

9. Tarjan R.E. Efficiency of a good but non linear set union algorithm // J. ACM. – 1975. – № 22. – P. 690–715.

10. Pirova A., Meyerov I. MORSy – a new tool for sparse matrix reordering // An International Conference on Engineering and Applied Sciences Optimization, 4–6 June. – Kos Island, Greece, 2014. – P. 1952–1964.

И.В. Ашарина

Национальный исследовательский университет «МИЭТ», Зеленоград

МЕТОД ПОСТРОЕНИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ВЗАИМНОГО ИНФОРМАЦИОННОГО СОГЛАСОВАНИЯ В МНОГОКЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Предлагается метод построения параллельного алгоритма системного взаимного информационного согласования в многокластерной вычислительной системе.

Введение

Рассматривается задача построения распределенного алгоритма вычисления согласованных значений в многомашинной вычислительной системе, в которой одновременно решается несколько взаимодействующих задач, при этом для каждой задачи выделяется кластер (комплекс [1]) , т.е. подсистема, удовлетворяющая определенным структурным требованиям [2, 3], все ЦВМ которой решают копии данной задачи (репликация задачи), обмениваются результатами и выбирают из них правильный.

В качестве модели сети применяется ориентированный граф (орграф), пронумерованные от 1 до n вершины которого отображают ЦВМ с соответствующими номерами, а дуги – существующие каналы межмашинной связи. Кластеры взаимодействуют между собой через соответствующие выделенные среды межкластерных посылок, отображаемые соответствующими орподграфами [1–3] и включающие среды взаимообмена между каждой парой задач.

Ставится задача создания метода, который по известным структурам выделенных кластеров и сред посылок строит распределенный параллельный алгоритм системного взаимного информационного согласования (СВИС) вычисления вектора согласованных значений системы (ВСЗС) во всех исправных ЦВМ системы. Алгоритм СВИС состоит из взаимосвязанных, индивидуальных для каждой ЦВМ алгоритмов, выполняемых этими ЦВМ. ВСЗС должны быть одинаковыми во всех исправных ЦВМ системы, причем согласованные значения, соответствующие исправным ЦВМ, должны быть равны их согласуемым значениям.

Работа системы в процессе подготовки СВИС

Для построения искомой многокластерной системы, способной к осуществлению СВИС, в данной сети выделяется, если это возможно, структура системы, содержащая непересекающиеся кластеры, соответствующие

задачам, и необходимые среды межкластерных посылок. Наличие непересекающихся кластеров и непересекающихся межкластерных сред посылок является достаточным условием достижения СВИС в рассматриваемой многокластерной системе. Эти достаточные условия, формулируемые на языке теории графов, приведены и доказаны в работах [2, 3].

Полным орграфом называется орграф, имеющий пару разнонаправленных дуг между любыми двумя вершинами. Орграфы являются гомеоморфными, если существуют их изоморфные подразбиения, образующиеся в результате подразбиения дуг [4]. Вершины орподграфа T, взаимно однозначно соответствующие вершинам полного орграфа, называются основными и составляют множество $X = \{x_1, x_2, \ldots x_i, \ldots\}$. Остальные вершины орподграфа T называются неосновными и составляют множество $N = PUQ$, $N = \{n_1, n_2, \ldots n_i, \ldots\}$. Вершины из P называются также разделительными [1]. Кластер содержит орподграф, полный или гомеоморфный полному орграфу, с количеством вершин более 3μ, где μ – допустимое количество неисправных ЦВМ в кластере.

Процесс СВИС состоит из следующих этапов: 1) посылка согласуемых значений из неосновных ЦВМ каждого кластера в основные ЦВМ этого кластера; 2) внутрикластерное взаимное информационное согласование (ВКИС) [1–3] с вычислением в каждой основной ЦВМ каждого кластера вектора согласованных значений этого кластера (ВСЗК); 3) посылка векторов согласованных значений из основных ЦВМ каждого кластера в неосновные ЦВМ этого кластера; 4) межкластерный обмен векторами согласованных значений кластеров через соответствующие среды межкластерных посылок; 5) внутрикластерное согласование векторов согласованных значений кластеров-источников в кластерах-получателях [1–3] и формирование ВСЗС в каждой из ее ЦВМ.

Наличие в системном орграфе кластера орподграфа, полного или гомеоморфного полному, является первой частью (Ч1) достаточных условий достижения ВИС в орграфе T. Вторая составная часть (Ч2) достаточных условий достижения ВИС в кластере: для каждой его неосновной вершины существуют исходящие из этой вершины и далее не пересекающиеся пути к не менее чем $2\mu + 1$ конечных основных вершин. Третья составная часть (Ч3): для каждой его неосновной вершины существуют входящие и ранее не пересекающиеся пути от не менее чем $2\mu + 1$ начальных основных вершин.

В работах [2, 3] предлагаются алгоритмы А1–А4 последовательного выделения в системе заданного множества непересекающихся кластеров, удовлетворяющих требуемым характеристикам по сбое- и отказоустойчивости, и сред межкластерных посылок, обеспечивающих возможность

СВИС. Разработанные алгоритмы являются подготовительным этапом к построению распределенного параллельного алгоритма СВИС вычисления ВСЗС во всех исправных ЦВМ системы. Результатом работы алгоритмов А1–А4 являются построенные логические выражения, которые полностью отображают структуру системы и, соответственно, данных, необходимых для реализации предлагаемого здесь алгоритма.

Принципы работы алгоритма, обеспечивающего построение распределенного алгоритма СВИС

Рассматривается задача разработки алгоритма А5, который должен строить распределенный алгоритм, реализованный в виде множества взаимодействующих между собой алгоритмов, индивидуальных для каждой ЦВМ, каждый из которых реализуется на своей ЦВМ в составе системы, обеспечивая в каждой ЦВМ вычисление ВСЗС, одинакового во всех исправных ЦВМ системы, такого, что для исправных ЦВМ согласованное значение совпадает с согласуемым значением, а для одной и той же неисправной ЦВМ согласованное значение является одинаковым во всех исправных ЦВМ.

Количество принимаемых сообщений, их состав, форматы и кванты передачи определяются на основе анализа выражений, построенных при работе алгоритмов А1–А4 [2, 3], которые разбивают МВС на требуемое число кластеров с заданными степенями отказоустойчивости, выделяют среды межкластерных посылок, ранжируют вершины каждого кластера в процессе СВИС.

В процессе работы параллельного алгоритма СВИС происходит многократный обмен сообщениями между ЦВМ, организованный по временным квантам. Квант – это период обмена сообщениями между соседними ЦВМ, включающий действия по формированию и обработке этих сообщений внутри ЦВМ. Собственное согласуемое значение *k*-й ЦВМ системы, формируемое в первом кванте процесса СВИС, обозначим через σ_k, где *k* – номер ЦВМ. На этапе ВКИС согласуемое значение основной ЦВМ, кроме собственного согласуемого значения этой ЦВМ, также может содержать согласуемые значения некоторых неосновных ЦВМ, поступивших в нее на первом этапе. Составное согласуемое значение *i*-й основной ЦВМ обозначим $\Xi_i^t = \{\sigma_{p,i}, \sigma_{q,i}, \sigma_{s,i}, ..., \sigma_i\}$, где верхний индекс обозначает номер кванта, нижние индексы – номера ЦВМ, из которых формируется это составное согласуемое значение.

В данной работе предлагается метод, заключающийся в следующем. Согласуемые значения являются элементами сообщения и передаются ме-

жду ЦВМ в процессе СВИС в составе сообщений $S_{i,j}^{u}$, где u – номер кванта передачи, i – номер ЦВМ-передатчика, j – номер ЦВМ-приемника. При подготовке процесса СВИС для каждого сообщения $S_{i,j}^{t}$ каждого кванта в памяти каждой i-й ЦВМ-передатчика системы выделяется специально организованная область памяти ЦВМ – выходной контейнер $K_{i,j}^{\mathrm{Out},t}$, а в памяти j-й ЦВМ-приемника системы – входной контейнер $K_{i,j}^{\mathrm{In},t}$. Состав и последовательность согласуемых значений (элементов) в сообщении определяют его формат, соответствующий формату контейнера, согласно информации, заложенной в построенных ранее логических выражениях.

В каждой ЦВМ количество входных контейнеров равно количеству ожидаемых входных сообщений и количество выходных контейнеров – количеству создаваемых выходных сообщений. Разрабатываемый алгоритм задает для каждой ЦВМ, откуда, в каком кванте и какого формата поступают к ней входные сообщения, а также выходные сообщения каких форматов должны быть составлены ею из элементов поступивших входных сообщений и когда и куда должны быть переданы эти выходные сообщения.

Передача сообщения происходит между двумя соседними ЦВМ-передатчиком и ЦВМ-приемником. Посылка сообщения из ЦВМ-источника в ЦВМ-получатель по назначенному пути осуществляется в виде поквантовой последовательности передач сообщений в этом пути. В течение кванта в ЦВМ-получателе выполняются определенные действия по обработке поступивших сообщений и формированию выходных сообщений, называемые квантовыми действиями. Номер кванта передачи каждого выходного сообщения определяется индивидуальным алгоритмом ЦВМ.

Квантовые действия по отношению к элементу создаваемого выходного сообщения $S_{i,j}^{u}$ бывают двух видов: 1) элемент не формируется, так как еще не получены все входные сообщения, элементы которых необходимы для его формирования; 2) элемент выходного сообщения формируется по правилам, предусмотренным индивидуальным алгоритмом: переносится из элемента входного сообщения или вычисляется по определенным правилам из элементов нескольких входных сообщений, при этом вычисленное значение может заноситься в некоторый элемент вектора СВИС.

В одном и том же кванте некоторая ЦВМ может участвовать в разных путях посылки сообщений, поэтому над разными входными сообщениями, а также над разными элементами одного и того же входного сообщения в одном и том же кванте ею могут выполняться квантовые действия разных видов. Это определяется индивидуальным алгоритмом каждой ЦВМ.

В процессе работы параллельного алгоритма над элементами всех выходных сообщений выполняется алгоритм связывания, при котором для каждого элемента выходного контейнера определяются элементы входных контейнеров, участвующие в формировании данного элемента, и квантовые действия, которые при этом должны быть выполнены, например передача согласуемого значения из ЦВМ *i* в ЦВМ *j*, *k*, ... : $\Xi_{i,j}^{t+dt} = \{\sigma_{p,i,j}, \sigma_{q,i,j}, \sigma_{s,i,j}, ..., \sigma_{i,j}\}$, $\Xi_{i,k}^{t+dt} = \{\sigma_{p,i,k}, \sigma_{q,i,k}, \sigma_{s,i,k}, ..., \sigma_{i,k}\}$, ... , где *dt* – приращение по времени, составляющее 1 или более квантов. Кроме того, определяются номера квантов, в которых сформированные выходные сообщения должны быть переданы из ЦВМ-передатчиков и приняты в ЦВМ-приемниках.

В определенных квантах в некоторых или всех ЦВМ системы определяются действия по вычислению элементов ВСЗК или ВСЗС. Работа алгоритма связывания завершается тогда, когда определены квантовые действия по формированию всех элементов ВСЗС индивидуального алгоритма СВИС для каждой ЦВМ системы, который должен выполнять только действия, предписанные алгоритмом связывания, над элементами сформированных выходных и входных контейнеров.

По результатам работы алгоритма связывания для каждой ЦВМ системы строится индивидуальный алгоритм выполнения СВИС, по которому вычисляется ВСЗС в данной ЦВМ. ВСЗС, сформированный в каждой *i*-й исправной ЦВМ системы, является выходным параметром процедуры процесса СВИС в каждой ЦВМ системы и имеет вид $\Omega_i = \{\Theta_1, \Theta_2, ..., \Theta_k, ...\}$, где Θ_k – согласованное значение *k*-й ЦВМ, соответствующее согласуемому значению этой ЦВМ σ_k.

Результаты процесса связывания формируются в виде набора индивидуальных алгоритмов для каждой ЦВМ системы, представленных на специальном объектно-ориентированном языке. Затем каждый такой алгоритм транслируется в исполняемый код на машинном языке соответствующей ЦВМ и переносится в эту ЦВМ в виде вызываемой процедуры.

По времени целевая работа системы организуется на основе создаваемой подсистемы единого системного времени. В соответствии с такой организацией время работы системы может делиться на этапы, циклы, фазы, периоды, раунды, кванты [5]. При необходимости выполнения СВИС все ЦВМ системы в один и тот же момент времени вызывают собственную процедуру процесса СВИС и по этому вызову каждая ЦВМ системы переходит к выполнению собственного индивидуального алгоритма СВИС. Организация распределенного алгоритма СВИС в системе, взаимодействие индивидуальных алгоритмов отдельных ЦВМ, основанное на результатах выделения в сети кластеров и сред межкластерного обмена, гарантируют,

что при наличии допустимых неисправностей во всех кластерах и средах межкластерных обменов в один и тот же момент времени (квант) завершится процесс СВИС такой, что в каждой исправной ЦВМ будет сформирован одинаковый вектор СВИС.

Заключение

В работе рассматривается задача создания алгоритма А5 по построению в каждой ЦВМ индивидуального алгоритма СВИС с учетом структуры системы, совокупность которых составляет распределенный параллельный алгоритм СВИС в многокластерной вычислительной системе. Синхронизированное на основе подсистемы общесистемного времени выполнение этих индивидуальных взаимодействующих алгоритмов гарантирует, что при наличии допустимых совокупностей враждебных неисправностей во всех кластерах и средах межкластерных обменов в один и тот же момент времени (квант) завершится процесс СВИС такой, что в каждой исправной ЦВМ будет сформирован одинаковый правильный вектор СВИС.

Представленные в работе подходы, методы и алгоритмы применимы при организации вычислений с заданной достоверностью в grid- и «облачных» вычислениях и будут также полезными при проектировании обслуживаемых и необслуживаемых распределенных вычислительных и управляющих систем ответственного применения.

Список литературы

1. Лобанов А.В. Модели замкнутых многомашинных вычислительных систем со сбое- и отказоустойчивостью на основе репликации задач в условиях возникновения враждебных неисправностей //Автоматика и телемеханика. – 2009. – № 2. – С. 171–189.

2. Ашарина И.В., Лобанов А.В. Выделение комплексов, обеспечивающих достаточные структурные условия системного взаимного информационного согласования в многокомплексных системах //Автоматика и телемеханика. – 2014. – № 6. – С. 115–131.

3. Ашарина И.В., Лобанов А.В. Выделение структурной среды системного взаимного информационного согласования в многокомплексных системах // Автоматика и телемеханика. – 2014. – № 8. – С. 146–156.

4. Нефедов В.Н., Осипова В.А. Курс дискретной математики: учеб. пособие. – М.:Изд-во МАИ, 1992. – 264 с.

5. Гришин В.Ю., Лобанов А.В., Сиренко В.Г. Взаимное информационное согласование в многомашинных вычислительных системах с обнаружением и идентификацией кратных враждебных неисправностей // Автоматика и телемеханика. – 2003. – № 4. – С. 123–135.

К.А. Баркалов

Нижегородский государственный университет им. Н.И. Лобачевского

ОБ ОДНОМ КЛАССЕ ТЕСТОВЫХ ЗАДАЧ ДЛЯ ПАРАЛЛЕЛЬНЫХ МЕТОДОВ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

Рассматривается проблема построения класса тестовых задач для алгоритмов глобальной оптимизации. Предложен подход к генерации многомерных многоэкстремальных задач, в которых можно наглядно наблюдать характер лучшего текущего приближения вне зависимости от размерности задачи. Генератор был апробирован при построении и последующем решении 100 задач параллельным индексным алгоритмом глобальной оптимизации. Приведены результаты работы алгоритма с использованием разного числа вычислительных ядер, наглядно демонстрирующие его ускорение и безызбыточность.

Введение

Один из известных подходов к исследованию и сравнению параллельных алгоритмов оптимизации основан на применении этих методов для решения множества тестовых задач, выбираемых случайным образом из некоторого сконструированного класса. При этом каждая тестовая задача может рассматриваться как конкретная реализация случайной функции, задаваемой с помощью специального генератора. Применение алгоритмов параллельной многоэкстремальной оптимизации на больших выборках таких реализаций позволяет оценивать операционные характеристики методов (вероятности корректного определения глобального экстремума за заданное число шагов поиска) и тем самым характеризовать эффективность каждого конкретного алгоритма.

Известные генераторы (например, представленные в работе [1]) порождают непосредственно саму оптимизируемую функцию. При этом в случае задач, размерность которых более 2, не удается наблюдать сам процесс оптимизации. В связи с этим представляет интерес несколько иной подход, инициированный в работе [2]. Согласно этому подходу оптимизируемая функция возникает как решение некоторой содержательной задачи (задачи аппроксимации), что позволяет независимо от числа используемых параметров наблюдать характер лучшего текущего приближения (а также окончательного результата). При этом усложнение постановки задачи (включающее также введение невыпуклых ограничений) ведет к увеличению ее размерности. Фактически предлагаемый генератор порождает некоторую задачу аппроксимации, с которой связана оптимизируемая функция.

Генерация серии задач

Рассмотрим математическую модель движения заряженной частицы вдоль оси u в виде

$$m\ddot{u} = -eu + F, \tag{1}$$

где m – масса частицы, $m > 0$; $u(t)$ – текущее положение частицы в момент времени $t \geq 0$, $-eu$ – сила притяжения, действующая на частицу; F – внешняя сила, приложенная к частице вдоль оси u (управляющее воздействие). Предполагается, что управляющее воздействие F является функцией времени и задается в виде

$$F = m\left(A_0 + \sum_{i=1}^{n} A_i \sin(\omega_i t + \varphi_i) \right).$$

Здесь n определяет число частот в управляющем воздействии и является неотрицательным, т.е. управляющее воздействие может быть просто константным.

Заменой $\omega_0^2 = e/m$ уравнение (1) сводится к известному уравнению вынужденных колебаний

$$\ddot{u} + \omega_0^2 u = A_0 + \sum_{i=1}^{n} A_i \sin(\omega_i t + \varphi_i). \tag{2}$$

Ставится задача подобрать частоту, управляющее воздействие и начальные условия следующим образом:

– при $t \in [a,b]$ частица отклонялась бы от заданного положения q_0 не более чем на $\delta > 0$;

– при $t = t_1, t_2, t_3$ частица отклонялась бы от положений q_1, q_2, q_3 соответственно не более чем на $\delta > 0$;

– при $t = t_3$ скорость частицы была бы максимальной.

Данная постановка задачи имеет следующую интерпретацию (рис. 1): траектория движения частицы $u(t)$ должна пройти в «трубке», затем – в трех «окнах», причем в последнем из «окон» она должна иметь максимальный наклон.

Используя запись решения уравнения (2) в виде

$$u(t, \omega, c) = c_0 + \sum_{i=0}^{n} \left[c_{2i+1} \sin(\omega_i t) + c_{2i+2} \cos(\omega_i t) \right],$$

где $c = (c_0, c_1, ..., c_{2n+2})$, $\omega = (\omega_0, \omega_1, ..., \omega_n)$, можно представить исходную задачу оптимального выбора как задачу максимизации по параметрам c и ω при наличии ограничений:

$$|u_t'(t_3,\omega,c)|\to\max;$$

$$|u(t_i,\omega,c)-q_i|\le\delta,\ i=1,2,3; \qquad (3)$$

$$|u(t,\omega,c)-q_0|\le\delta,\ t\in[a,b].$$

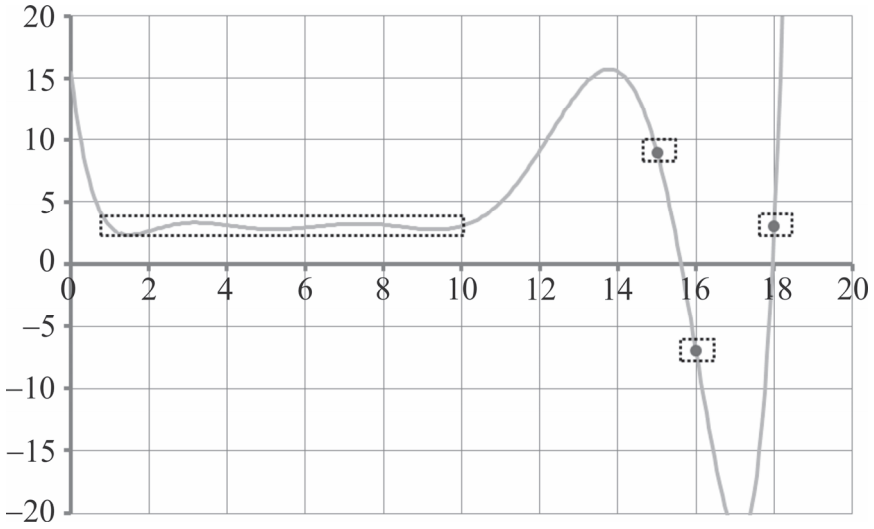


Рис. 1. Траектория движения частицы

Параметром, определяющим конкретный вид задачи, будет расположение центров «окон» и «трубки» q_i, через которые должна пройти траектория частицы. Варьируя q_i (например, равномерно в диапазоне от -10 до 10), можно получать различные постановки оптимизационной задачи.

Параллельный алгоритм глобального поиска

Для исследования данного класса задач оптмизации будем применять параллельный метод глобального поиска, разработанный в ННГУ им. Н.И. Лобачевского. Основная идея метода состоит в следующем.

Запишем задачу оптимизации в общем виде:

$$\varphi(y^*)=\min\{\varphi(y): y\in D,\ g_j(y)\le 0,\ 1\le j\le m\};$$

$$D=\{y\in R^N: a_i\le y_i\le b_i,\ 1\le i\le N\}.$$

Тогда, используя развертки типа кривой Пеано $y(x)$, однозначно отображающие отрезок [0,1] на N-мерный гиперкуб D:

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

исходную задачу можно редуцировать к одномерной задаче:

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

что позволяет применить для ее решения эффективные параллельные алгоритмы одномерной оптимизации. Учет ограничений задачи осуществляется по специальной индексной схеме. Ее характерной чертой является раздельная обработка каждого из ограничений задачи, штрафные функции не используются. В соответствии с индексной схемой каждая итерация метода включает последовательную проверку выполнимости ограничений задачи в этой точке (данная процедура называется в дальнейшем испытанием), а обнаружение первого нарушенного ограничения прерывает испытание и инициирует переход к точке следующей итерации [3]. Распараллеливание алгоритмов может быть осуществлено на основе характеристической схемы, описанной в работе [4].

Результаты экспериментов

С использованием параллельного алгоритма глобального поиска было решено 100 задач вида (3). На рис. 2 приведена операционная характеристика алгоритма.

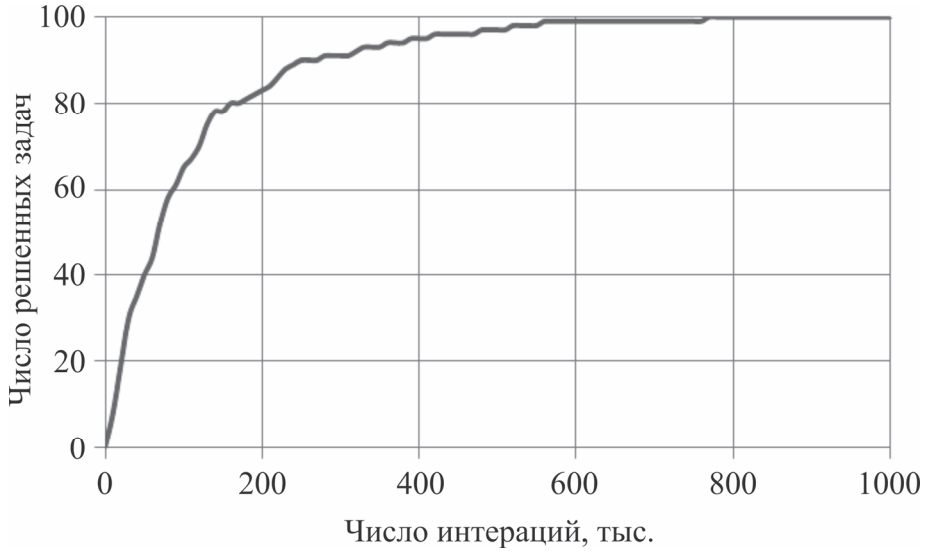


Рис. 2. Операционная характеристика алгоритма глобального поиска

Данная кривая показывает, что для решения любой из задач данного класса потребовалось менее $8 \cdot 10^5$ итераций. При этом более 80 % задач было решено всего за $2 \cdot 10^5$ итераций. Для сравнения: оценка оптимума с той же точностью методом перебора по равномерной сетке потребовала бы проведения порядка 10^8 итераций.

Список литературы

1. Квасов Д.Е., Сергеев Я.Д. Исследование методов глобальной оптимизации при помощи генератора классов тестовых функций. – Н. Новгород: Изд-во ННГУ, 2011.

2. Гришагин В.А., Стронгин Р.Г. Оптимизация многоэкстремальных функций при монотонно унимодальных ограничениях // Известия АН СССР. Техническая кибернетика. – 1984. – № 4. – С. 203–208.

3. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. – Dordrecht: Kluwer Academic Publishers, 2000.

4. Параллельные вычисления в задачах глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов. – М.: Изд-во МГУ, 2012.

Ю.Г. Бартенев[1,2], В.А. Ерзунов[1], А.П. Карпов[1], Д.А. Петров[1], И.А. Пищулин[1], А.Н. Стаканов[1], Е.Б. Щаникова[1], И.Е. Капорин[3], О.Ю. Милюкова[4], С.А. Харченко[3,5], И.Н. Коньшин[3,5], И.Б. Мееров[2], А.В. Сысоев[2]

[1]ФГУП РФЯЦ-ВНИИЭФ, Саров
[2]Нижегородский государственный университет им. Н.И. Лобачевского,
[3]Вычислительный центр им. А.А. Дородницына РАН, Москва
[4]Институт прикладной математики им. М.В. Келдыша РАН, Москва
[5]ОАО «ТЕСИС», Москва

КОМПЛЕКС БИБЛИОТЕК ПАРАЛЛЕЛЬНЫХ РЕШАТЕЛЕЙ СЛАУ LPARSOL ВЕРСИИ 3

Приводится описание библиотеки решателей СЛАУ в виде комплекса библиотек LPARSOL. Приводятся свойства, объединяющие оболочки. Отмечены относительные особенности каждой библиотеки. Проводится сравнение времени решения прочностной задачи разными методами.

В отчете [1] американских специалистов обращено внимание на важность продолжения инвестиций в библиотеки решателей (PETSc, HYPRE, Trilinos, SuperLU и др.) и исследований по преодолению ограничений архитектуры компьютеров, что перекликается с исследованиями в России [2–5].

В 2010–2013 гг. ряд предприятий совместно разработали библиотеку решателей СЛАУ в виде комплекса библиотек LParSol [5], который включает библиотеки PMLP/ParSol [5, 10], TParFBSS [3, 5], VC_RAN_SLAU [4,11] с единым пользовательским интерфейсом и средства подключения Intel PARDISO MKL [6], BoomerAMG HYPRE [7], MUMPS [8], AMG Parallel Toolbox [9] (два последних доработаны [10, 11]).

Объединяющая оболочка LParSol обеспечивает решение потока СЛАУ с одним и разными коммуникаторами MPI. Ниже перечислены ее свойства:

– можно использовать любые решатели, так как формирование СЛАУ от этого не зависит;

– используется внешнее задание решателя и его настроек, поэтому изменения не касаются исходного кода;

– ориентация на поток СЛАУ, что позволяет снизить затраты и повысить адаптивность;

– используются итерационные решатели, предобусловленные в подпространствах Крылова.

В LParSol для распределенных СЛАУ реализованы следующие модули:

– итерационные решатели CG, BiCGStab, CR, BiCRStab, GMRES, SOFGMRES;

– сквозной предобусловливатель IC2/ILU2 [3];

– блочный предобусловливатель (mcILU0, IC0/ILU0, ICk/ILUk, ICt/ILUt);

– блочно-перекрытый предобусловливатель (ICt/ILUt, IC2/ILU2/MILUt, LLt/LU);

– крупноблочно-перекрытый предобусловливатель (IC2/ILU2);

– грубосеточный корректировщик поправки предобусловливания;

– оптимизированный селективный AMG [9,11];

– оптимизированный для блочных матриц LDLt-решатель MUMPS.

В таблице представлено решение СЛАУ P4 из задачи прочности разными методами.

Решение СЛАУ P4 из задачи прочности разными методами

Решатель	*p*	Dens Fct	N iter	Fct Time	Iter Time	Total Time
PARDISO*Intel® MKL	12	~20.	0	–	–	430.4
MUMPS+ParMetis	64	~20.	20	118.3	1.6	119.9
PMLP_Schwartz(PARDISO)	192	< 20.	85	15.3	106.2	121.5
VC_RAN_Schwartz(IC2)	385	4.3	6965	4.1	105.2	109.3
TFParBSS(IC2_float)	384	8.9	447	80.0	34.	114.

Особенностями LParSol являются следующие:

– единый формат СЛАУ и файл параметров (приложение не зависит от решателя);

– одинарная и двойная точность предобусловливания (IC/ILU, IC2/ILU2) [3, 5];

– точечный и блочный разреженный формат матриц в ряде решателей [3, 10];

– оптимизирующие переупорядочивания матрицы (RCM, ND, др.);

– наследование построенных структур решателя предыдущей СЛАУ;

– адаптивный выбор для потока СЛАУ наискорейшего решателя [12].

Отраженные выше возможности опробованы в ряде приложений математического моделирования, разработанные в РФЯЦ-ВНИИЭФ, ряд из них находится в постоянной эксплуатации при проведении расчетов на многопроцессорных ЭВМ.

Управляющая оболочка LParSol позволяет использовать в одном приложении решатели разных библиотек, обеспечивает адаптивный выбор метода решения или параметров из заданного пользователем набора, обеспечивает параметризацию решателя в оперативной памяти и в файле.

Ряд методов в библиотеках текущего комплекса LParSol пересекается. Отметим наиболее отличительные особенности каждой библиотеки:

– в PMLP/ParSol имеются MPI-параллельные: предобусловливатель Шварца с Coarse Grid Correction и селективный предобусловливатель AMG; мультипоточные предобусловливатели двойной и одинарной точности: ILU0- и IC0-разложения точечных и мелкоблочных матриц и блочные ILUt- и ICt-разложения точечных матриц с отсечением значений по величине;

– в TParFBSS имеются сквозной и крупно-блочный с наложением MPI-мультипоточные предобусловливатели двойной и одинарной точности IC2- и ILU2-разложения точечных и мелкоблочных матриц второго порядка точности с отсечением значений по структуре и величине, ориентированные на решение потока СЛАУ с большой динамикой изменения структуры матрицы;

– VC_RAN_SLAU имеет блочные с наложением предобусловливатели двойной точности: BIIC-IC2 – разложение точечных SPD-матриц второго порядка точности с отсечением значением по величине [4] и BIILU-MLILU – разложение точечных несимметричных матриц с отсечением значений по величине и выбором ведущего элемента в блоках [13];

– модифицированный MPI-мультипоточный прямой LDL^t-решатель библиотеки MUMPS для точечных и блочных SPD-матриц.

К комплексу LParSol примыкают инструментальные средства тестирования и исследования матриц; параллельный спектральный решатель VC_RAN_SLAU для нахождения собственных пар SPD-матрицы [14] и библиотека оптимизации и декомпозиции графов ParMATRUZ, разработанная в ННГУ [5, 15].

Ряд методов, демонстрирующих либо лучшее качество предобусловливания, либо более высокий параллелизм, либо то и другое: AMG, коррекция на грубой сетке, многоцветные ILU, прямой мультипоточный LLt-решатель, полиномиальный предобусловливатель [2], находится в стадии испытаний.

Список литературы

1. Dongarra J., Hittinger J., BellExascale J. Mathematics Working Group // Applied Mathematics Research for Exascale Computing: LLNL TR-651000. – 2014.

2. Капорин И.Е. Дискретный метод наименьших квадратов для полиномиального предобусловливания несимметричных матриц // Оптимизация и приложения: сб. тр. отд. проблем приклад. оптимизации ВЦ РАН / под ред. В.Г. Жадана. – М.: Изд-во ВЦ РАН, 2013. – Вып. 3. – С. 120–139.

3. Коньшин И.Н., Сушко Г.Б., Харченко С.А. Сквозной параллельный алгоритм построения неполного треугольного разложения второго порядка точности с динамическим выбором декомпозиции и упорядочивания // Супервычисления и математическое моделирование: тез. докл. XIV Междунар. конф. – Саров, 2012. – С. 110–111.

4. Капорин И.Е., Милюкова О.Ю. Массивно-параллельный алгоритм предобусловленного метода сопряженных градиентов для численного решения систем линейных алгебраических уравнений // Оптимизация и приложения: сб. тр. отд. проблем приклад. оптимизации ВЦ РАН / под ред. В.Г. Жадана. – М.: Из-во ВЦ РАН, 2011. – С. 32–49.

5. Комплекс LParSol для решения СЛАУ / Ю.Г. Бартенев [и др.] // Супервычисления и математическое моделирование: тез. докл. XIII Междунар. сем. – Саров, 2011. – С. 34–36.

6. Intel Math Kernel Library Reference Manual. – URL: http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf.

7. Henson V.E., Yang U.M. BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner // Applied Numerical Mathematics. – 2002. – № 41. – P. 155–177.

8. 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.

9. Liebmann M. Algebraic Multigrid Methods on GPU-Accelerated Hybrid Architectures / Institute for Mathematics and Scientific Computing; University of Graz. – 2011.

10. Оптимизация и применение пакета MUMPS для решения трехмерных стационарных задач прочности на кластерных системах / С.А. Лебедев [и др.] // Вестник УГАТУ. – 2014. – Т. 18, № 3 (64).

11. Адаптация некоторых решателей СЛАУ для арифметических ускорителей NVIDIA / Ю.Г. Бартенев, А.П. Карпов, И.А. Крючков, И.А. Пищулин // Супервычисления и математическое моделирование: тез. докл. XIV Межд. конф. – Саров, 2012. – С. 30–32.

12. Ерзунов В.А., Горбунов А.А. Механизм адаптивного выбора решателя в библиотеке PMLP/Parsol // Вопросы атомной науки и техники. Математическое моделирование физических процессов. – 2009. – Вып. 1. – С. 55–62.

13. Multilevel ILU Preconditioning for Iterative Solution of General Sparse Linear Systems / I. Kaporin, O. Diyankov, S. Diyankova, V. Pravilnikov // Book of abstracts: Int. Conf. on Preconditioning of Iterative Methods – Theory and Applications (PIM-2013), 1–5 July. – Prague, 2013. – P. 30–31.

14. Капорин И.Е., Милюкова О.Ю., Бартенев Ю.Г. Решение сверхбольших спектральных задач с разреженными матрицами на современных суперкомпьютерах // Управление и прикладная математика: тр. 56-й науч. конф. МФТИ, 25–30 ноября 2013 года. – М.: Изд-во МФТИ, 2013. – Т. 1. – С. 13–14.

15. Бартенев Ю.Г., Старостин Н.В., Филимонов А.В. Многоуровневый алгоритм уменьшения ширины ленты симметрической распределенной матрицы // Системы управления и информационные технологии. – 2014.

Д.С. Батищев, В.М. Михелев

Белгородский государственный национальный
исследовательский университет

ИНФРАСТРУКТУРА ОБЛАЧНОЙ СИСТЕМЫ ДЛЯ ОБРАБОТКИ МЕДИЦИНСКИХ ДАННЫХ

Широкое распространение цифровых снимков в медицине дает возможность их анализа и обработки. Но сначала надо накопить достаточный объем. Медицинские снимки широко варьируются по размерам, и можно использовать различные технологии для их хранения. Одной из таких технологий является облачное хранение данных. Гибкое, легко масштабируемое решение. Выбор сделан в пользу OpenStack. И результатом является описание полученной платформы с учетом предметной области.

В настоящее время информационные технологии все интенсивнее развиваются и все глубже проникают в повседневную жизнь обычных людей. Это касается практически всех аспектов современного быта – за телефон и квартиру можно заплатить в Интернете, купить нужные товары также можно, не отходя от компьютера, появилось понятие «интернет-вещи». Все это стало возможным за счет появления цифровых аналогов привычных в реальности вещей.

Не исключение и медицина. Практически подавляющее большинство сканирующих аппаратов на сегодняшний момент являются цифровыми и предоставляют результаты сканирования в виде файла в определенном формате. Поскольку это такие же цифровые данные, как и музыкальная композиция или цифровой снимок, выходные файлы могут быть подвергнуты компьютерному анализу и обработке. Для обработки какой-либо информации ее нужно накопить, придумать возможные схемы или методики ее хранения – это и является целью данной работы. Проблема накопления особенно актуальна из-за большого количества и объема необходимых для хранения данных. Например, рентген-снимок может занимать до 15 Мб для одного паци-

ента, а модель СКТ до 1 Гб. С учетом количества проходящих пациентов эту проблему можно косвенно классифицировать как Big data.

Традиционно все медицинские снимки какое-то время хранятся в управляющем компьютере рентгеновского аппарата, томографа. Безусловно, данный способ прост и доступен всем: жесткие диски растут в объеме, и в обычном персональном компьютере можно собрать внушительный массив. Однако этот способ плох тем, что данные находятся в одном месте, а значит, увеличивается риск их потери.

Частично решить проблему аккумуляции данных в одном месте с использованием нескольких компьютеров могут кластерные файловые системы – вид файловых систем, хранение на которых производится прозрачно на нескольких узлах с дублированием данных и многопоточной выгрузкой. Это удобно.

Еще удобнее – облачное окружение. Гибкость виртуализированного окружения, масштабируемость и надежность кластерных файловых систем вкупе с простотой добавления новых вычислительных узлов – те основополагающие факторы, сыгравшие ключевую роль в выборе технологии для решения поставленной задачи. Также с учетом виртуализации удобно реализовывать единую точку входа для пользователей: когда пул виртуальных машин с бэкенд-приложением физически расположен за парой реверс-прокси узлов – безопасно и просто администрировать. Таким образом, концептуальная модель облачного сервиса представлена на рис. 1.

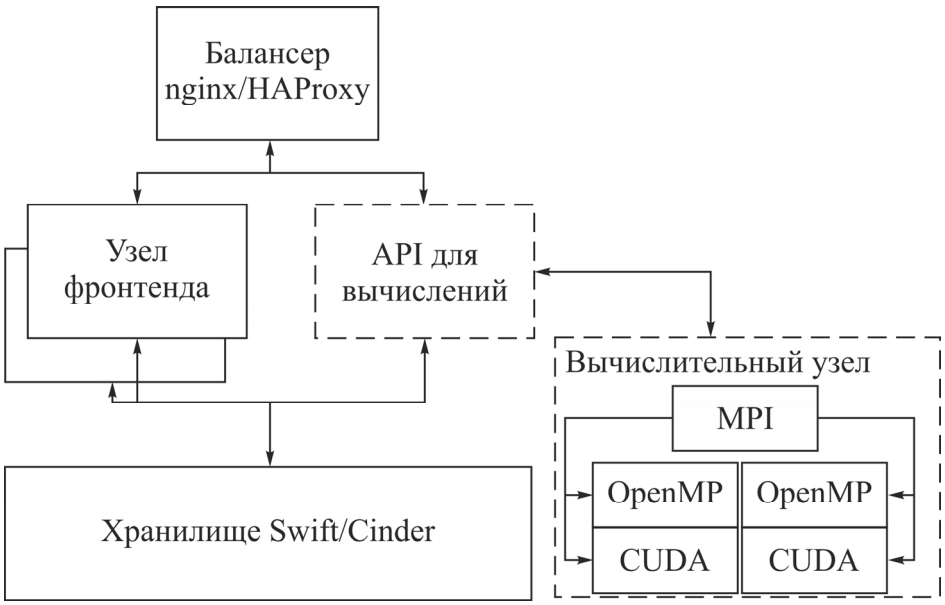


Рис. 1. Концептуальная модель облачного сервиса

Программных продуктов для реализации частного облачного окружения много. OpenStack – один из самых поддерживаемых, используемых, гибких и развиваемых. Это повлияло на выбор OpenStack в качестве плат-

формы. OpenStack имеет модульную архитектуру, а в силу ограниченности в аппаратных ресурсах на момент реализации прототипа информационной системы это оказалось весьма удобно. Результатом стало рабочее частное облачное окружение, развернутое на двух физических компьютерах. Один из них играл роль управляющего узла с установленными сервисами хранения образов виртуальных машин, образов установочных дисков, контроллера сетевых интерфейсов, а также системной СУБД MySQL и брокера сообщений RabbitMQ, которые используются для связи всех компонентов между собой. Второй же физический узел был выделен полностью под гипервизор. Данное решение впоследствии позволяет буквально в несколько команд добавить узлы-гипервизоры, не трогая при этом управляющий узел. А ведь гипервизор – основа виртуализации в облаках. Реализованная архитектура представлена на рис. 2.

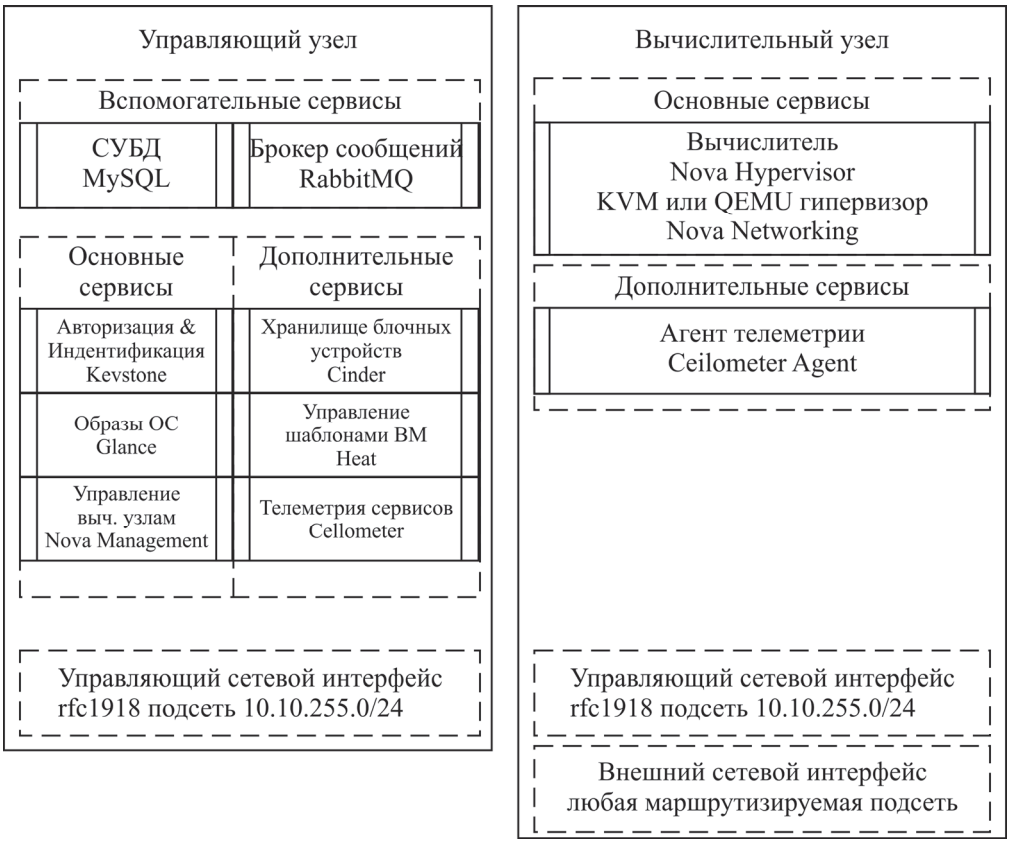


Рис. 2. OpenStack-реализации архитектуры

Таким образом, представленная архитектура еще не является законченным облачным сервисом для обработки медицинских данных. Для завершенности не хватает описанного бэкенд-приложения, которое реализует пользовательский интерфейс и диалоги регистрации, загрузки и выгрузки снимков. В рамках данной статьи представляемое приложение также носит характер прототипа для исследования возможности использования облаков для хранения снимков и предоставления доступа к ним.

Поскольку OpenStack по большей части написан на языке Python, веб-фреймворк для реализации веб-приложения тоже был выбран на этом языке. Предполагается, что в дальнейшем это позволит глубже и проще интегрировать веб-сервис непосредственно с облачным окружением. Django по праву можно считать одним из самых распространенных инструментариев для разработки динамических веб-страниц. Вся разработка строится на основе трех сущностей: модель, представление, шаблон.

Разработанное веб-приложение было запущено на двух веб-серверах с одним фронтенд-балансером, каждый веб-сервер мог получать данные из пула узлов хранения. Подробнее общая схема представлена на рис. 3.

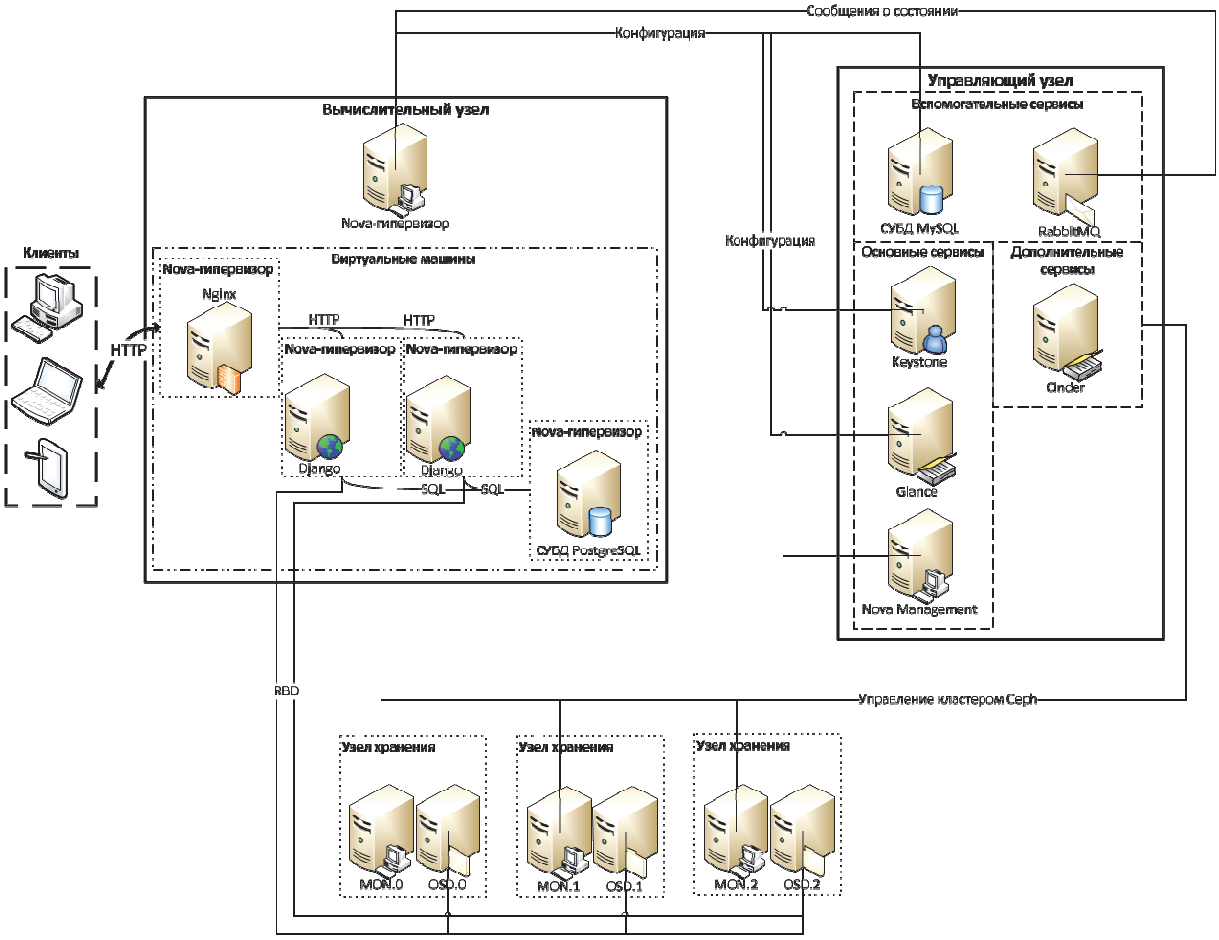


Рис. 3. Общая архитектура облачного сервиса

В рамках данной статьи проведено исследование возможности использования частного облачного окружения для решения задачи хранения снимков и предоставления доступа к ним. Результатом является рабочий прототип облачного окружения, системы хранения данных и веб-приложения. Как развитие можно рассматривать разворачивание наработок на нескольких физических узлах, переход к более совершенному

встроенному объектному хранилищу Swift, реализацию и интеграцию метода обработки медицинских снимков для достижения конечной цели – анализа медицинских данных.

Список литературы

1. Medical Image Analysis on Grids / C. Germain [et. al]. – URL: http://hal.archives-ouvertes.fr/docs/00/17/42/84/PDF/AGIRJCMCALL.PDF.

2. Foster, Ian, Kesselman Carl. The Grid: Blueprint for a New Computing Infrastructure Second Edition. – Morgan Kaufmann Publishers, 2004. – P. 31–32.

3. О возможностях параллельной обработки данных в ГРИД-системе хранения медицинских изображений для повышения эффективности диагностики / С.В. Баранник [и др.] // Кластерные вычисления: электрон. сб. тр. второй междунар. конф., 3–5 июня. – Украина, Львов, 2013. – С. 22–25.

4. Батищев Д.С., Петров Д.В., Михелев В.М. Использование облачных ресурсов для высокопроизводительных процессорных вычислений // Теория и практика системного анализа: сб. тр. Всерос. молодеж. конф., 1–3 октября 2012 г. – Белгород: Белгород, 2012. – С. 16–17.

А.Т. Бектемесов, Д.Ж. Ахмед-Заки

Казахский национальный университет им. аль-Фараби, Алма-Ата

РАСПРЕДЕЛЕНИЕ И ВЕРИФИКАЦИЯ МОДЕЛЕЙ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА БАЗЕ MPI

Параллельные программы очень сложны с точки зрения их организации и распределения. При тестировании распределенного алгоритма по узлам кластерных систем не проверяются все варианты выполнения, так как, оно зависит от сложности поставленной задачи. Сложностью поставленной задачи является ее распределение в мультикомпьютерах с некоторым числом процессорного элемента с различной работоспособностью и коммутативной сетью. Предлагается проверка корректности коллективного выполнения технологий MPI версии 3.0. Показаны недостатки выполнения распределения коллективных операций для некоторых задач. Верификация осуществляется с помощью пакета Spin.

Введение

В мире программирования существует проблема устойчивости и корректности. Разработчики алгоритмов и программных продуктов должны гарантировать хотя бы на 90 % корректность работы параллельной программы. Тестирование и верифицирование параллельных программ очень сложны, так как выполнение отдельной программы одно, а правильная рас-

пределенная программа требует тщательного анализа и строгой взаимосвязи процессоров. Без корректного распределения параллельных программ среди вычислительных аппаратов невозможно получить высокой производительности для больших задач.

Поведение параллельной программы, имеющей несколько процессов и выполняющейся на нескольких вычислительных устройствах, становится непредсказуемым. В таких случаях для проверки поведения программы может использоваться метод трассировки.

Одним из самых перспективных методов проверки программ и аппаратов является совокупность верификации и тестирования – модельная проверка ПО (ModelChecking-MC) [1].

В работе продемонстрирован метод проверки эффективности распределения параллельной программы в мультикомпьютерах. Инструментом параллелизации является функция коллективных операций технологии MPI3.0 [2].

Для достижения хорошего качества параллельная программа должна обладать рядом полезных свойств [3].

Свойство 1. Взаимодействующие процессы должны отображаться в один и тот же ПЭ либо в соседние ПЭ (связанные физическими линками).

Свойство 2. Все ПЭ должны быть загружены работой примерно одинаково, одновременно начинать работу и одновременно ее завершать.

Свойство 3. Каждый ПЭ должен иметь достаточное количество ресурсов, необходимых для исполнения всех назначенных на него процессов.

В качестве задачи для вычисления применяется умножение вектора на матрицу с использованием статических методов конструирования отображения модели в системах параллельного программирования:

$$M = \begin{matrix} b_{00} & b_{01}... & b_{0n} \\ b_{10} & b_{11}... & b_{1n} \\ b_{n0} & b_{n1}... & b_{nn} \end{matrix} \qquad V = \{a_0, a_1, a_2, ...$$

Отсюда – умножение:

$$P = M \times V = \begin{matrix} a_0 b_{00} + & a_1 b_{01} + & a_n \\ a_0 b_{10} + & a_1 b_{11} + & a_n \\ ... & ... & \\ a_0 b_{n0} + & a_1 b_{n1} + & a_n \end{matrix}$$

Параллельная программа состоит из N процессов, независимо выполняющих операции умножения вектора на матрицу. Нулевой процесс распределяет нагрузку по процессам в неоднородной среде. После некоторой

итерации главный процесс начинает собирать все обработанные задачи и возвращает результат для каждого процесса. В алгоритме не предусмотрен механизм распределения и планирования по нагрузке или свойствам вычисляющих узлов. Процессы выполняются на одном или нескольких вычислительных устройствах.

Псевдокод алгоритма распределенного вычисления для задачи умножения вектора на матрицу при количестве *n* процессов:

```
Var
matrix vv[n];
vector[n];

procedure multi(int id)
BEGIN
if ( id == root ) ->
        for do ( iter < n*n ) ->
        /*----------send order----------------*/
        send_to_proc(Order);
        recv_from_proc(s1,s2);
        sum[q] = sum[q] + s1 + s2;
                if (iter == k*nn-1) ->
send_to_proc(sum[q]); send_to_proc(sum[q]); q++; k=k+1;
                else -> skip;
                end if; iter++;
  else -> break; end do;
else -> j = id;
        for do ( i <= n ) ->
        for do ( j < n ) -> j = j + numprocs;
        recv_from_root(Order); sum1 = vv[j].hh[i] * vector[j];
        send_to_root(sum1); j = j + 2;
        else -> j=0; recv(q1); sumP1[i]=q1; break
        end do;  i++; prop++;
else -> break
end do;
end if;
END.
```

Основной процедурой распределения является функция multi(). Операция взаимосвязи процессов имеет вид send_to_proc(), recv_from_proc(), send_to_root(),recv_from_root(). Функции с метками _proc и _root операций со стороны главного процесса и остальных соответственно. Алгоритм имеет два канала взаимосвязи.

Данный алгоритм применен для распределения между двумя процессами (рис. 1):

Процесс	1(PROC 1)	2(PROC 2)	1(PROC 1)	2(PROC 2)
sum[0]	$a_0b_{00}+$	$a_1b_{01}+$	$a_2b_{02}+$	$a_3b_{03}+$
sum[1]	$a_0b_{10}+$	$a_1b_{11}+$	$a_2b_{12}+$	$a_3b_{13}+$
sum[2]	$a_0b_{20}+$	$a_1b_{21}+$	$a_2b_{22}+$	$a_3b_{23}+$
sum[3]	$a_0b_{30}+$	$a_1b_{31}+$	$a_2b_{32}+$	$a_3b_{33}+$

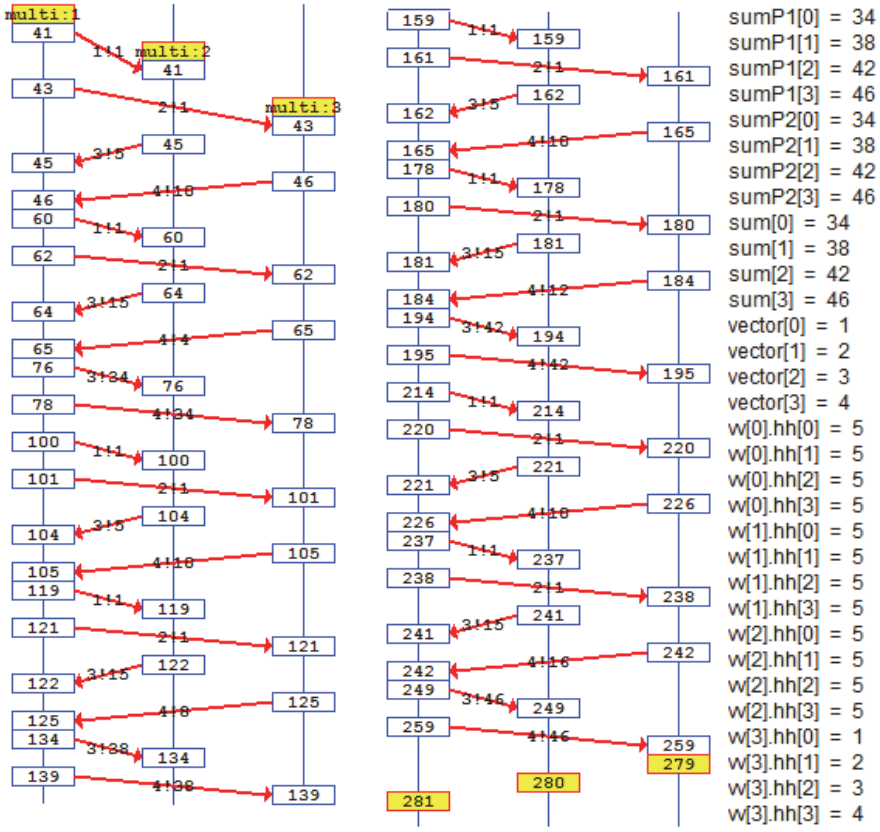


Рис. 1. Алгоритм распределения

В результате главный процесс multi:1 на состоянии 41 делает запрос о готовности для получения нагруженной процессами задачи. После запроса главного процесса остальные 2 процесса multi:2 и multi:3 возвращают нагрузки в состоянии 45 и 46. Входные данные матрицы M вектора V:

$$M = \begin{matrix} 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \end{matrix} \quad \times \quad V = 1\ 2\ 3$$

В результате sum[0] = 34, sum[1] = 38, sum[2] = 42, sum[3] = 4, причем по правилу mpi_allreduce() результаты сохраняются на каждом процессе.

Допустим, что матрица не имеет четного количества элементов по ширине. При этом трудно распределить на нечетное количество процессов, и это является проблемой распределения функции mpi_allreduce(). Если даже разделить ширину на ровное (нечетное) количество процессов, не гарантирована производительность вычислений, так как блокирование работы процессов из-за дополнительной работы одного процесса приводит к затратам времени либо неверным вычислениям задачи в случае отсутствия блокировки работы процессов.

Допустим, в программе есть циклический участок, модель которого является исполнением процессора с ошибкой либо слабо работающих процессов. На языке Spin это будет выглядеть следующим образом:

```
m1:    do ::( prop> 2 ) –>
                    if ::(prop==1000)->break;
                    ::else –> prop++ fi; goto m1;
                    :: else –>break
              od;
```

При верификации на Spin циклические участки не выявляются (рис. 2).

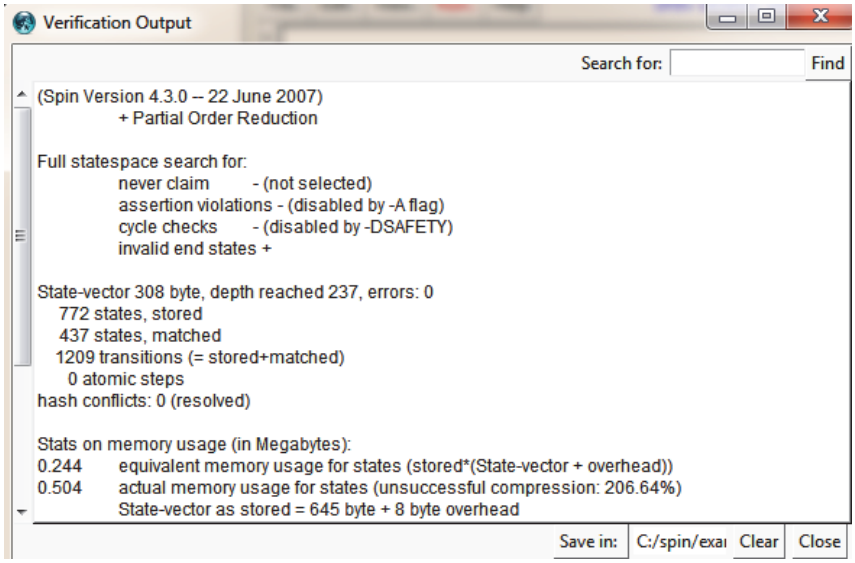


Рис. 2. Верификация алгоритма без выявления ошибки

В методах МС спецификации распределенных систем задаются с помощью формул темпоральной логики [4]. Формулы темпоральной логики LTL описывают временные свойства вычислений модели.

Задана формула спецификации ϕ темпоральной логики относительно переменных модели M. Необходимо проверить выполнимость формулы ϕ на модели M, т.е. $M \models \phi$.

На верификаторе Spin предусмотрена проверка свойств по темпоральной логике LTL. Для проверки по спецификации в код надо внести предикаты. Поскольку для нашего алгоритма время выполнения процессов очень важно, добавлены глобальное время CLOCK, счетчик нулевого процесса aCLOCK1 и локальное время для каждого процесса CLOCK2. Все процессы имеют предикаты PROPERTY1 и PROPERTY2, которые в начале выполнения зарегистрированы как true.

Свойства LTL:

$$\phi = FG\neg property1 \vee FGproperty2 = \\ = FG\neg(\, property1 \vee property2). \qquad (1)$$

После верификации с применением формулы LTL (2) выявилась ошибка в состоянии 664. Это можно считать как неисправность одного процессора либо как наличие низкой мощности (рис. 3). В практике может получиться так, что все процессы закончили выполнение своих задач, а последний процесс, задерживая завершение других процессов, еще долго будет работать. Таким образом, откладывать в конец назначение долго исполняющихся процессов приведет к построению неудачных распределений [3].

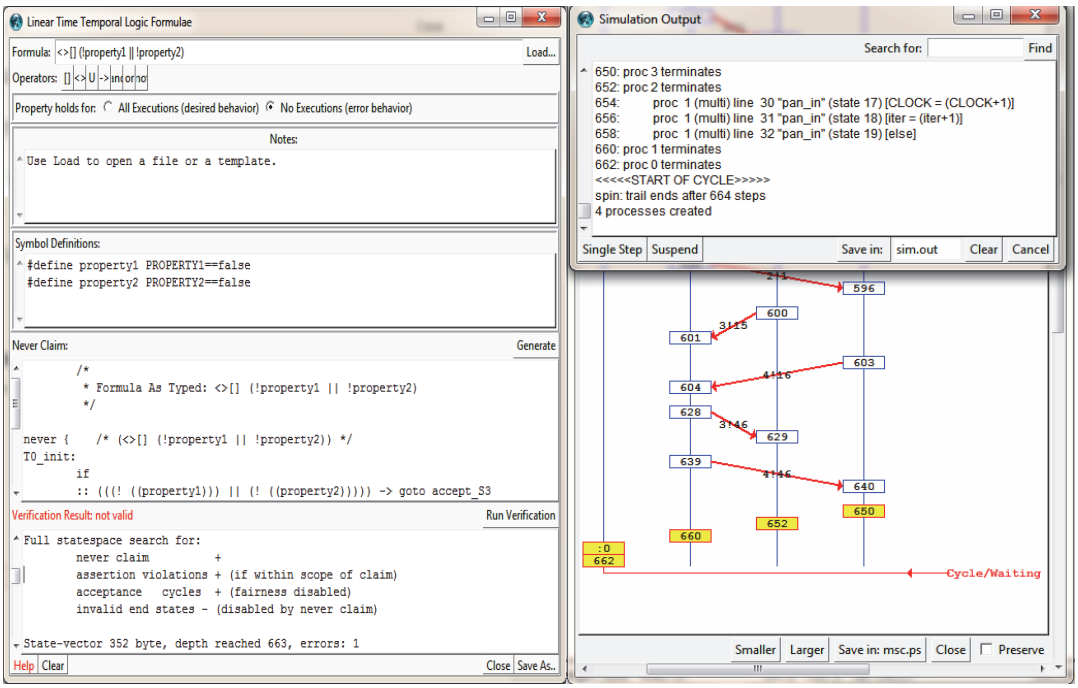


Рис. 3. Верификация с применением LTL-утверждения и выявление ошибки

Заключение

Приведено описание алгоритма построения распределенной системы, и выявлена неисправность для некоторых задач. При коллективном распределении задач по процессам продемонстрирована задержка работы всех

процессов из-за неисправности одного процесса. Метод проверки на модели показал, что такого рода параллельные программы не соответствуют свойству 3 [3], это объясняется тем, что любой неопытный программист, не учитывая свойств каждого процессора, не сможет качественно спроектировать распределение параллельных программ с использованием статических методов.

Список литературы

1. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. – М.: Изд-во Москов. центра непрерыв. математич. образ., 2002.

2. MPIdocuments. – URL: www.mpi-forum.org/docs/mpi-3.0.

3. Малышкин В.Э., Корнеев В.Д. Параллельное программирование мультикомпьютеров (Сер. учебники НГТУ). – Новосибирск: Изд-во НГТУ, 2006. – 296 с.

4. Карпов Ю.Г. ModelChecking: Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010.

В.А. Биллиг

Тверской государственный технический университет

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНСТРУМЕНТАРИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ – OPEN MP И DOT NET

Практически любая современная среда разработки предлагает свой инструментарий организации параллелизма. Какой из них выбрать, если есть такая возможность? Влияет ли этот выбор на эффективность вычислений? Насколько просто и комфортно вести разработку параллельных программ и, что более важно, насколько удобно выполнять отладку этих программ?

Посвящена сравнению инструментария Dot Net и Open MP, реализации этого инструментария в рамках Visual Studio.

Введение

Параллельные вычисления становятся практикой повседневного программирования. Для суперзадач, требующих высокопроизводительных вычислений на суперкомпьютерах, у программистов нет выбора: такие задачи требуют применения параллельных алгоритмов. Для широкого спектра повседневных задач программисту приходится применять для их решения как последовательные, так и параллельные алгоритмы. В зависимо-

сти от сложности задачи, прежде всего от объема обрабатываемых данных, последовательный алгоритм может оказаться эффективнее параллельного, поскольку не требует затрат на организацию параллелизма.

Помимо необходимости сочетания в своей программе реализации как последовательных, так и параллельных алгоритмов, еще одной важной проблемой для программиста является выбор подходящего инструментария для реализации параллелизма.

В отсутствие единой теории параллельных процессов, позволяющей решать многообразные задачи, возникающие при организации параллельных вычислений, существует ряд подходов, имеющих свои достоинства и недостатки.

Скажу прежде всего о двух стандартах – Open MP и MPI. Первый из них предназначен для решения проблем систем с общей памятью. При создании он был ориентирован на язык Fortran, позже на C, C++. Эта ограничительная особенность сохранилась и доныне, с появлением уже четвертой версии стандарта. Стандарт MPI ориентирован на решение проблем обмена сообщениями в распределенных системах. Ни один кластер не обходится без реализации этого стандарта.

В языке Эйфель предложен своеобразный, объектно-ориентированный подход к реализации параллелизма.

Фирма Intel предлагает собственную разработку Parallel Studio, включающую широкий спектр инструментов, предназначенных для разработки параллельных программ, начиная от проектирования до их отладки и анализа.

Технология CUDA от NVidia позволяет эффективно использовать возможности многоядерных графических процессоров этой фирмы для организации параллельных вычислений не только в задачах, связанных с построением графических изображений.

Разнообразный инструментарий предлагает среда разработки Visual Studio фирмы Microsoft.

Инструментарий Dot Net

Когда программа создается на одном из языков Dot Net, то в ней можно использовать библиотеку классов FCL, включающую множество классов, позволяющих организовать параллельные вычисления на разных уровнях абстракции.

Далее в качестве такого языка я буду использовать язык C#, большинство примеров и результаты численных расчетов получены для проектов, написанных на C#.

Классы Dot Net позволяют вести разработку на разных уровнях абстракции. Приведу краткую характеристику основных классов:

– Thread – позволяет работать на нижнем уровне абстракции непосредственно с потоками операционной системы. Дает возможность создавать и удалять потоки, синхронизировать их работу;

– ThreadPool – обеспечивает более высокий уровень абстракции, позволяя работать на более высоком уровне абстракции, непосредственно с пулом потоков операционной системы. Программисту в этом случае не требуется создавать и удалять потоки. Эти функции берет на себя операционная система;

– Task – на этом уровне абстракции программист может «ничего не знать» о таких низкоуровневых понятиях, как потоки. Для него существуют задачи – объекты класса Task, с которыми он и работает, ставя их в очередь на параллельное выполнение;

– Parallel – на сегодняшний день обеспечивает самый высокий уровень абстракции. Методы этого класса – Parallel.For, Parallel.ForEach и Parallel.Invoke – позволяют реализовать параллелизм по данным и параллелизм по задачам соответственно. Для программиста, работающего с этими средствами, уровень абстракции столь же высок, как при работе с обычными операторами цикла for, foreach и оператором вызова метода. Всю заботу о распараллеливании итераций цикла, организации параллельного выполнения вызванных методов берут на себя методы класса Parallel;

– Semaphore, Monitor – реализуют известные концепции семафоров Дейкстры и мониторов Хоара. Они позволяют организовать безопасную работу с совместно используемыми ресурсами, достаточно сложное взаимодействие параллельно исполняемых методов. Оператор языка Lock, позволяющий закрывать критическую секцию, реализует простейшую версию класса Monitor;

– ReaderWriterLockSlim – позволяет реализовать известный паттерн параллельного выполнения, так называемую модель «читатели – писатели».

Это далеко не полный перечень классов, предлагаемых в Dot Net для организации параллельных вычислений.

В стандарте Open MP предлагается свой подход, свой уровень абстракции, также позволяющий эффективно реализовать параллельные вычисления: их организацию, синхронизацию, использование общих ресурсов.

В OpenMP используется модель параллельного выполнения fork – join – «ветвление–слияние».

Программа OpenMP начинается с единственного основного потока выполнения, который в ходе работы может создать параллельную секцию, включающую группу потоков, состоящую из себя и некоторого числа до-

полнительных потоков, оставаясь главным в группе. Все члены новой группы, включая главный, выполняют код внутри параллельной конструкции. В конце параллельной конструкции имеется неявный барьер. После параллельной конструкции выполнение пользовательского кода продолжает только главный поток. В параллельную секцию могут быть вложены другие параллельные секции, в которых каждый поток первоначальной секции становится основным для своей группы потоков. Вложенные секции могут, в свою очередь, включать секции более глубокого уровня вложенности.

Общие переменные, доступные для всех потоков группы, играют роль общих ресурсов.

Управление распараллеливанием поддерживается в OpenMP с помощью директив, основными из которых являются следующие: parallel, for, parallel for, section, sections, single, master, critical.

Директива parallel for – аналог метода Parallel.For; позволяет реализовать параллелизм данных и организовать параллельное выполнение итераций цикла.

Параллелизм задач реализуется за счет введения параллельных секций с помощью директив parallel section.

Секции позволяют задать неявную синхронизацию. Явную синхронизацию можно задать директивой barrier.

Директива critical позволяет задать критические секции.

Директивы single и master позволяют указать, что некоторый код должен выполняться одним потоком – любым (single) или главным (master).

Отличаясь в концепциях и деталях, оба инструментария позволяют достаточно просто организовать параллельные вычисления при решении широкого круга задач (таблица).

Результаты сравнения двух инструментариев по ряду критериев

Критерий	.Net –TPL	OpenMP
Обработка исключений	Присутствует	Только внутри параллельной области
Планирование	Автоматически	Автоматически и вручную
Управление циклом Break, Continue	Поддерживается	Запрещено
Установка параметров (число потоков и др.)	Отсутствует	Поддерживается

Для иллюстрации приведу фрагменты кода, дающие решение известной задачи «Обедающие философы. Вот как выглядит метод, моделирующий обед философов, написанный на C# и использующий класс Parallel:

```
public void DinnerTime()
{
Parallel.For(0, count, num => {DinnerForFhilosophers(num);});
}
```

При программировании на C++ с использованием стандарта Open MP этот метод можно записать следующим образом:

```
void DinnerTime()
{
#pragma omp parallel for
for(int num = 0; num < count; num++)
        DinnerForPhilisopher(num);
}
```

В одном случае используется метод Parallel.For, в другом – директива parallel for. Вызываемый в цикле метод DinnerForPhilisopher(num) моделирует обед философа с номером num. Философы обедают одновременно, соответствующие методы будут выполняться параллельно.

Приведу некоторые фрагменты кода для обоих вариантов из реализации метода DinnerForPhilisopher(num):

Dot Net:

```
void DinnerForFhilosophers(int num)
SemaphoreSlim left_fork = forks[num];
SemaphoreSlim right_fork = forks[(num + 1) % count];
left_fork.Wait();    right_fork.Wait();
left_fork.Release(); right_fork.Release();
```

Open MP:

```
void DinnerForPhilisopher(int num)
omp_lock_t* leftFork = &forks[num];
omp_lock_t* rightFork = &forks[( num + 1) % count];
Wait(think_time); omp_set_lock(leftFork);
Wait(wait_time);  omp_set_lock(rightFork);
omp_unset_lock(leftFork);       omp_unset_lock(rightFork);
```

Общими ресурсами в этой задаче являются вилки. Каждому философу для еды нужны две вилки, поэтому n философам для одновременной еды нужны $2n$ вилок, а их всего n. В результате конкурентной борьбы за общие ресурсы в этой задаче может возникать ситуация клинча.

В модели Dot Net общие ресурсы описываются как объекты класса Semaphore, в Open MP – как замки, lock-объекты.

В целом, как можно видеть, обе программы имеют изоморфную структуру, совпадая с точностью до числа строк. Подход, позволяющий справиться с клинчем, также одинаково просто решается в обоих случаях.

Наиболее интересен вопрос, насколько эффективен инструментарий, насколько хорошо справляются с решением задачи программы, создаваемые инструментарием. Для ответа на этот вопрос в рамках Visual Studio было создано решение, позволяющее сравнивать результаты решений C++ Open MP проектов и C# проектов для широкого класса задач линейной алгебры.

Программный проект, позволяющий проводить такое сравнение, является частью дипломной работы моего дипломника М.Ю. Смирнова [3].

Анализ данных показывает, что параллельные алгоритмы для данной задачи существенно выигрывают у последовательных аналогов, хотя в этой задаче есть последовательная часть, не допускающая распараллеливания. Результаты получены для процессора с четырьмя физическими ядрами. Эффективность, рассчитываемая в таблице, оценивается по отношению к восьми виртуальным ядрам. Если расчеты вести по отношению к физическим ядрам, то она близка к 0,85.

Самое интересное, какая же программа работает быстрее? В данном случае выигрыш за инструментарием Open MP. Программа, написанная на управляемом C++, использующая средства стандарта Open MP, выигрывает у программы, написанной на C, использующей для распараллеливания инструментарий TPL – Task Parallel Library (рисунок).

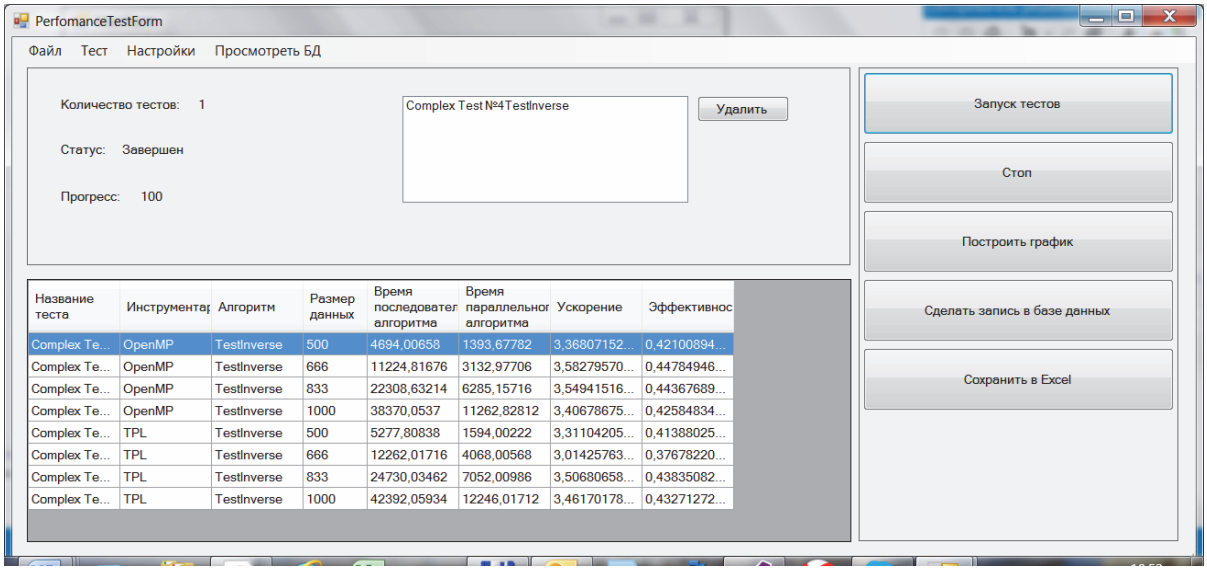


Рис. Результаты сравнения для задачи обращения матриц

На данной задаче инструментарий Open MP создает более эффективную программу. Выигрыш по времени во всех примерах составляет около 10 %. Следует отметить, что на некоторых задачах TPL программы выигрывают у Open MP программ, но выигрыш во всех ситуациях находится в 10%-ной зоне.

Подводя общие итоги, можно отметить, что оба инструментария предоставляют адекватные средства для создания параллельных программ, практически одинаково удобные для написания и отладки программ. Создаваемые программы практически не отличаются по эффективности – времени решения задачи.

Список литература

1. Биллиг В.А. Параллельные вычисления и многопоточное программирование / ИНТУИТ – ЗАО НИИ ЦПС. – М.: Тверь, 2013.

2. Антонов А.С. Технологии параллельного программирования MPI и Open MP. – М.: Изд-во МГУ, 2012.

3. Смирнов М.Ю. Параллельные вычисления. Сравнительный анализ инструментария Open MP и .NET TPL. – Тверь: Изд-во ТГТУ, 2014.

А.В. Бовырин[1,2], П.Н. Дружков[1], В.Л. Ерухимов[2], Н.Ю. Золотых[1], К.В. Корняков[1,2], В.Д. Кустикова[1], И.Д. Лысенков[2], И.Б. Мееров[1], В.Н. Писаревский[2], А.Н. Половинкин[1], А.В. Сысоев [1]

[1] Нижегородский государственный университет им. Н.И. Лобачевского
[2] Itseez, Нижний Новгород

ОБРАЗОВАТЕЛЬНЫЙ КУРС «РАЗРАБОТКА МУЛЬТИМЕДИЙНЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК OPENCV И IPP»

Представлено описание образовательного курса, посвященного разработке мультимедийных приложений, с акцентом на алгоритмах компьютерного зрения. Рассматривается роль курса в системе подготовки специалистов ИТ-профиля, дается описание структуры курса, приводится содержание теоретической и практической его части. Указывается особая роль суперкомпьютерных технологий в приложениях интеллектуальной обработки визуальной информации.

Разработка курса выполнена специалистами факультета ВМК ННГУ и компании Itseez при поддержке корпорации Intel.

Введение

В настоящее время продолжает бурно развиваться область создания, обработки и интеллектуального анализа мультимедийных данных [1, 2]. При этом сохраняются две тенденции: с одной стороны, растет сложность решаемых задач (увеличение разрешения контента и объемов данных), и одновременно с этим расширяется спектр используемых уст-

ройств (от высокопроизводительных облачных серверов до энергоэффективных носимых устройств). Подобная ситуация создает большой спрос на технологии высокопроизводительной обработки мультимедийных данных и, как следствие, на специалистов, имеющих соответствующие компетенции.

Предлагаемый образовательный курс призван помочь в решении указанной проблемы и ориентирован на использование в системе профессиональной подготовки по направлению высокопроизводительных вычислений. Отличительной особенностью курса является акцент на практическую его составляющую, на формирование навыков использования современных программных библиотек, таких как OpenCV [3,4] и IPP [5]. Важно отметить, что разработка курса выполнена специалистами факультета ВМК ННГУ и компании Itseez при поддержке корпорации Intel с учетом многолетнего опыта профессиональной деятельности в указанной предметной области. Курс к настоящему моменту опубликован [6] и прошел апробацию в нескольких молодежных школах.

В секции 1 дается общая характеристика курса, в секциях 2 и 3 – описание теоретической и практической части соответственно. В заключении описывается опыт апробации курса и делаются соответствующие выводы.

Общая характеристика курса

Курс посвящен тематике интеллектуальной обработки визуальной информации, включает теоретическую и практическую часть. Теоретическая часть курса рассматривает типичные задачи указанной области и методы их решения. Практикум ставит задачей формирование навыков решения практических задач и подразумевает написание программного кода для их решения. Особо рассматривается использование оптимизированных библиотек OpenCV и IPP, имеющих высокую производительность на основных современных аппаратных платформах.

Курс состоит из двух частей: вводной и расширенной. Вводная часть курса покрывает основные задачи, модели и методы, встречающиеся при создании мультимедийных приложений с фокусом на задачах компьютерного зрения. Расширенная часть призвана дополнить вводную несколькими новыми разделами как в плане теории, так и в плане практики. Таблица отражает содержание курса.

Предполагается, что по итогами обучения слушатели смогут обоснованно выбирать алгоритмы решения возникающих задач и создавать эффективные реализации под целевые аппаратные платформы.

Содержание курса

Версии	Теоретическая часть (темы лекций)	Практикум (лабораторные работы)
Вводная	1. Обзор задач и методов компьютерного зрения. 2. Введение в машинное обучение. 3. Детекторы и дескрипторы ключевых точек. Алгоритмы классификации изображений. Задача детектирования объектов на изображениях и методы ее решения. 4. Введение в OpenCV. 5. Обзор библиотеки IPP	1. Сборка и установка библиотеки OpenCV. Использование библиотеки в среде разработки Microsoft Visual Studio. 2. Базовые операции над изображениями. 3. Машинное обучение в библиотеке OpenCV. 4. Классификация изображений с использованием bag-of-words методов. 5. Библиотека IPP и ее использование в среде разработки Microsoft Visual Studio. 6. Сравнение производительности библиотек IPP и OpenCV
Расширенная	6. Отслеживание движения и сопровождения объектов. 7. Проективная геометрия в компьютерном зрении	7. LatentSVM. 8. Детектирование пешеходов

В качестве предварительных требований предполагается, что слушатели владеют базовыми навыками программирования на языке C++ (алгоритмы и структуры данных, объектно-ориентированное программирование), а также знаниями высшей математики в объеме первых двух курсов факультетов естественно-научного профиля.

Лекционная часть

Лекционная часть курса содержит 5 лекций вводной части курса и 2 дополнительные лекции. Вводная часть во многом ориентирована на задачу детектирования и классификации объектов. Приводятся необходимые базовые сведения из компьютерного зрения и машинного обучения, затем разбирается несколько конкретных алгоритмов. Кроме того, дается введение в библиотеки OpenCV и IPP, подробно рассматривается функционал, применимый для решения указанных выше задач. В двух дополнительных лекциях материал расширяется на задачи отслеживания объектов, а также дается необходимый материал из проективной геометрии.

Каждая лекция сопровождается конспектом и презентацией.

Практикум

Лабораторный практикум делает упор на практическое освоение библиотек OpenCV и IPP. Описывается первичная установка и настройка биб-

лиотек, рассматриваются базовые возможности, производится сравнительный анализ библиотек. В качестве предмета исследования предлагаются базовые операции обработки изображений, методы машинного обучения и классификации изображений. Дополнительные лабораторные работы рассматривают методы детектирования объектов на примере подхода LatentSVM и задачи детектирования пешеходов.

Лабораторные работы сопровождаются методическими указаниями (в формате текстового документа и слайдов), а также примерами исходного кода.

Заключение

Предлагаемый курс посвящен актуальной на сегодняшний день тематике – разработка мультимедийных приложений. Он содержит теоретические знания, необходимые для решения ряда популярных задач компьютерного зрения. Вместе с тем практикум ставит упор на решение задач с использованием современных библиотек, опеспечивающих высокую производительность. Тем самым обеспечивается всесторонняя подготовка будущих специалистов.

Первая редакция курса прошла апробацию в рамках научной школы «Робототехника, компьютерное зрение и машинное обучение» (форум «Суперкомпьютерные технологии в образовании, науке и промышленности», Нижний Новгород, ННГУ, 20–28 ноября 2012 г.).

Список литературы

1. Forsyth D., Ponce J. Computer Vision: A Modern Approach. – Prentice Hall Professional Technical Reference. – 2002.

2. Shapiro L., Stockman G. Computer Vision. – Prentice Hall, 2001.

3. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. – O'Reilly Media Inc., 2008.

4. Библиотека OpenCV [Электронный ресурс]. – URL: http://opencv.org.

5. Библиотека Intel IPP [Электронный ресурс]. – URL: https://software.intel.com/en-us/intel-ipp.

6. Материалы курса на сайте Национального Открытого Университета «ИНТУИТ» [Электронный ресурс]. – URL: http://www.intuit.ru/studies/courses/10622/1106/info.

И.П. Болодурина, Д.И. Парфёнов

Оренбургский государственный университет

ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧИ ОПТИМАЛЬНОГО ИСПОЛЬЗОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ В ОБЛАЧНЫХ СИСТЕМАХ

Задача оптимального использования вычислительных ресурсов весьма актуальна для информационных систем, имеющих физические ограничения по производительности оборудования. Высокая интенсивность использования мультимедийных систем вузами требует постоянного наращивания вычислительных ресурсов, но не всегда такие запросы могут быть своевременно удовлетворены. Облачные технологии позволяют использовать особенности планирования задач в GRID-системах для оптимизации запуска приложений и виртуальных машин с учетом требований к качеству обслуживания.

Введение

Информационные потоки, передаваемые между вычислительными узлами как в локальных, так и в глобальных сетях, с каждым годом растут. Это касается не только больших центров обработки данных (ЦОД), но и локально расположенных вычислительных центров (ВЦ), специализирующихся на решении профильных задач в таких областях, как промышленность, экономика, здравоохранение и др. Одной из сфер применения локальных ВЦ является образование. Вузы все чаще используют собственные ВЦ для поддержки интегрированных автоматизированных информационных систем (ИАИС), предоставляющих конечным пользователям сетевые мультимедийные услуги [1].

Одной из проблем высоконагруженных ИАИС является потребность в больших объемах вычислительных ресурсов. При этом рост потребления ресурсов в отличие от изменения доступного объема имеет экспоненциальный характер [2]. При анализе потоков запросов к сервисам ИАИС установлено, что они имеют неоднородную структуру [1]. Современные сервисы ИАИС строят на базе концепции облачных вычислений. Однако проблема ограниченности ресурсов, используемых для работы облачной системы, остается актуальной [4].

Потоки передаваемых данных в ИАИС, как правило, обрабатываются в несколько этапов, называемых фазами обслуживания. При этом на каждой фазе может применяться несколько однотипных элементов, обеспечивающих балансировку и распределение нагрузки между элементами ин-

формационной системы. Количество элементов в каждой фазе зависит от функционала информационной системы и количества приложений, входящих в ее состав. Представим информационную систему в виде кортежа

$$IS = \{S_1, \ldots, S_r\},$$

где S_i – элемент, выполняющий обработку данных, на основе поступающего потока запросов пользователей, $i = 1...r$ (r – общее число элементов в информационной системе).

Количество фаз обслуживания f на пути потока запросов пользователей информационной системы зависит от ее архитектуры. В дальнейшем потоки данных, формируемые в информационной облачной системе, будем рассматривать для системы дистанционного обучения, реализующей трехфазное обслуживание запросов пользователей.

Определим назначение каждой из фаз обслуживания запросов относительно их расположения в последовательности обработки запросов пользователей:

– первая фаза – распределение потоков данных между ресурсами ИС в облаке;

– вторая фаза – динамическое масштабирование вычислительных ресурсов в облаке;

– третья фаза – обработка данных пользователя приложениями, системами хранения данных и СУБД.

Детализируем кортеж элементов, входящих в состав информационной системы:

$$IS = \{S_1^1, \ldots, S_n^1, S_1^2, \ldots, S_m^2, S_1^3, \ldots, S_k^3\},$$

где S_i^j – i-й элемент j-й фазы; $m \in N, n \in N, k \in N$ – число элементов, входящих в состав системы на соответствующих фазах f.

Введем также входные S_i^0 элементы, передающие поток данных в информационную систему, и выходные S_i^4 элементы, принимающие потоки данных из облачной инфраструктуры. Следовательно, кортеж, описывающий элементы информационной системы, преобразуется к виду

$$IS = \{S_1^0, \ldots, S_l^0, S_1^1, \ldots, S_n^1, S_1^2, \ldots, S_m^2, S_1^3, \ldots, S_k^3, S_1^4, \ldots, S_p^4\},$$

где $m \in N, n \in N, k \in N$ – число элементов, входящих в состав системы на соответствующих фазах f; $l \in N, p \in N$ – число элементов на входе и выходе информационной облачной системы.

Каждый элемент S_i^j информационной системы в любой момент времени может обслуживать несколько запросов, поступивших от разных

пользователей. При этом в процессе выполнения запроса пользователя образуются потоки данных на входе и выходе из элемента системы, обладающие индивидуальными характеристиками, изменяющимися во времени. Обозначим все входящие потоки элемента S_i^j как X_i^j, а выходящие – Y_i^j, где i – номер элемента на j-й фазе обслуживания. Каждый поток запросов можно описать набором характеристик. Пусть для элемента системы S_i^j существует l_i^j входящих потоков и p_i^j выходящих. Тогда для входящего υ-го потока, $\upsilon = 1...l_i^j$, введем набор характеристик

$$X_i^{(j,\nu)}(t) = \left(x_{1,i}^{(j,\nu)}(t),\ldots,x_{k,i}^{(j,\nu)}(t)\right)^T,$$

где $x_{1,i}^{(j,\nu)}$ – интенсивность поступления заявок в каждом ν-м потоке на входе элемента S_i^j; $x_{2,i}^{(j,\nu)}$ – время обслуживания ν-го потока заявок в элементе S_i^j; $x_{3,i}^{(j,\nu)}$ – интенсивность обслуживания υ-го потока заявок в элементе S_i^j; $x_{4,i}^{(j,n)}$ – дисциплина обслуживания υ-го потока заявок в элементе S_i^j, определяющая порядок обслуживания в соответствии с алгоритмом расстановки приоритетов в информационной системе; $x_{5,i}^{(j,\nu)}$ – класс обслуживания υ-го потока заявок в элементе S_i^j; $x_{6,i}^{(j,\nu)}$ – количество заявок, поступивших из υ-го потока в элемент S_i^j;

Для выходящего μ-го потока, $\mu = 1...p_i^j$, элемента системы S_i^j набор характеристик следующий:

$$Y_i^{(j,\mu)}(t) = \left(y_{1,i}^{(j,\mu)}(t),\ldots,y_{k,i}^{(j,\mu)}(t)\right)^T,$$

где $y_{1,i}^{(j,\mu)}$ – интенсивность поступления заявок в каждом μ-м потоке на выходе элемента S_i^j; $y_{2,i}^{(j,\mu)}$ – интенсивность обслуживания μ-го потока заявок на выходе элемента S_i^j; $y_{3,i}^{(j,\mu)}$ – дисциплина обслуживания μ-го потока заявок на выходе элемента S_i^j, определяющая порядок передачи данных в соответствии с алгоритмом расстановки приоритетов в информационной облачной системе; $y_{4,i}^{(j,\mu)}$ – класс обслуживания заявок μ-го потока заявок на выходе элемента S_i^j; $y_{5,i}^{(j,\mu)}$ – количество обслуженных заявок в μ-м потоке на выходе элемента S_i^j.

Интенсивность обслуживания рассматриваемых потоков заявок информационной системы, запрашиваемых пользователями, зависит от класса обслуживания, определяемого уровнем конечного приложения. В общем

случае интенсивность поступления и обслуживания потока заявок для каждого класса приложений определяется функцией собственной функцией распределения.

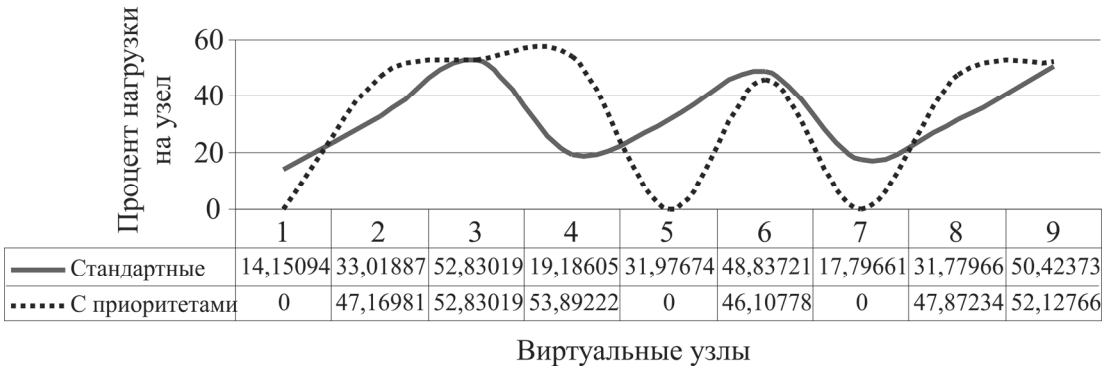
При обслуживании заявок в информационной системе, построенной с использованием облачной архитектуры, траектория прохождения фаз для каждого потока может изменяться динамически. При этом количество уникальных потоков напрямую зависит от числа элементов на каждой фазе.

Таким образом, для описания всей многофазной информационной облачной системы формализуем описание потоков каждой фазы в виде отображения $R^j : X^j \to Y^j$. Тогда потоки данных в информационной системе, расположенной в облаке, можно представить в виде

$$Y_i^j = R^j(X_i^j) = \begin{cases} R(X_i^j), & X_i^j \in X^j \\ P_Y^j(X^{j*}), & X^{j*} \in \bigcup\limits_{S_i^j} \left[X_i^{j,0} \bigcup \left(\bigcup\limits_{S_i^{j-1}} X_i^{j,j-1} \right) \right] \\ Q_X^j(Y^{j*}), & Y^{j*} \in \bigcup\limits_{S_i^j} \left[Y_i^{j,0} \bigcup \left(\bigcup\limits_{S_i^{j+1}} Y_i^{j,j+1} \right) \right] \end{cases}$$

Поскольку потоки данных и их характеристики могут изменяться во времени, следует дополнить полученное описание потоков данных в информационной системе параметром времени *t*. Кроме того, поскольку на информационную систему оказывают влияние не только внутренние процессы и потоки предаваемых данных, но и внешние факторы, описание системы можно дополнить параметром внешнего воздействия *F*. Тогда потоки передаваемых данных в облачной информационной системе можно описать выражением вида $Y_i^j = R^j(X_i^j, t, F)$.

Предложенная модель положена в основу разработанного алгоритма управления ресурсами облачной системы, позволяющего оптимизировать использование вычислительных ресурсов в условиях их ограниченности. В рамках экспериментального исследования в облачной системе OpenStack [3] проведено сравнение реального потребления выделенных виртуальных ресурсов. Оно проводилось по следующим критериям оценки эффективности: время отклика сервера и количество выделенных виртуальных серверов для каждой из подсистем. На рисунке представлен график, показывающий эффективность работы системы управления вычислительными ресурсами облачной системы.



	1	2	3	4	5	6	7	8	9
Стандартные	14,15094	33,01887	52,83019	19,18605	31,97674	48,83721	17,79661	31,77966	50,42373
С приоритетами	0	47,16981	52,83019	53,89222	0	46,10778	0	47,87234	52,12766

Рис. График эффективности использования вычислительных ресурсов облачной системы

За счет оптимального выделения ресурсов на каждом вычислительном узле гарантировано обеспечение совместной работы всех запущенных экземпляров приложений, что удовлетворяет требованиям потенциальных пользователей. При этом благодаря работе алгоритмов оптимизации возможно высвобождение от 20 до 30 % выделенных ресурсов (виртуальных серверов).

Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 13-07-00198 и 13-01-97050.

Список литературы

1. Болодурина И.П., Решетников В.Н., Парфёнов Д.И. Распределение ресурсов в информационной системе дистанционной поддержки образовательного процесса. Программные продукты и системы. – Тверь: Фактор, 2012. – № 3. – С. 151–155.

2. Бойченко И.В., Корытников С.В. Управление ресурсами в сервис-ориентированных системах типа «приложение как сервис» // Доклады Том. гос. ун-та систем упр. и радиоэлектроники. – 2010. – Вып. 1–2. – С. 156–160.

3. Open Source Cloud Computing Software // OpenStack. – URL: http://www.openstack.org.

4. Проблема адекватной оценки производительности веб-серверов в корпоративных сетях на предприятиях ЦБП / О.В. Гусев, А.В. Жуков, В.В. Поляков, С.В. Поляков // Новые информационные технологии в ЦБП и энергетике: материалы 6-й науч.-техн. конф. – Петрозаводск, 2004. – С. 84–87.

М.Г. Бояршинов, А.А. Скутин

Пермский национальный исследовательский политехнический университет

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ И РЕСУРСОВ ЦВВС ПНИПУ ДЛЯ МОДЕЛИРОВАНИЯ ВЗАИМОДЕЙСТВИЯ ТВЕРДОЙ ЧАСТИЦЫ С ПОТОКОМ ЖИДКОСТИ

Разработан и проверен с использованием точного решения Стокса алгоритм численного решения задачи об обтекании твердой частицы (сферы заданного радиуса) однородным потоком жидкости, имеющим заданную скорость. Распараллеливание алгоритма вычислительной программы реализовано на основе технологии MPI. Установлено, что использование вычислительного кластера позволяет существенно сократить время расчетов.

Введение

Вычислительное моделирование обтекания тел набегающим потоком жидкости или газа актуально при решении практически важных инженерных задач в авиационной, космической, автомобильной, железнодорожной технике, при проектировании машин, механизмов, конструкций, приборов и аппаратов. Экспериментальное исследование взаимодействия отдельных твердых, жидких или газовых частиц с несущими потоками чрезвычайно сложно, поэтому целесообразно использовать возможности математического моделирования для определения основных характеристик силового, кинематического, температурного и прочих видов взаимодействия отдельных фаз и компонентов, составляющих движущиеся смеси.

Численное решение задач гидродинамики требует использования значительных вычислительных ресурсов, многократного решения систем линейных и нелинейных уравнений с сильно разреженными матрицами, имеющими нерегулярные структуры заполненности. Для повышения производительности вычислительных работ целесообразно использовать современные технологии распараллеливания вычислительных алгоритмов.

Постановка задачи

Рассматривается задача об обтекании сферы с радиусом R однородным потоком со скоростью V_∞ (рис. 1). Центр сферы помещен в начало координат [1].

Задача решалась в приближении несжимаемой ньютоновской жидкости. В рамках принятой модели процесс описывается нестационарной системой уравнений Навье – Стокса:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{U}\mathbf{U} = \frac{1}{\mathrm{Re}} \nabla \cdot \nabla \mathbf{U} - \nabla p, \qquad (1)$$

$$\nabla \cdot \mathbf{U} = 0, \tag{2}$$

где p – скалярное поле давления, $\mathbf{U}$ – векторное поле скорости, ∇ – оператор Гамильтона, Re – число Рейнольдса.

Форма расчетной области представляет собой куб, в центр которого помещена сфера. Радиус сферы в 10 раз меньше ребра куба. Граничные условия: на входной границе задаются постоянные значения скорости и давления, на поверхности сферы – условие прилипания жидкости. На остальных границах расчетной области задаются условия свободного переноса. Начальные условия: скорость потока во всей области направлена вдоль оси Ox и равна 1, давление задается равным 0 (за исключением твердой частицы).

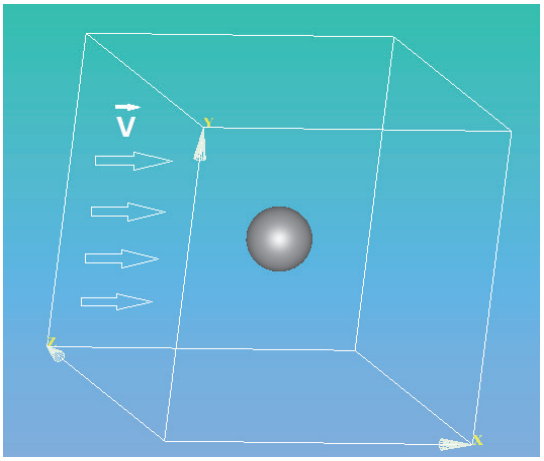


Рис. 1. Расчетная схема задачи

Численный метод

Для численного решения задачи используется метод контрольных объемов. Расчетную область разбивают на некоторое число непересекающихся контрольных объемов таким образом, что каждая узловая точка разностной сетки содержится в своем контрольном объеме. Дифференциальное уравнение интегрируют по каждому контрольному объему. В результате строится дискретный аналог дифференциального уравнения. В работе получена конечно-объемная схема с погрешностью второго порядка, которая базируется на вычислении потока среды сквозь грани контрольных объемов [2].

Уравнения движения (1), (2) интегрируются по контрольному объему m (рис. 2):

$$\frac{\partial}{\partial t}\int_{V_m} \mathbf{U} dV + \int_{V_m} \mathbf{U}\mathbf{U} dV = \frac{1}{\mathrm{Re}}\int_{V_m} \nabla \cdot \nabla \mathbf{U} dV - \int_{V_m} \nabla p dV, \tag{3}$$

$$\int_{V_m} \nabla \cdot \mathbf{U} dV = 0 \tag{4}$$

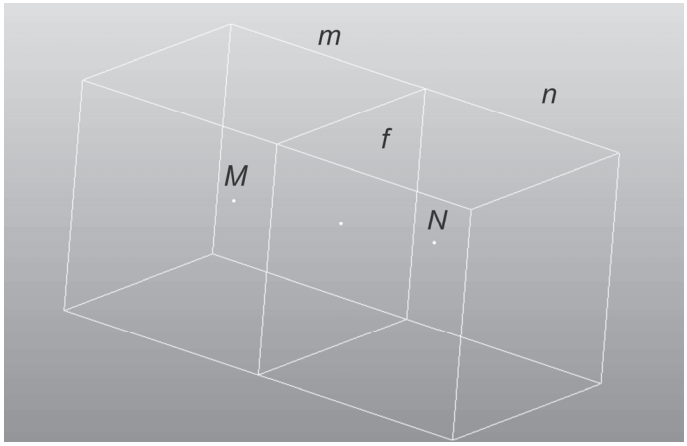


Рис. 2. Контрольные объемы *m* и *n*, имеющие общую грань *f*

Интегралы в формулах (3), (4) вычисляются по формуле Остроградского – Гаусса:

$$\int_{V_m} \nabla \cdot \mathbf{A} dV = \int_{S_m} d\mathbf{S} \cdot \mathbf{A}; \tag{5}$$

$$\int_{V_m} \nabla \mathbf{A} dV = \int_{S_m} d\mathbf{S}\mathbf{A};$$

$$\int_{V_m} \nabla a dV = \int_{S_m} d\mathbf{S} a; \tag{6}$$

где S_m – поверхность контрольного объема *m*, $\mathbf{A}$ – вектор, *a* – скаляр, ∇ – оператор Гамильтона.

Вычисляя интеграл в формуле (5), получаем приближенную форму теоремы Остроградского – Гаусса:

$$(\nabla \cdot \mathbf{U})_M V_M \approx \sum_f \mathbf{S}_f \cdot \mathbf{U}_f, \tag{7}$$

где $\mathbf{S}_f$ – вектор внешней нормали к грани *f* (рис. 2), модуль которого равен площади грани *f*. Суммирование в правой части проводится по всем граням контрольного объема *m*. Вычисляя интеграл в формуле (6), получаем следующее соотношение:

$$(\nabla p)_M V_m \approx \sum_f \mathbf{S}_f h_f.$$

Дискретизация конвективных членов приводит к соотношению

$$\int_{V_m} \nabla \cdot \mathbf{U}\mathbf{U} dV = \sum_f \mathbf{S}_f \cdot \mathbf{U}_f \mathbf{U}_f = \sum_f F\mathbf{U}_f,$$

где *F* – поток среды сквозь грань *f*, $\mathbf{U}_f$ – значение скорости на грани *f*, которое определяется из значений скорости в центрах соседних ячеек *M* и *N* посредством конвективных разностных схем.

Диффузионный член дискретизируется аналогичным образом:

$$\int_{V_m} \nabla \cdot \mathbf{U} dV = \sum_f \mathbf{S}_f \cdot (\nabla \mathbf{U})_f.$$

Правую часть формулы (12) можно представить в виде

$$\mathbf{S}_f \cdot (\nabla \mathbf{U})_f = \mathbf{S}_f \frac{\mathbf{U}N - \mathbf{U}M}{|MN|}.$$

Для дискретизации производной по времени используется неявная схема Эйлера, которая имеет первый порядок погрешности по временному шагу:

$$\int_{V_m} \frac{\partial \mathbf{U}}{\partial t} dV = \left(\frac{\partial \mathbf{U}}{\partial t} \right)_M V_m = \frac{\mathbf{U}_M^k - \mathbf{U}_M^{k-1}}{\Delta t} V_m.$$

В конечном итоге уравнение (1) в дискретной форме можно записать следующим образом:

$$\frac{\mathbf{U}_M^k - \mathbf{U}_M^{k-1}}{\Delta t} V_m + \sum_f F \mathbf{U}_f^k - \frac{1}{\mathrm{Re}} \sum_f \mathbf{S}_f (\nabla \mathbf{U}^k)_f = -\sum_f \mathbf{S}_f p_f^k,$$

а уравнение (2) – в виде

$$\sum_f \mathbf{S}_f \cdot \mathbf{U}_f^k = 0.$$

Для обеспечения устойчивости вычислительного процесса используется критерий Куранта – Фридрихса – Леви. Этот критерий означает, что частица жидкости за один шаг по времени не должна продвинуться больше, чем на один сеточный шаг по координате. Для достижения устойчивого решения с малой погрешностью требуется, чтобы число Куранта было менее 1. Для обеспечения этого условия во всей расчетной области шаг по времени задан равным 0,02.

Для определения поля давления применяется алгоритм PISO [3], состоящий в расщеплении решения на каждом временном интервале на ряд шагов, на каждом из которых уравнение для давления выделяется из уравнений движения.

Для описания геометрии расчетной области и построения разностной сетки используется открытая интегрируемая платформа численного моделирования SALOME [4]. Алгоритм численного решения задачи реализован с помощью платформы OpenFOAM [5]. Построенная с использованием SALOME расчетная область интегрировалась в OpenFOAM, который содержит поддержку запуска расчетов на кластере с использованием технологии MPI (Message Passing Interface).

Распараллеливание алгоритма

MPI – наиболее распространенная технология программирования параллельных компьютеров с распределенной памятью [6]. Основным способом взаимодействия параллельных процессов является передача сообщений друг другу. MPI-программа – это множество параллельных взаимодействующих процессов. Каждый процесс работает в своем адресном пространстве, никаких общих переменных или данных в MPI нет. Применяется метод декомпозиции расчетной области, который заключается в разбиении рассматриваемой области на части, для каждой из которых расчеты выполнялись на отдельном процессоре. В решаемой задаче расчетная область разбивается на 8 частей. Вычислительный эксперимент проводился на вычислительном кластере ПНИПУ.

При разработке параллельных алгоритмов решения задач вычислительной математики принципиальным моментом является анализ эффективности использования параллелизма, состоящий обычно в оценке получаемого ускорения процесса вычисления (сокращения времени решения задачи) [7]. Формирование подобных оценок ускорения может осуществляться применительно к выбранному вычислительному алгоритму (оценка эффективности распараллеливания конкретного алгоритма). Другой важный подход может состоять в построении оценок максимально возможного ускорения процесса получения решения задачи конкретного типа (оценка эффективности параллельного способа решения задачи).

Результаты расчетов

На рис. 3 приведено распределение модуля скорости в сечениях $z = 0$ и $y = 0$ рассматриваемой области, на рис. 4 – поле давления в сечении $z = 0$ расчетной области по истечении времени 10 с после начала движения потока жидкости для числа Рейнольдса Re = 0,7277. Скорость потока уменьшается при достижении сферы, принимает нулевое значение на границах сферы, что соответствует заданным граничным условиям задачи. За сферой скорость постепенно возрастает.

На рис. 4 наблюдается повышенное давление при достижении потоком жидкости поверхности сферы, одновременно за сферой образовалась область пониженного давления. Оценка погрешности результатов численного эксперимента проводилась сравнением коэффициента сопротивления C_x, определяемого численным моделированием, с точным значением, полученным в задаче Стокса [1]. Результаты сравнения приведены в таблице.

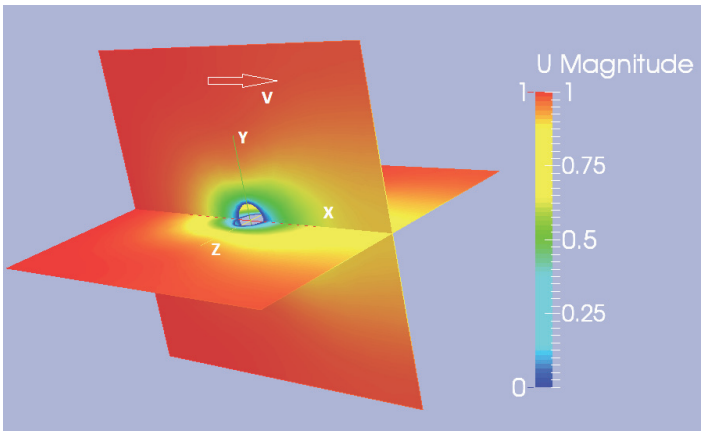


Рис. 3. Распределение модуля скорости
в сечениях *Oxy* и *Oxz*

На рис. 5 и 6 представлены зависимости времени и ускорения выполнения расчетов от количества процессоров. Ускорение, получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется как

$$S_p = T_1 / T_p,$$

т.е. как отношение времени решения задач на скалярной ЭВМ к времени выполнения параллельного алгоритма [7].

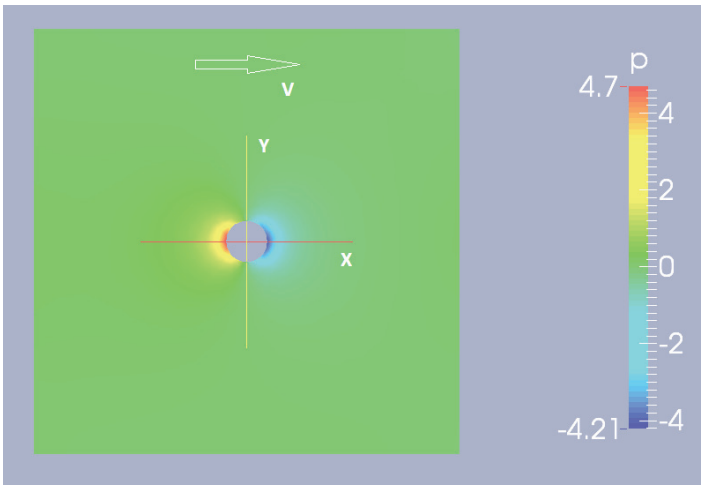


Рис. 4. Распределение поля давления в сечении *Oxy*

Сравнение коэффициента сопротивления C_x, полученного в вычислительном эксперименте, с точным значением в решении Стокса

Число Re	Значение коэффициента C_x		Относительная погрешность δ
	Точное решение [1]	Численное решение	
0,7277	32,98	30,01	0,09
0,2437	98,5	95,8	0,03

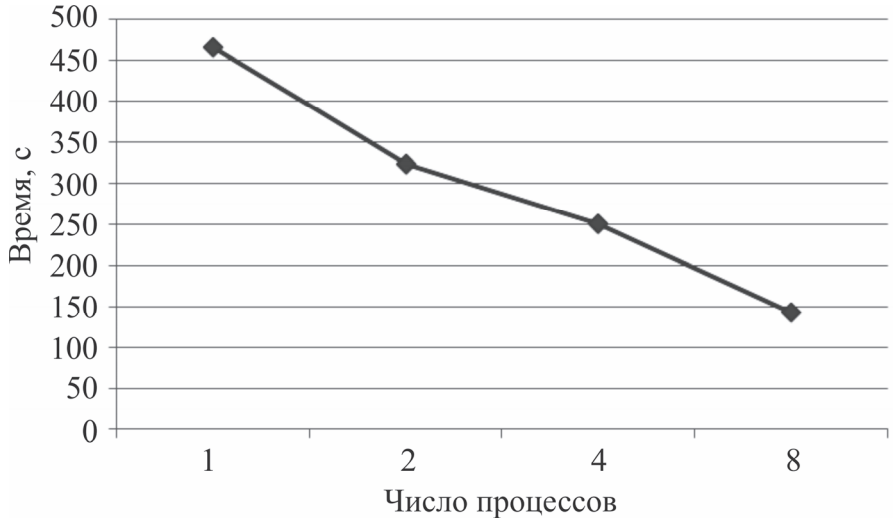


Рис. 5. Зависимость времени выполнения расчетов
от числа процессоров

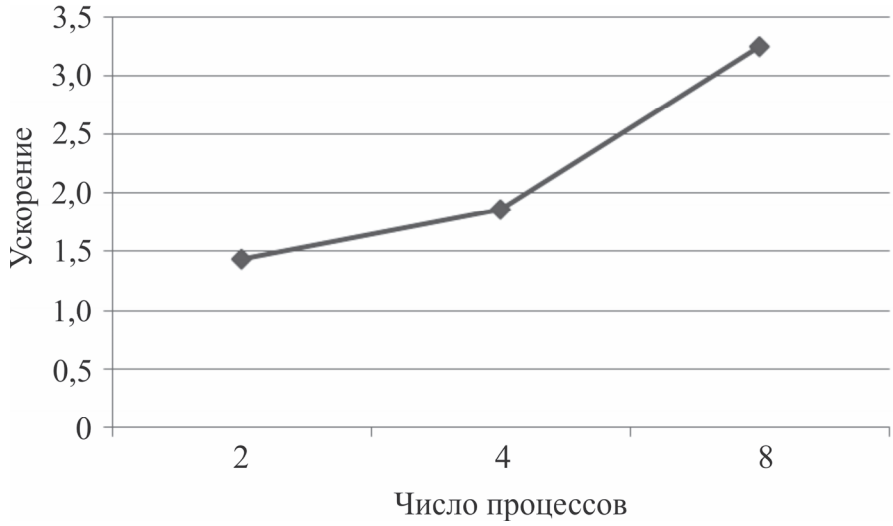


Рис. 6. Зависимость ускорения выполнения расчетов
от числа процессоров

Выводы

Разработаны и проверены на точном решении Стокса алгоритм и вычислительная программа, позволяющие моделировать взаимодействие частиц произвольной формы и состава с несущими потоками газа и жидкости, определять параметры их взаимодействия. Это позволит обобщать получаемые результаты на взаимодействие множества частиц между собой и с несущими потоками при моделировании движения многофазных сред. Реализовано распараллеливание алгоритма вычислительной программы путем запуска расчетов на кластере через MPI. Установлено, что использование восьми ядер процессора позволяет сократить время расчета более чем в 3 раза.

Список литературы

1. Лойцянский Л.Г. Механика жидкости и газа. – М.: Наука, 1973. – 847 с.
2. Патанкар С.В. Численное решение задач теплопроводности и конвективного теплообмена при течении в каналах. – М.: Изд-во МЭИ, 2003. – 312 с.
3. Issa R.I. Solution of implicitly discretised fluid flow equations by operator-splitting // J. Comput. Phys. – 1986. – № 62. – P. 40–65.
4. SALOME. – URL: http://www.salome-platform.org.
5. OpenFOAM. – URL: http://www.openfoam.org.
6. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.
7. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем: учеб. пособие. – Н. Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184 с.

М.Г. Бояршинов, В.А. Трушков, А.В. Казанцев

Пермский национальный исследовательский политехнический университет

ИСПОЛЬЗОВАНИЕ РЕСУРСОВ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА ЦВВС ПНИПУ ДЛЯ ОПРЕДЕЛЕНИЯ ДИНАМИЧЕСКИХ И ПРОЧНОСТНЫХ ХАРАКТЕРИСТИК МЕХАНИЧЕСКОГО ОБОРУДОВАНИЯ

Рассмотрена возможность применения вычислительного моделирования для замены вычислительным экспериментом проведения натурных динамических испытаний механического оборудования на примере электрического двигателя. Выбран инструментарий для решения поставленной задачи, подготовлены исходные данные, выполнен ряд вычислительных работ по определению динамических и прочностных характеристик двигателя. С помощью ресурсов вычислительного кластера ПНИПУ выполнена верификация результатов компьютерного моделирования динамических и прочностных характеристик на основе данных натурных испытаний. Проведенный анализ показал, что исследуемый двигатель соответствует нормативным требованиям к динамическим характеристикам.

В последние годы параллельные вычисления привлекают пристальное внимание самых разных специалистов. Применение параллельных вычислений позволяет исследователю находиться на передовом крае науки. Эта сфера деятельности затрагивает разработку численных методов, изу-

чение структурных свойств алгоритмов, создание новых языков программирования, построение интерфейса между пользователем и компьютером и пр. Инженерные новшества в области компьютерной техники активно стимулируют разработку новых алгоритмов, эффективно использующих эти новшества. Кроме этого, рассматриваемая сфера деятельности сравнительно доступна для начинающих исследователей [1].

Приобретение, техническое обслуживание, расходы на эксплуатацию и обслуживание оборудования, необходимого для проведения регламентированных ГОСТами и ТУ механических испытаний, как и непосредственное проведение таких испытаний в сторонних организациях, требует от промышленного предприятия значительных материальных, организационных и финансовых затрат.

Компьютерное моделирование статических и динамических характеристик машин, приборов и аппаратуры, их элементов и деталей является наиболее целесообразным способом анализа напряженно-деформированного состояния, мгновенной и длительной прочности, собственных форм и частот колебаний, отклика на ударные и циклические воздействия и прочих механических показателей как механизмов и конструкций в целом, так и их отдельных составляющих. Вычислительный эксперимент на основе математической модели позволяет, во-первых, получать наиболее полные данные о состоянии механических объектов и параметрах их нагружения; во-вторых, такой подход экономичен; в-третьих, более информативен по сравнению с традиционными экспериментальными методами и, наконец, позволяет делать оценки аварийных ситуаций, натурное моделирование которых опасно.

Сказанное тем более актуально, что согласно ГОСТ 16962.2–90 [2] допускается наряду с механическими испытаниями оценивать изделия расчетным или расчетно-экспериментальным методом, особенно в том случае, когда масса или габаритные размеры готового изделия не позволяют проводить его испытания на существующем оборудовании и оно после изготовления не может быть разделено на отдельные блоки и узлы.

В настоящее время оценка динамических механических характеристик электродвигателей основана на проведении следующих видов механических испытаний [2]:

– определение диапазона частот собственных колебаний конструкции проводится с целью проверки механических свойств изделий и получения исходной информации для выбора методов испытаний на вибропрочность, виброустойчивость;

– испытания на прочность при кратковременном и длительном воздействии синусоидальной вибрации проводятся для проверки способности

изделий противостоять разрушающему действию вибрации и сохранять свои параметры после ее воздействия;

– испытание на ударную устойчивость проводят с целью проверки способности изделий противостоять разрушающему действию одиночных ударов с большими ускорениями и выполнять свои функции после воздействия ударов.

Вычислительное моделирование позволяет заменить перечисленные выше виды механических динамических испытаний конструкций, машин и механизмов проведением следующих вычислительных экспериментов:

– определение спектра частот собственных колебаний электродвигателя для любого заданного диапазона;

– исследование напряженно-деформированного состояния электродвигателя при воздействии кратковременной вибрации, т.е. в пределах одного синусоидального цикла;

– определение показателей длительной прочности электродвигателя при воздействии длительной синусоидальной вибрации;

– исследование напряженно-деформированного состояния электродвигателя при воздействии одиночного удара по заданному закону изменения ускорения.

Современное программное обеспечение для выполнения вычислительных экспериментов с целью определения напряженно-деформированного состояния строительных конструкций, машин и приборов основывается на фундаментальных законах физики и механики, использует современный аппарат численного решения дифференциальных уравнений математической физики, что позволяет с высокой достоверностью и низкой погрешностью описывать механические процессы в деталях и конструктивных элементах механизмов и машин, в том числе электродвигателей.

В настоящее время разработано и используется большое количество программных продуктов, предназначенных для выполнения инженерного анализа механических систем и процессов, для исследования напряженно-деформированного состояния устойчивости, надежности прочности механизмов и машин, строительных конструкций, зданий и сооружений в целом и их отдельных деталей и узлов: Abaqus, APM Winmachine, APM FEM для КОМПАС-3D, MSC Nastran, SolidWorks Simulation, ANSYS и целый ряд других. Компьютерное моделирование с использованием современных коммерческих пакетов прикладных программ позволяет обоснованно отказаться от проведения дорогостоящих натурных испытаний механических объектов в пользу проведения вычислительных экспериментов.

Для проведения настоящего исследования выбран конечно-элементный программный комплекс ANSYS, объективно обладающий целым рядом преимуществ:

– ANSYS отлично зарекомендовал себя в сфере инженерных расчетов, с его помощью возможно решение широкого спектра задач механики деформирования твердого тела, жидкости и газа;

– Центр высокопроизводительных вычислительных систем (ЦВВС) Пермского национального исследовательского политехнического университета (ПНИПУ) имеет лицензию на использование этого коммерческого программного продукта;

– имеется возможность распараллеливания расчетов, т.е. использования всех преимуществ вычислительного кластера ПНИПУ;

– доступно простое и мощное средство автоматизации численного анализа – язык APDL, что существенно упрощает решение задач инженерного анализа;

– подробное руководство пользователя содержит примеры решения конкретных задач;

– опубликован и доступен значительный объем справочной литературы.

Вычислительные ресурсы ЦВВС ПНИПУ предоставляют возможность проводить параллельные вычисления. Это позволяет повысить эффективность процесса решения больших и очень больших задач, значительно сокращая время проведения расчетов (рис. 1).

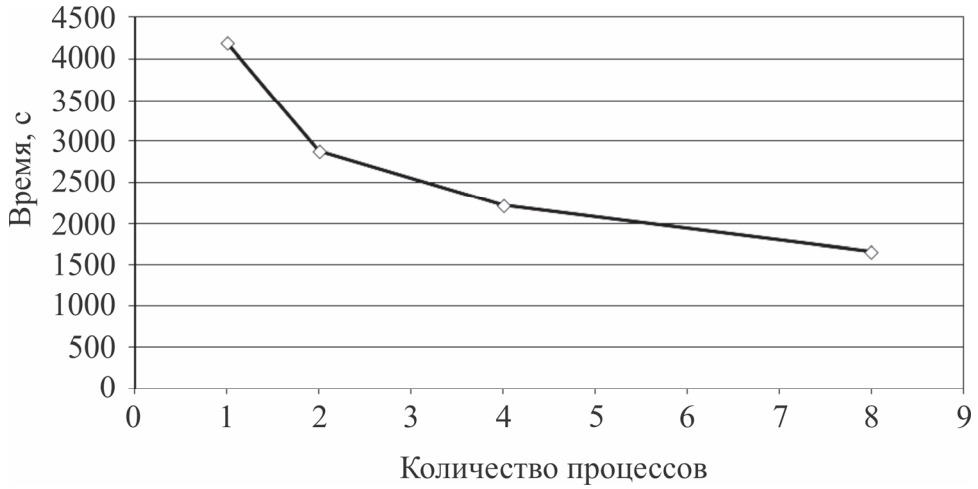
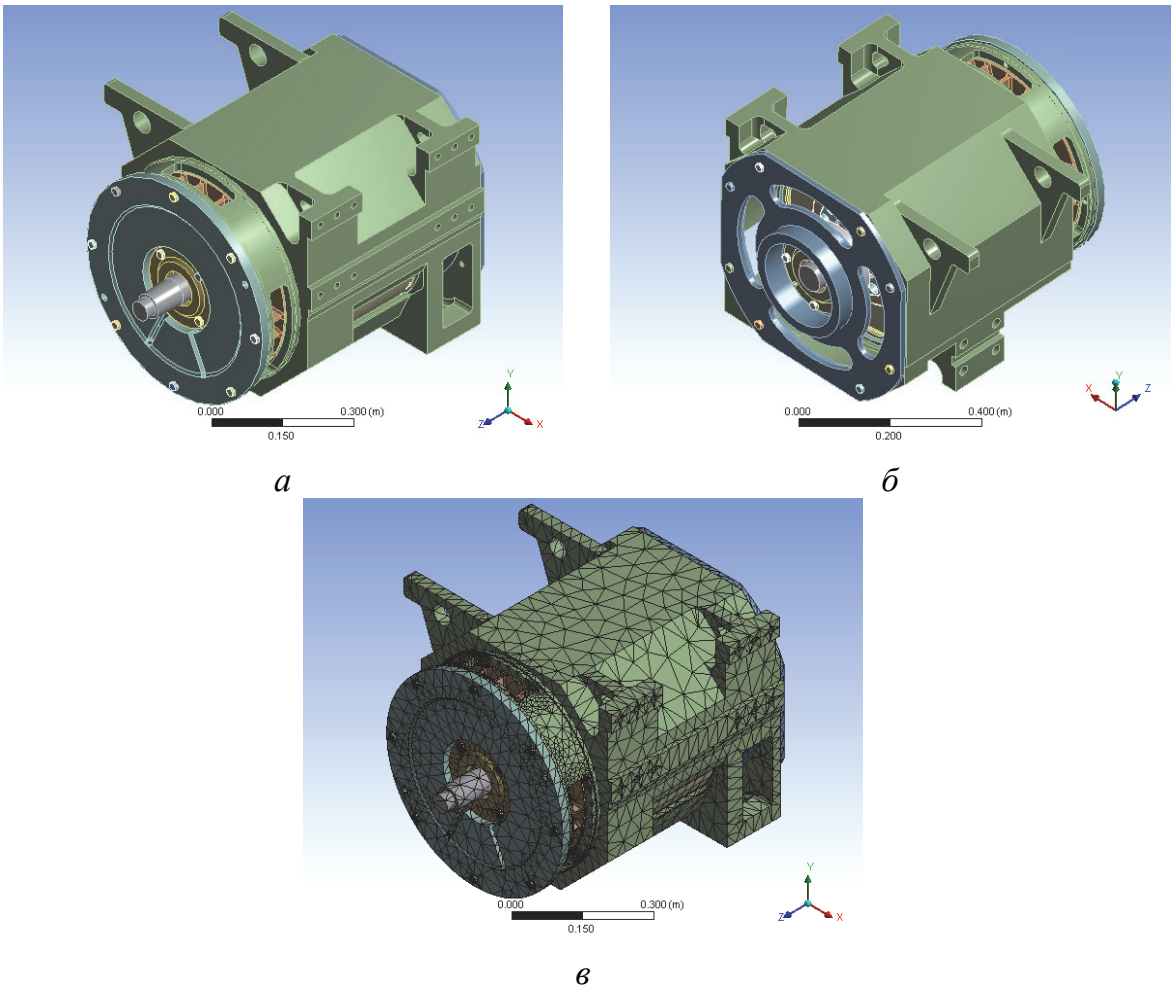


Рис. 1. Зависимость времени проведения расчета электродвигателя на ударную прочность от количества процессоров

Для аппроксимации областей, занимаемых отдельными составляющими двигателя (рис. 2), используются пространственные (трехмерные) конечные элементы SOLID186, наиболее часто применяющиеся для решения задач механики деформируемого твердого тела, которые реализуют

квадратичную аппроксимацию поля перемещения в пределах занимаемой области. Конечные элементы указанного типа содержат по 20 узлов, каждый из которых имеет 3 степени свободы (перемещения в направлении каждой из координатных осей).



а *б*

в

Рис. 2. Геометрическая модель двигателя: со стороны привода (*а*) и с противоположной стороны (*б*); аппроксимация с использованием 74 865 конечных элементов (*в*)

Характеристики материалов, применяемых для моделирования: плотность пластин статора 6500 кг/м^3, плотность остального материала деталей двигателя 7850 кг/м^3, масса двигателя в сборе 750 кг; коэффициент Пуассона 0,3, модуль упругости 210 ГПа, предел прочности 460 МПа, предел текучести 250 МПа. На рис. 3 приведена зависимость циклической прочности металла от числа N циклов переменного нагружения.

Вентилятор, пластины ротора и балансировочные кольца жестко закреплены на валу и не допускают перемещений относительно друг друга. Ротор в сборе сбалансирован. Вентилятор имеет незначительный дисбаланс относительно продольной оси, который компенсируется дополнительными грузами при проведении динамических балансировочных испытаний.

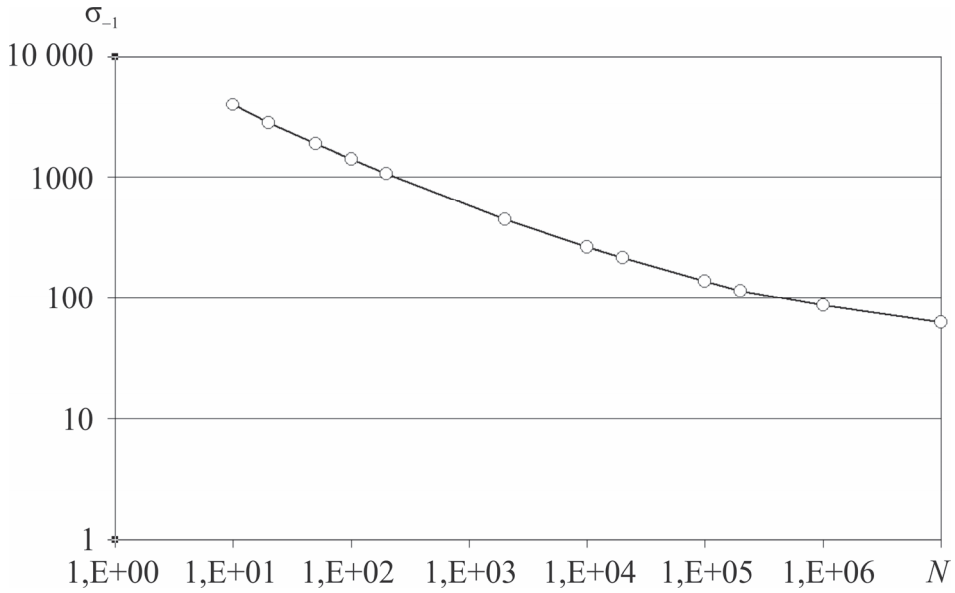


Рис. 3. Диаграмма длительной прочности:
зависимость предела прочности σ_{-1} (МПа) от числа *N* циклов

Коэффициент трения качения между поверхностями подшипника принят равным 0,01. Для прочих контактирующих поверхностей корпуса и деталей принят коэффициент трения покоя, равный 0,15 [3]. Болтовые соединения считаются жесткими, не допускают перемещений.

Граничные условия по закреплению модели двигателя ДАТ-170-4У2 соответствуют условиям закрепления прототипа на испытательном стенде.

С использованием модуля модального анализа программного комплекса ANSYS определены резонансные частоты отдельных элементов электродвигателя ДАТ-170-4У2 и двигателя в целом. Собственные частоты, определенные численно и экспериментально, представлены в таблице.

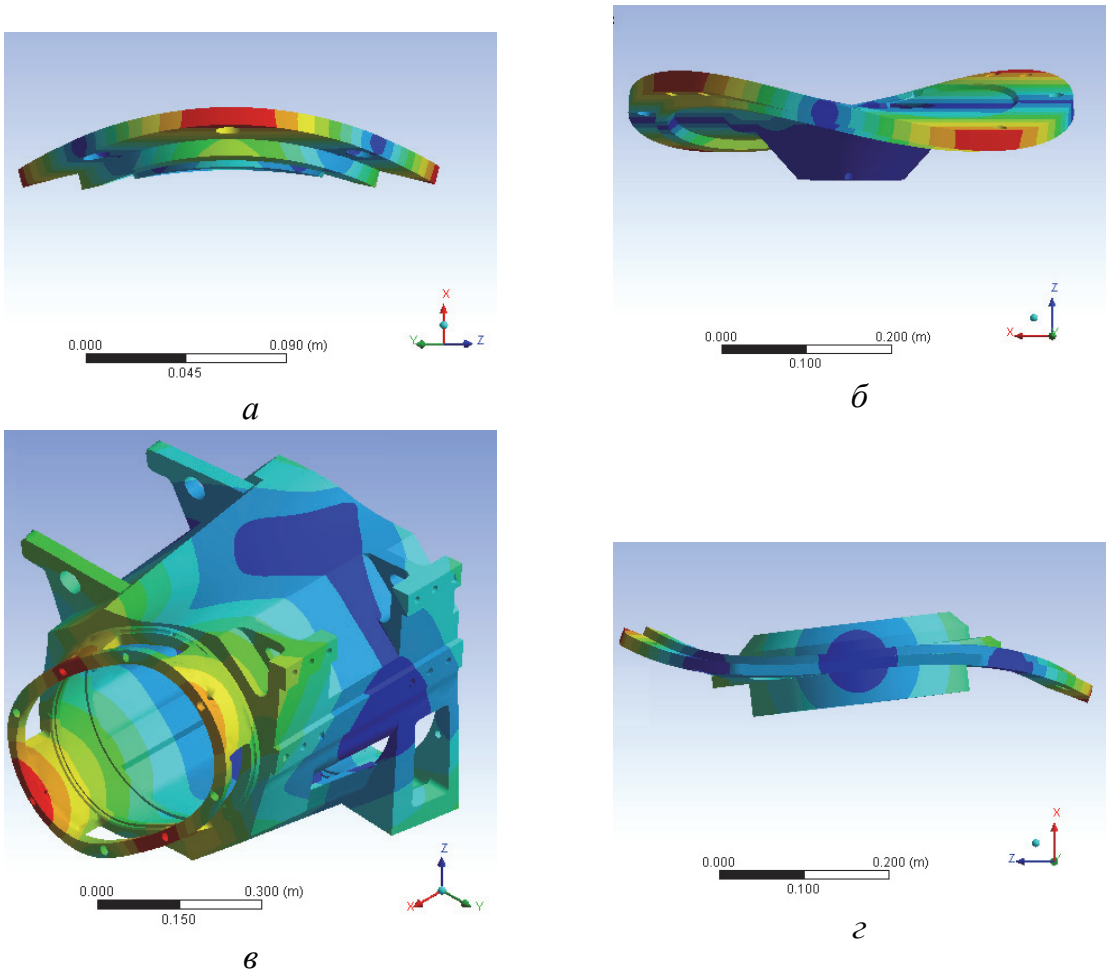
Низшие собственные частоты отдельных деталей двигателя ДАТ-170-4У2

Название элемента двигателя	Низшая собственная частота, Гц		
	без учета закрепления	с учетом закрепления	по протоколу испытаний
Крышка подшипникового щита со стороны, противоположной приводу	1808,2	397,4	372,0
Подшипниковый щит со стороны привода	571,8	453,8	625,0
Подшипниковый щит со стороны, противоположной приводу	297,5	237,5	297,0
Двигатель в сборе	297,5	237,0	–

Наименьшая резонансная частота всего двигателя в сборе составила 237 Гц. Данная частота значительно выше номинального режима работы

21 Гц, превышает максимальную установленную предприятием-изготовителем частоту 60 Гц (3600 об/мин) и превышает верхнюю частоту диапазона требований (100 Гц).

Формы собственных колебаний отдельных частей электрического двигателя представлены на рис. 4.



а *б* *в* *г*

Рис. 4. Формы собственных колебаний элементов электродвигателя: крышка подшипника (*а*, 1-я форма), передний щит (*б*, 2-я форма), корпус (*в*, 3-я форма), задний щит (*г*, 4-я форма)

С использованием модуля гармонического анализа программного комплекса ANSYS выполнены вычислительные работы по исследованию длительной прочности электродвигателя ДАТ-170-4У2 для следующих значений границ октав регламентированного диапазона (Гц): 10, 20, 40, 80, 160.

На рис. 5 представлены результаты выполнения вычислительного эксперимента по моделированию напряженно-деформированного состояния отдельных элементов двигателя при исследовании вибропрочности. Наибольшее эквивалентное напряжение (по Мизесу) за единичный вибрационный цикл в заднем щите достигает величины 26,06 МПа и при испытании на частоте 160 Гц при амплитуде ускорения 30 м/с^2.

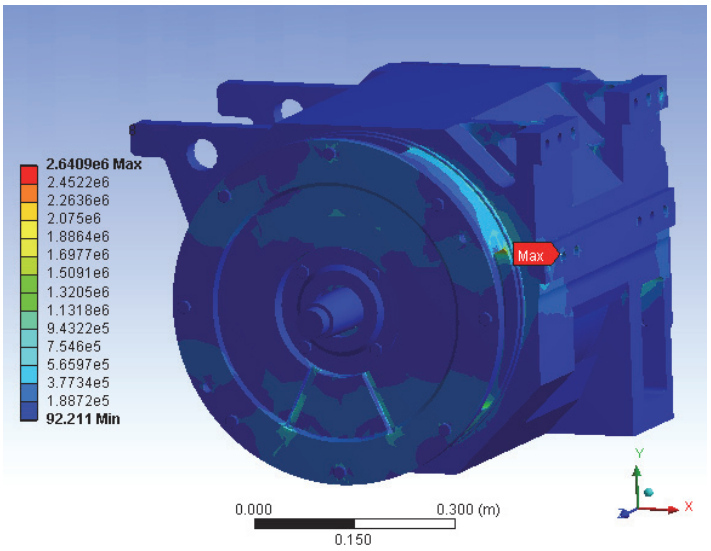


Рис. 5. Результат испытаний электродвигателя
на воздействие длительной вибрации

Согласно данным о длительной прочности материала количество N циклов до разрушения, соответствующее указанному эквивалентному напряжению, составит не менее 10^7, что превышает 50 млн 56 тыс. циклов, выполненных при проведении натурных виброиспытаний [4].

Следует ожидать, что рассмотренную вибрационную нагрузку элементы двигателя ДАТ-170-4У2 выдержат без разрушения.

С использованием модуля динамического (транзиентного) анализа программного комплекса ANSYS выполнены вычислительные эксперименты по исследованию ударной прочности при нанесении одиночных ударов по электродвигателю ДАТ-170-4У2 (вдоль каждой из координатных осей).

Экспериментальные кривые ударных импульсов представлены в протоколе испытаний [4] и аппроксимировались квадратичной параболой вида

$$a = 400\left[\frac{t}{6\cdot 10^{-3}} - \frac{t^2}{36\cdot 10^{-6}}\right] \qquad (1)$$

при заданной амплитуде ускорения 100 м/с^2 и длительности 6 мс (ГОСТ 16962.2–90 [2]).

Для задания граничных условий при моделировании в перемещениях выполнено интегрирование выражения (1) и получены зависимости от времени скорости и перемещения (рис. 6).

$$v = 400\left[\frac{t^2}{12\cdot 10^{-3}} - \frac{t^3}{108\cdot 10^{-6}}\right];$$

$$u = 400\left[\frac{t^3}{36\cdot 10^{-3}} - \frac{t^4}{432\cdot 10^{-6}}\right].$$

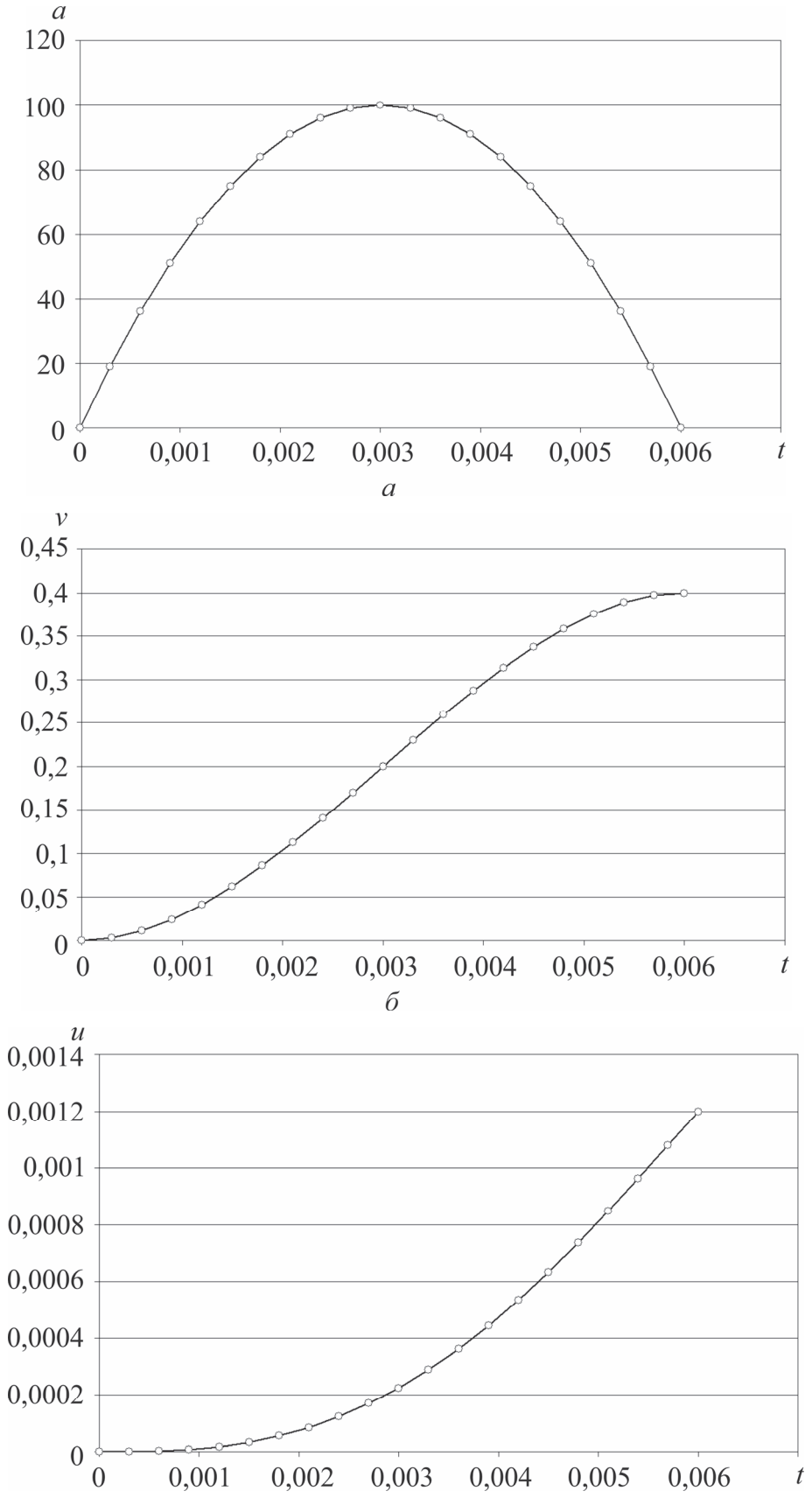


Рис. 6. Зависимости ускорения *а* (*а*, м/с^2), скорости *v* (*б*, м/с) и перемещения *и* (*в*, м) при моделировании одиночного удара

Результаты вычислительного эксперимента по моделированию одиночных ударов по двигателю представлены на рис. 7. Наибольшее эквивалентное напряжение (по Мизесу) при единичном ударе наблюдается в вентиляторе и составляет 191,05 МПа.

Согласно данным о свойствах материала предел текучести на сжатие (растяжение) составляет $\sigma_т$ = 250 МПа, т.е. превышает достигаемое при ударе значение эквивалентного напряжения.

Рассмотренную ударную нагрузку элементы двигателя ДАТ-170-4У2 выдерживают без разрушения.

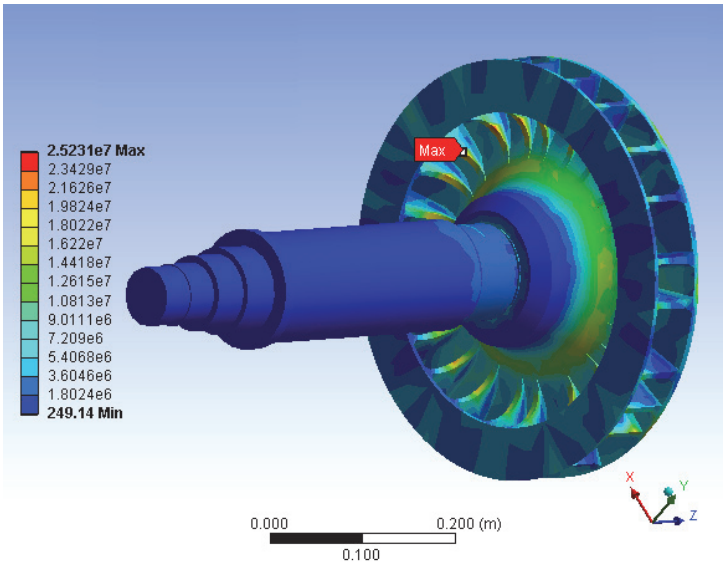


Рис. 7. Результат испытаний электродвигателя
при воздействии механического удара одиночного действия

Выводы

Обоснована возможность замены динамических механических испытаний электродвигателя вычислительным моделированием с использованием программного комплекса ANSYS. Использование параллельных вычислений на базе ЦВВС ПНИПУ обеспечило принципиальную возможность проведения данного исследования в заданные сроки с высоким качеством.

Список литературы

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.

2. ГОСТ 16962.2–90. Изделия электротехнические. Методы испытаний на стойкость к механическим внешним воздействующим факторам / Мин. электротехн. пром. и приборостроения СССР. – М., 1990. – 55 с.

3. Кухлинг Х. Справочник по физике. – М.: Мир, 1985. – 520 с.

4. Двигатель асинхронный тяговый ДАТ-170-4У2. Протокол испытаний на стойкость к механическим внешним воздействующим факторам. ТХ.221.2787 / Гос. предпр. завод «ЭЛЕКТРОТЯЖМАШ». – Харьков, 2012. – 16 с.

5. Гергель В.П. Теория и практика параллельных вычислений / Интернет-ун-т информац. технологий. – М.: БИНОМ. Лаборатория знаний, 2010. – 423 с.

Д.С. Бугаев, В.В. Муромцев

Национальный исследовательский университет
«Белгородский государственный университет»

ЭЛЕКТРОННЫЙ СТЕТОФОНЕНДОСКОП, ПОДДЕРЖИВАЮЩИЙ ДИАГНОСТИКУ СЕРДЕЧНО-СОСУДИСТОЙ СИСТЕМЫ

Рассматриваются преимущества и недостатки электронного фонендоскопа, ставится задача разработки прототипа электронного фонендоскопа, поддерживающего диагностику сердечно-сосудистой системы, а также подбирается специальная графическая среда программирования LabView и алгоритм динамического трансформирования времени DTW для решения поставленной задачи.

Старейшим методом диагностики заболеваний сердечно-сосудистой системы (ССС) является аускультация, которая заключается в выслушивании звуков, образующихся в процессе функционирования сердца. Выслушивание сердца было впервые введено во II в. до н.э. греческим врачом Аретеем и до сих пор широко используется во врачебной практике, что обусловлено простотой метода [1].

Различают прямую аускультацию, т.е. выслушивание непосредственно ухом, приложенным плотно к телу больного, и непрямую аускультацию – выслушивание с помощью приборов для аускультации.

Аускультацию как диагностический метод впервые применил французский врач Рене Лаэннек. В 1819 г. он описал основные аускультативные феномены и дал им название. Для проведения аускультации Рене Лаэннек использовал первый прибор для аускультации – стетоскоп, представляющий собой трубку, имеющую на концах раструбы в виде воронок, один из которых прикладывается к телу обследуемого, другой – к уху врача. Позднее устройство стетоскопа претерпело ряд изменений. Был изобретен фонендоскоп, имеющий натянутую мембрану для усиления звука, и стетофонендоскоп, имеющий на одном конце наконечник фонендоскопа с мембраной, а на другом – наконечник стетоскопа без мембраны [1]. Стетоскопы,

фонендоскопы и стетофонендоскопы относятся к классу акустических приборов и состоят из наконечника, воспринимающего звуковые колебания с участка тела, и звукопровода, представляющего собой жесткую или гибкую трубку (или две трубки), конец которой направляется в ухо.

Можно выделить следующие недостатки акустических приборов:

1. Низкий уровень полезного звукового сигнала.

2. Требуется герметизация в звукопроводе. Для достижения герметичности в ушах требуется подбор типоразмера герметизирующих олив.

3. Искажения, связанные с наличием звукопровода.

4. Ограничение на выбор частотного диапазона исследуемых звуков. При использовании стетоскопа хорошо выслушиваются низкочастотные звуки. Для выслушивания высокочастотных признаков используют фонендоскоп.

От перечисленных недостатков свободны электронные приборы для аускультации. Их отличие от акустических приборов состоит в том, что звуковые сигналы, воспринятые с участка тела, преобразуются с помощью микрофона в электрические сигналы, усиливаются, подвергаются обработке и выводятся на наушники. Обработка сигналов в основном заключается в подавлении шумов окружающей среды и фильтрации, которая заключается в выделении определенных диапазонов частот. Аускультативные признаки подразделяют на низко-, средне- и высокочастотные с диапазоном частот соответственно: от 20 до 180 Гц, от 180 до 710 Гц и от 710 до 1400 Гц.

В данной работе рассматривается прототип электронного стетофонендоскопа с расширенной функциональностью. Основной функцией является возможность диагностики ССС.

Прототип электронного стетофонендоскопа создан на базе нетбука IRU Intro 107 с операционной системой Windows 7. Можно использовать любой ПК, со следующими характеристиками:

– процессор «Пентиум 4» или эквивалетный ему,

– оперативная память – 1 Гб,

– операционная система – Windows 8/7 (32- и 64-разрядная).

Требования к ПК определяются используемой инструментальной системой LabVIEW, которая представляет собой высокоэффективную среду графического программирования, предназначенную для создания гибких и масштабируемых приложений.

Для съема звуковых сигналов с участка тела использовались наконечники стетоскопа с односторонней головкой ST71 и стетофонендоскопа Раппапорта ST77 фирмы Microlife, а также микрофонов WM-61A [1]. Внешний вид датчиков представлен на рисунке. В качестве наушников использовались SHP2500/10 (15–22 000 Гц). Еще необходима внешняя звуко-

вая карта, так как внешние звуковые карты лишены таких недостатков, как задержка звука и ограничения пропускной способности звуковых данных от компьютера к звуковой плате, которая вставляется через разъем USB.

Программное обеспечение. Для разработки программного обеспечения использовано инструментальное средство LabVIEW. Программное обеспечение включает в себя:

– программу регистрации сигналов фонокардиографических исследований (ФКГ),

– программу анализа сигналов ФКГ.

Программа регистрации сигналов ФКГ представляет собой удобный интерфейс, предназначенный для прослушивания звуков сердца и его последующей записи в аудиофайл. Вторая программа используется для анализа записанных звуков на основе уже имеющейся базы патологий ССС.

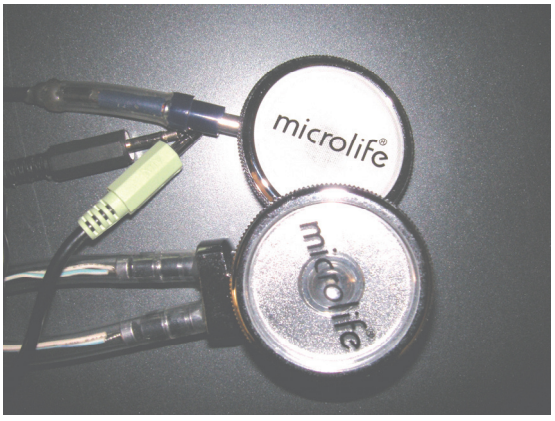


Рис. Звуковые датчики

Функционал программы состоит из следующих элементов:

1. Исходный график сигнала.
2. Различные фильтры.
3. График после процесса фильтрации.
4. Преобразованный график, предназначенный для записи.

Работу программы можно поделить на несколько этапов:

1. Настройка программы.
2. Запись звукового сигнала.
3. Получение сигнала с микрофонного датчика.
4. Расчет спектральных характеристик сигнала.
5. Фильтрация сигнал.
6. Подача звукового сигнала на наушники.

Дальнейшие исследования будут направлены на создание диагностического программного обеспечения, которое будет включать в себя различные функции:

– ведение базы данных результатов, с последующим использованием в медицинских целях;

– вывод осциллограммы на дисплей ПК,

– запись сердцебиения пациента в файл с разрешением wav,

– возможность выдачи диагноза сердечно-сосудистого заболевания.

При решении последней задачи мы столкнулись с проблемой сравнения звуков с образцовыми звуковыми файлами. По предварительным оценкам наиболее перспективным для решения этой задачи является метод динамического трансформирования времени (DTW) [4]. Метод предполагает представление звука в виде зависимости давления от времени, а далее разбивает исходный сигнал на синусоиды с различными частотами, амплитудами и фазами.

Алгоритм динамического трансформирования времени вычисляет значения деформации между двумя временными рядами [2, 3]. Это делается для того, чтобы разбить звуковые волны на синусоидальный график. Алгоритм динамического трансформирования времени вычисляет оптимальную последовательность трансформации (деформации) времени между двумя временными рядами. Алгоритм вычисляет оба значения деформации между двумя рядами и расстоянием между ними, а если сложить эти синусоиды, то мы получим звуковую волну на синусоидальной кривой с различными частотами, амплитудами и фазами. Этот процесс называется анализом Фурье. Далее складываются эти синусоидальные кривые, и мы получаем звуковую волну. Данный алгоритм предназначен для обработки речи. Это происходит в процессе сравнения полученного сигнала с эталонным. При этом требуются большие объемы машинной памяти. При небольшом наборе данных эта проблема преодолима и алгоритм может быть реализован. Но так как особого разнообразия звуков здесь не будет и этот алгоритм является очень точным при небольшом наборе данных, то мы решили использовать его в обработке биения сердца.

В результате нами был получен прототип стетофонендоскопа, поддерживающего диагностику сердечно-сосудистой системы на базе персонального компьютера. Следующим этапом нашего исследования будет обработка сигнала и ведение регистрационной карты пациента, а в дальнейшем – реализация устройства на базе мобильного телефона.

Список литературы

1. Электронный фонендоскоп и его функциональность [Электронный ресурс]. – URL: http://pochemuha.ru/eljektronnyj-fonjendoskop-i-jegounkcionalnost.

2. Стетоскопы, фонендоскопы. Продукция [Электронный ресурс]. – URL: http://www.dealmed.ru/stetoskopy.html.

3. National Instruments Россия [Электронный ресурс]. – URL: http://www.labview.ru.

4. Динамическое программирование в алгоритмах распознавания речи [Электронный ресурс] – URL: http://habrahabr.ru/post/135087.

А.Н. Волкович

Объединенный институт проблем информатики Национальной академии наук Беларуси

ИСПОЛЬЗОВАНИЕ GPGPU В СИСТЕМАХ КОМПЬЮТЕРНОГО ЗРЕНИЯ РЕАЛЬНОГО ВРЕМЕНИ

Рассмотрена реализация стереодальнометрической системы компьютерного зрения. Приведены ограничения, налагаемые на систему при ее фактической реализации, и предложены способы их разрешения. Произведена реализация и сравнение функционирования системы в CPU- и GPGPU-вариантах.

Введение

Вопрос определения положения объектов в пространстве находит свое отражение во многих прикладных задачах. В общем случае, учитывая тот факт, что положение условного наблюдателя и направление наблюдения известны в некоторой системе координат, задача сводится к вычислению расстояния до объекта интереса, основываясь на котором возможно вычисление точных координат исследуемого объекта (прямая геодезическая задача). Также следует отметить, что существует ряд задач, предполагающих невозможность применения активных дальнометрических систем (лазерные дальномеры и т.п.). Таким образом возникает необходимость вычисления расстояний до объекта на основе изображений.

Ввиду невозможности определения дистанции до объекта на основе одного изображения возможно применение методов, основанных на стереопараллаксе, которые достаточно хорошо зарекомендовали себя в системах оптических стереодальномеров.

Решение стереодальнометрической задачи

Непосредственно сам процесс определения дистанции до объекта на основе стереоснимков можно обобщенно описать при помощи следующей теоретической модели. Оптическую систему можно представить геометрически в виде треугольника (рис. 1).

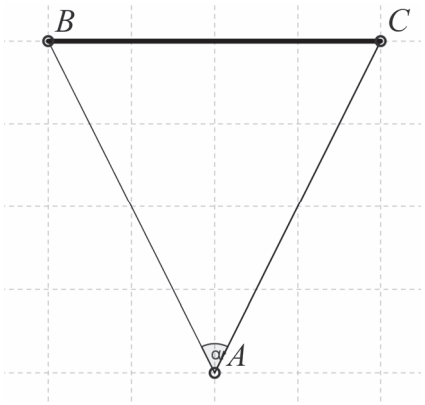


Рис. 1. Упрощенная геометрическая модель камеры:
A – точка наблюдения; BC – изображение;
α – горизонтальный угол раскрытия объектива

Задача вычисления расстояний до объекта представляет собой задачу решения треугольника (рис. 2).

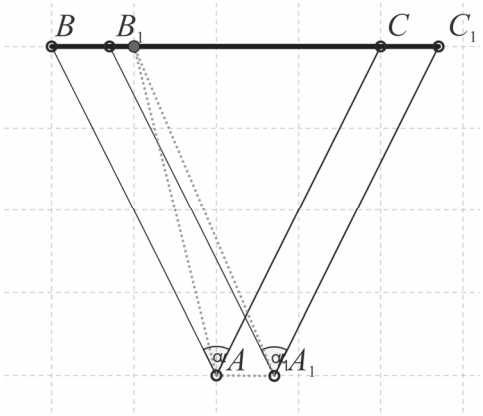


Рис. 2. Геометрическая модель стереосистемы:
A, A_1 – точка наблюдения; BC – левое изображение;
B_1C_1 – правое изображение; B_1C – зона перекрытия;
α – горизонтальный угол раскрытия объектива

В рамках системы известно основание треугольника – база стереосистемы. Углы при основании, в случае малодисторсионной оптики, могут быть вычислены через угол раскрытия объектива. Далее по теореме синусов возможно определение длин сторон A_1E и AE.

Таким образом, одной из главных и сложных задач в процессе стереорекострукции является поиск соответствия точек на проекциях. Можно утверждать, что задача предполагает определение для точки некоторых характеристик, которые бы уникальным образом характеризовали точку изображения, причем таким образом, чтобы парная точка на опорном изображении обладала идентичными или максимально близкими значениями аналогичных характеристик.

Поскольку изображения реального мира включают в себя достаточно узкий набор цветов или яркостей, рассматривать значения цвета или яркости точек изображения, сопряженных при решении задачи определения сопряженных точек, нельзя. При решении поставленной задачи речь идет об отождествлении не отдельных точек, а фрагментов изображений, лежащих в окрестностях этих точек. Таким образом, крайне важно знать, если на одном изображении взята какая-то точка, где на втором изображении находится ей сопряженная. И каким образом сопоставить эти фрагменты.

Сопоставление окрестностей сопряженных точек не поддается строгой формализации, поскольку в его основе лежит проблема идентификации по изображениям фрагментов трехмерного мира, которые с трудом поддаются адекватному формальному описанию. Возникающие при съемке и зависящие от ракурса проективные и яркостные искажения приводят к тому, что в изображениях одних и тех же участков поверхности, снятых под разными ракурсами, могут появиться значительные отличия. Принципиально важно то, что эти отличия зависят не только от геометрии съемки, но и от геометрических и физических характеристик самой поверхности. Расположение источника света по отношению к поверхности влияет на распределение освещенности. Положение элементов поверхности, их отражательные и рассеивающие свойства определяют количество энергии, попадающей в объективы камер, а следовательно, и локальные отличия в яркости сопряженных фрагментов изображений.

Величина различий зависит от разницы в ракурсах съемки. Как правило, чем больше эта разница (в частности, чем больше база), тем менее похожими становятся изображения. Поэтому все методы сопоставления окрестностей сопряженных точек в большей или меньшей степени опираются не на формальный подход, а на изучение характера изображений, на возможность их предварительной обработки, на приведение к эпиполярной стереопаре, на построение эффективных по точности и быстродействию описателей окрестностей сопряженных точек для сопоставления.

При сопоставлении особенностей точек, для принятия решения о том, соответствуют ли друг другу характерные точки или нет, сравнивается именно окружение этих точек, с использованием окна (окрестности) фиксированного или адаптируемого размера. В данном классе алгоритмов стоимость сопоставления пикселов двух изображений определяется как схожесть окрестностей этих пикселов в изображениях.

В идеализированной ситуации, значения функции схожести в процессе сканирования вдоль строки должны представлять собой точечный выброс для искомого пиксела, при возврате нулевого значения подобия для всех остальных пикселов (окрестностей) строки.

Безусловно, при работе с реальными графическими данными такая комбинация возвращаемых функцией сходства значений невозможна. Наличие различий ракурсов, неидеальное наведение фокуса, с одной стороны, ограниченное число значений цвета и яркости – с другой стороны, приводит к появлению более сглаженной картины. В свою очередь, при обработке исходных данных, обладающих достаточным количеством информации для идентификации локального региона, в функции сохраняется достаточно явный экстремум, позволяющий идентифицировать искомый пиксел изображения (рис. 3).

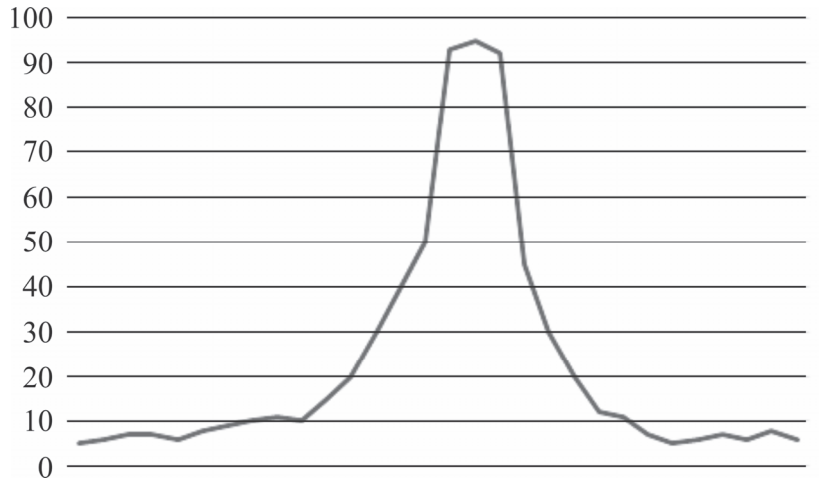


Рис. 3. Функция подобия региона с ярко выраженным экстремумом

Таким образом, главной задачей при построении карты диспаратности является подбор варианта сравнения регионов, при котором экстремум значений функции подобия будет наиболее выраженным.

Ограничения, возникающие при реализации метода

К сожалению, оптоэлектронные системы, генерирующие сигнал высокого качества, весьма чувствительны к температурным, вибрационным, гидродинамическим, пылевым характеристикам и зачастую не могут быть использованы в системах, к которым предъявляются повышенные требования «живучести».

Рассматривая системы, использование которых предполагает сохранение работоспособности в сложных условиях, можно говорить о возможности применения камер технического зрения, которые в свою очередь обладают более низкими качественными характеристиками по динамическому диапазону, передаче полутонов и/или оттенков. Ввиду того, что стандартные корреляционные функции являются чрезвычайно чувствительными к качеству и параметрам исходных данных (экспозиция, бликование, переэкспонирование, недоэкспонирование, шум матрицы, случайные вы-

бросы), необходим поиск методов дополнительной предобработки изображений. Применение градиентных измерений позволяет снизить чувствительность к различиям между настройками в цветопередаче камер, однако данный подход может приводить к утрате различных мелких малоконтрастных деталей, которыми, в принципе, возможно пренебречь в случае необходимости вычисления дистанции до некоторого объекта с определенной погрешностью при плохих условиях съемки с низким качеством в пользу общей стабильности функционирования системы.

Описанный выше процесс представляет собой решение задачи вычисления производной динамики яркости в окрестности некоторой точки с возможной бинаризацией – выделение контуров. Данная задача требует достаточно больших затрат времени при обработке изображений с разрешения более 720 пикселов (4–6 с на обработку кадра), что неприемлемо для систем, позволяющих производить обработку изображений как «по щелчку», так и с обработкой в реальном времени с частотой 25 fps.

Обработка с предварительным выделением контуров

Выделение контура предполагает определение точки изображения как точки перепада яркости в том случае, если ее двумерная производная первого порядка превышает некоторый заданный порог. Связывание точек контура состоит в анализе характеристик пикселов в небольшой (3×3 или 5×5) окрестности каждой точки (x, y) изображения, которая была отмечена как контурная точка (точка перепада).

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 для каждой точки. Один из способов нахождения первых частных производных в конкретной точке состоит в использовании градиентного оператора Собеля. Для оператора Собеля, который выявляет горизонтальные и вертикальные контуры (перепады яркости), следует определить соответствующие маски для свертки с исходным изображением. Также можно изменить приведенные формулы таким образом, чтобы они давали максимальный отклик для контуров, направленных диагонально. Для каждой из масок сумма коэффициентов равна нулю, таким образом, данные операторы будут давать нулевой отклик на областях постоянной яркости, что характерно для дифференциального оператора. Рассмотренные маски применяются для получения составляющих градиента Gx и Gy. Для вычисления величины градиента эти составляющие необходимо ис-

пользовать совместно, определяя величину градиента приближенно через абсолютные значения частных производных.

Реализация и сравнение производительности систем на CPU и GPU

Как видно из описания, метод обладает большим числом циклических операций, которые в свою очередь являются локально независимыми, что дает возможность говорить о возможном распараллеливании алгоритма согласно так называемой концепцией «параллелизма по данным».

В большинстве случаев системы, требующие оперативной работы, предполагают их размещение на мобильных платформах. Безусловно, мобильное размещение кластерных систем маловозможно, отсюда возникает необходимость использования иных систем параллельных вычислений, к которым можно отнести GPGPU-решения.

Поскольку авторами рассматривалась задача реализации дальнометрической системы компьютерного зрения в разрезе ее фактического применения, было принято решение производить ее реализацию с расчетом не на высокомощные GPGPU-платформы, а на технологические решения среднего класса, которые могут быть размещены на реальных программно-аппаратных системах, в том числе и мобильных (использующих ноутбуки в качестве вычислительной машины).

В качестве тестовой системы был использован персональный компьютер с графическим ускорителем NVidiaGeForceGTS-450 и было произведено сравнение выполнения полного цикла решения дальнометрической задачи на CPU и GPU для входного видеопотока различного разрешения (таблица).

Результаты обработки

Разрешение, пикселы	CPU	GPU
640×480	2 с	0,008–0,009 с
720×576	3 с	0,009 с
1024×720	6 с	0,014 с
1440×1080	10 с	0,02 с
1920×1080	12 с	0,025 с

Сравнение производительности осуществлялось на работающей системе в течение часа с получением данных о времени обработки кадра. Система на основе GeForceGTS-450 позволяла производить обработку и вычисления со стабильной скоростью около 40 кадров в секунду, что в свою очередь позволяло системе, получающей данные с частотой 25 fps, не находиться в режиме постоянной пиковой загрузки, что положительно сказалось на стабильности работы системы в целом.

Заключение

В ходе работ авторами произведена разработка программной дальнометрической системы компьютерного зрения, ориентированной на применение в системах реального времени. За счет использования средств GPGPU была достигнута производительность, позволяющая сформировать определенный задел по загрузке мощностей при переходе к исходным данным более высокого разрешения.

Список литературы

1. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман; пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2006. – 752 с.

2. Система автоматической реконструкции трехмерных сцен по нескольким изображениям / В.В. Ляковский, А.Н. Волкович, Д.В. Жук, А.В. Тузиков // Материалы V Белорусского космического конгресса, 25–27 октября 2011 г. – Минск, 2011. – Т. 2. – С. 129–133.

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

Р.Ф. Гибадуллин

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

ПАРАЛЛЕЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ КАРТОГРАФИЧЕСКИХ СЦЕН С АССОЦИАТИВНОЙ ЗАЩИТОЙ

Предлагаются алгоритмы модулей параллельной системы управления базами данных картографических сцен с ассоциативной защитой, описываются вычислительные процессы серверной части системы.

Введение

Статья является результатом проектной работы, направленной на создание и развитие принципов управления защищенными картографическими базами данных. Целью этой проектной работы является формирование новых методов управления защищенными картографическими базами данных, эффективных по критерию быстродействия.

Алгоритмы программных модулей серверной части системы

Блок-схема алгоритма процедуры выборки (модуль Sel) (рис. 1):

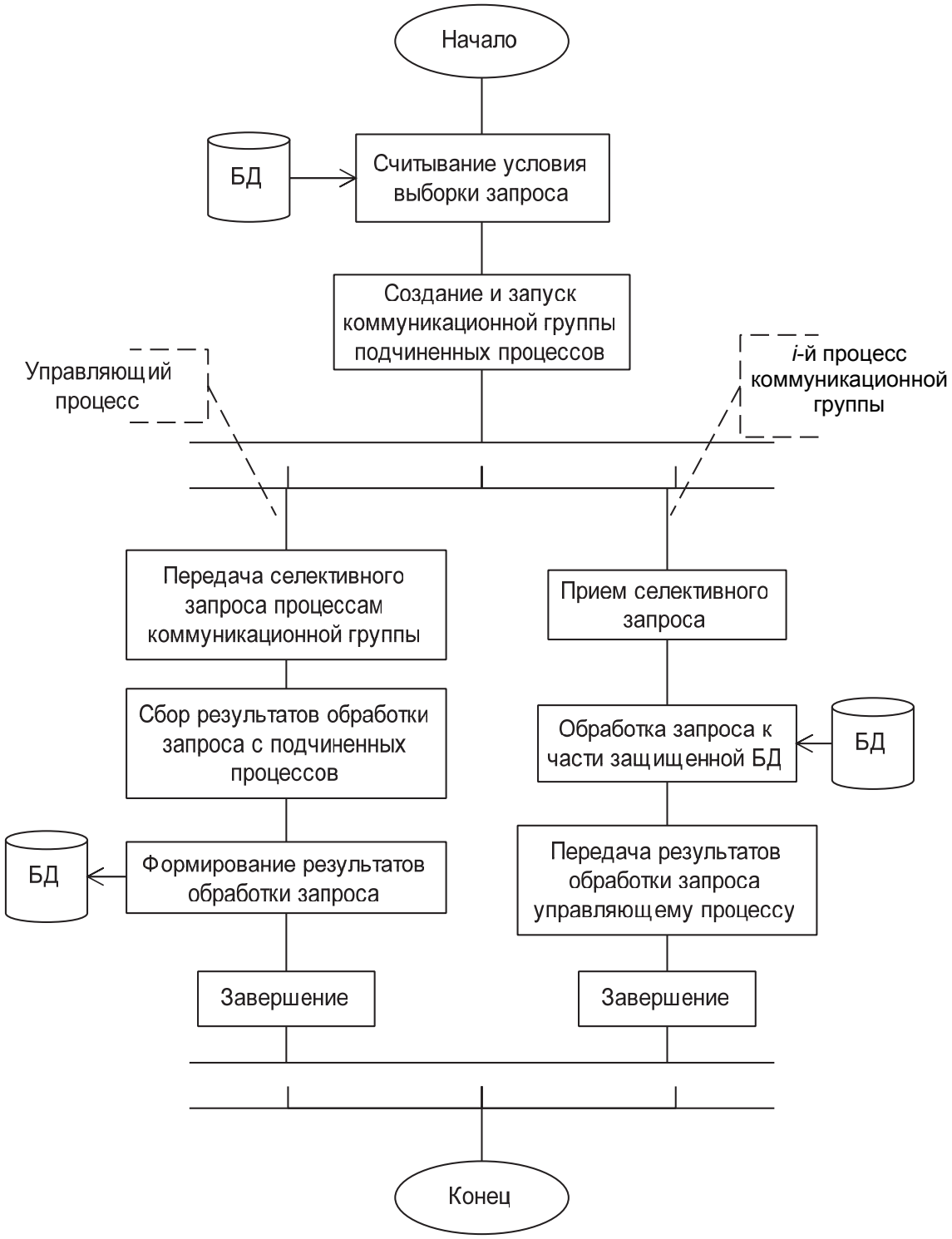


Рис. 1. Блок-схема алгоритма

Алгоритм процедуры добавления объекта (модуль Add):

1. Задаются атрибуты добавляемого объекта *q*: код тематического слоя, код объекта, координаты *x* и *y*.

2. Проверяется, не совпадают ли атрибуты добавляемого объекта *q* с атрибутами существующих объектов. Если совпадают, то выдается информация об ошибке и процедура прерывается.

3. Определяется фрагмент Q, в который добавляется q. Если такого фрагмента не существует, то его требуется создать, заполнить необходимым числом пустых объектов и непустым объектом q и завершить алгоритм.

4. Добавляется q во фрагмент Q.

5. Проверить, имеется ли в Q хотя бы один пустой объект. Если имеется, то следует удалить в Q любой пустой объект. Иначе – дополнить каждый фрагмент, кроме Q, одним пустым объектом. Данный шаг необходим для соблюдения равенства числа объектов в каждом фрагменте.

Алгоритм процедуры удаления объекта (модуль Del):

1. Задаются атрибуты удаляемого объекта q: код тематического слоя, код объекта, координаты x и y.

2. Опредяется фрагмент Q, в котором находится удаляемый объект q. Если такого фрагмента не существует, то следует выдать информацию об ошибке и прервать процедуру.

3. Проверяется, является ли q единственным непустым объектом в Q. Если является, то фрагмент Q удаляется. Иначе q заменяется пустым объектом.

Алгоритм процедуры изменения объекта (модуль Upd):

1. Задаются атрибуты изменяемого объекта q (код тематического слоя, код объекта, координаты x и y) и атрибуты объекта q' (код объекта, координаты x и y), которым требуется заменить q.

2. Определяется фрагмент Q, в котором находится изменяемый объект q. Если такого фрагмента не существует, то следует выдать информацию об ошибке и прервать процедуру.

3. Проверяется наличие изменяемого объекта q во фрагменте Q. Если такого объекта не существует, то следует выдать информацию об ошибке и прервать процедуру.

4. Определяется фрагмент Q', в который может переместиться изменяемый объект q. Если фрагмент Q' по новым координатам объекта q уже имеется, то следует перейти к шагу 7.

5. Если новые координаты объекта q не выходят за пределы фрагмента Q, следует изменить атрибуты объекта на новые, завершить процедуру.

6. Создается фрагмент Q' с необходимым числом пустых объектов и непустым объектом q', выполняется 8-й шаг алгоритма и завершается процедура.

7. Проверяется совпадение атрибутов объекта q' с атрибутами какого-либо объекта во фрагменте Q'. Если имеет место совпадение, то выдается информация об ошибке и процедура прерывается.

8. Проверяется, является ли объект q единственным непустым объектом во фрагменте Q. Если является, то удаляется весь фрагмент Q. Иначе объект q заменяется пустым объектом.

9. Добавляется объект q' во фрагмент Q'.

10. Проверяется, имеется ли во фрагменте Q' хотя бы один пустой объект. Если имеется, то следует удалить в Q' любой пустой объект. Иначе дополнить каждый фрагмент, кроме Q', одним пустым объектом. Данный шаг необходим для соблюдения равного числа объектов в каждом фрагменте заданного тематического слоя.

Перед запуском основного модуля сервера СУБД, который отвечает за управление вычислительными процессами системы, в папке data рабочей директории исполнительных модулей системы производится настройка конфигурационных файлов интерфейса MPI: mpich.conf и mpich_sel.conf.

В первых строчках файлов mpich.conf и mpich_sel.conf указываются места локализации mpi-модулей сокрытия и селекции соответственно. После слова "hosts" в обоих файлах перечисляются ip-адреса узлов и количество процессов, необходимых при запуске mpi-модулей, где первой строкой должен указываться ip-адрес главного узла системы.

По заданным настройкам конфигурационных файлов сокрытие данных будет проходить на узлах вычислительного кластера с запуском двух процессов на каждом. В выполнении процедуры селекции участвуют узлы системы, по которым распределялась базы данных картографических сцен с ассоциативной защитой (БД КС АЗ).

Перед стартом системы в рабочую директорию системы помещаются файлы ГИС MapInfo Professional, запускается модуль create_server_db для создания рабочих таблиц системы и запускается модуль create_db для формирования БД КС АЗ. По окончании работы этих модулей на серверной части системы будут созданы БД КС АЗ и база данных Server (рис. 2).

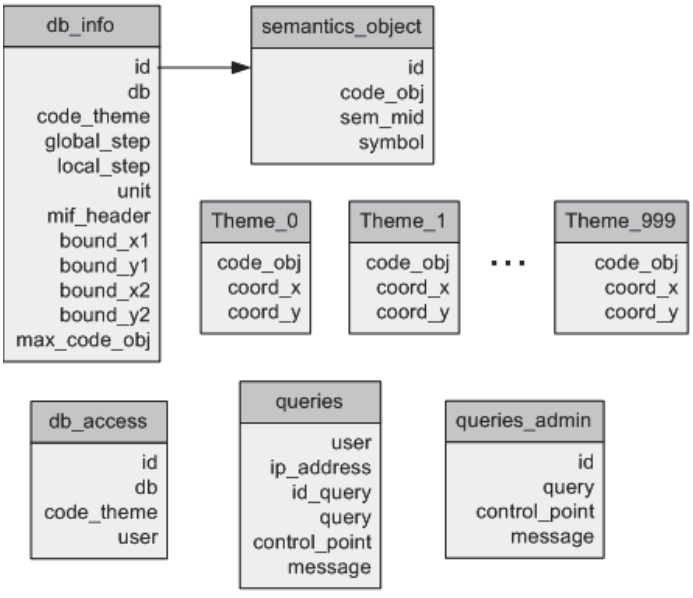


Рис. 2. База данных Server

Таблицы server.db_info и server.semantics_object используются для хранения семантической информации картографической сцены. Информация о максимальном коде объекта в таблице server.db_info хранится в сокрытом виде. Таблицы server.theme_0, server.theme_1, ..., server.theme_999 играют роль шаблонов и используются для синтаксического анализа запросов, вводимых администратором системы. Права владения информации о тех или иных тематических слоях карты назначаются пользователю администратором системы в таблице server.db_access. Таблицы server.queries и server.queries_admin используются для приема запросов от пользователя и администратора системы соответственно.

Динамические аспекты поведения процессов на серверной части СУБД представлены в виде диаграмм деятельности языка UML [1, 2].

Клиентская часть системы

Перед обращением клиента к серверу пользователь определяет фрагменты тематических слоев, которые понадобятся ему для работы. Например, если пользователю требуется работать с объектами с кодом 999 тематического слоя с кодом 9 и с объектами с кодом 777 тематического слоя с кодом 7, то ему необходимо в клиентском приложении указать следующие SQL-запросы:

"*SELECT * FROM Theme_9 WHERE code_obj=999*"

и "*SELECT * FROM Theme_7 WHERE code_obj=777*"

к БД со схемой на рис. 3.

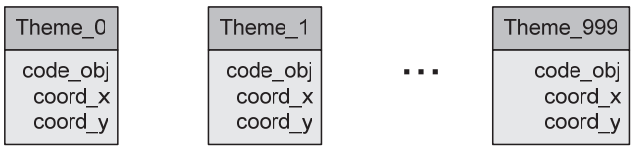


Рис. 3. Пользовательская схема базы данных

Клиентский модуль добавляет к запросам преамбулы ip-адрес клиентской машины, логин и id запроса (уникальный идентификатор запроса), заменяет константы (коды тематических слоев), указанные в запросах стегоконтейнерами, и включает модифицированный запрос в таблицу приема запросов queries, которая расположена на серверной части (рис. 4). Доступ к таблице queries клиентская программа получит лишь в том случае, если пользователь успешно подключится к MySQL-серверу нижнего уровня системы. После успешной авторизации пользователя таблица queries станет доступной клиентской программе.

После передачи запросов серверу, клиентская программа переходит в режим ожидания завершения обработки переданных запросов. Информацию об окончании обработки того или иного запроса клиентская програм-

ма получает посредством анализа флагов своей локальной таблицы ReadyData (рис. 5).

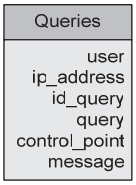


Рис. 4. Таблица queries

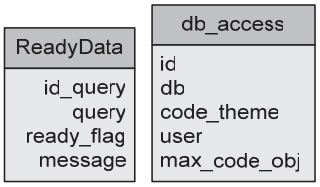


Рис. 5. Таблицы ReadyData и db_access

Сервер работает с БД КС АЗ, распределенной по узлам вычислительного кластера (случай монокластера). Схема распределенной БД КС АЗ представлена на рис. 6. Права владения информации о тех или иных тематических слоях карты назначаются пользователю администратором системы в таблице db_access со схемой (см. рис. 5). Доступность пользователю (user) конкретного слоя карты (code_theme) в БД (db) определяется наличием в таблице db_access кортежа с данными именем пользователя, кодом тематического слоя и именем базы данных.

Если пользователь обладает достаточными правами для выполнения запроса, сервер принимает запрос, проводит его сокрытие, выделяет в раскрытом запросе необходимые параметры и приступает к его обработке. По окончании обработки запроса сервер устанавливает соединение с MySQL-сервером клиента, заносит результат обработки в таблицы Themes, Frames, Objects БД клиента, присваивает значение 1 ячейкам столбца ready_flag и записывает сведения о результате и времени выполнения запроса в ячейку столбца message по id-запроса в таблице ReadyData. Если будут зафиксированы три неудачные попытки соединения с MySQL-сервером клиента (с интервалом ожидания 5 мин), то сервер завершает сеанс работы по данному запросу пользователя и удаляет строку, соответствующую этому запросу, в таблице queries.

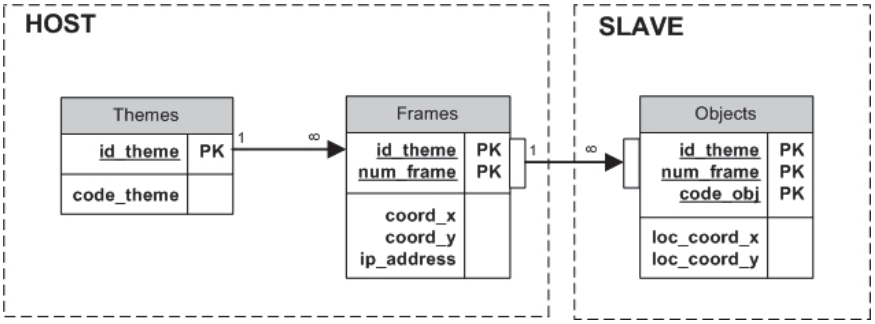


Рис. 6. Схема распределенной БД КС АЗ

При обнаружении двух одновременно работающих пользователей под одним учетным именем или ip-адресом сервер блокирует более раннюю сессию. Информация о том, на какой стадии выполнения (контроль-

ной точке) находится текущий запрос пользователя, фиксируется в таблице Queries БД сервера по атрибуту control_point.

В графическом окне клиентской программы доступна информация о готовности данных по тому или иному запросу пользователя. По готовности всех запрошенных данных пользователь может приступить к работе с ними с применением ГИС MapInfo Professional. Получив результат обработки запроса, клиентская программа извлекает из него требуемую информацию, раскрывает данные и формирует файлы формата MIF/MID.

Список литературы

1. Гибадуллин Р.Ф. Система баз данных картографии с ассоциативной защитой: автореф. дис. ... канд. техн. наук. – Уфа, 2011. – 16 с.

2. Unified Modeling Language™ (UML®) [Электронный ресурс]. – URL: http://www.uml.org.

Т.И. Габдуллин, П.В. Писарев

Пермский национальный исследовательский
политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ДЕФОРМАЦИЙ КОРПУСНОЙ ДЕТАЛИ И ОСНАСТКИ ПРИ ПРОВЕДЕНИИ ФИНИШНОЙ ОБРАБОТКИ

С целью снижения трудоемкости при изготовлении корпусной детали была спроектирована оснастка для обработки корпусной детали в буксирной установке MultiFinish. Данный вид современной финишной обработки позволяет заменить ручные слесарные операции. Произведен численный анализ напряженно-деформированного состояния корпусной детали и оснастки при разных режимах обработки с учетом упругости среды (наполнителя) в программном комплексе Ansys на высокопроизводительном вычислительном комплексе ПНИПУ.

На ОАО «Пермская научно-производственная приборостроительная компания» изготавливается большая номенклатура изделий, деталей. При механической обработке деталей, а именно при зачистной обработке, применяются ручные слесарные операции, что сопровождается большими временными затратами. В связи с этим возникает необходимость внедрения нового метода финишной обработки. Разработка нового технологического процесса, проектирование специализированной оснастки для фиксации детали позволит решить поставленную перед предприятием задачу внедрения установки и реализации технологии для конкретной детали.

В связи с этим была сформулирована следующая цель, которая заключается в анализе и проведении численного эксперимента по расчету максимальных напряжений и деформаций, возникающих в процессе финишной обработки на конструкции детали и оснастки.

Объектом исследования является деталь «корпус» (рис. 1, *а*), она представляет собой тело вращения сложной формы, изготавливается из сплава 79 НМ. Масса составляет 100 г. Максимальная высота 29,8 мм, максимальный диаметр 54 мм. Центральное отверстие диаметром 13 служит базовым при обработке и измерениях. Деталь несимметричная, имеется 4 паза, расположенных под углом к вертикальной оси детали. Множество отверстий служат для крепления и сопряжения с входящими деталями изделия.

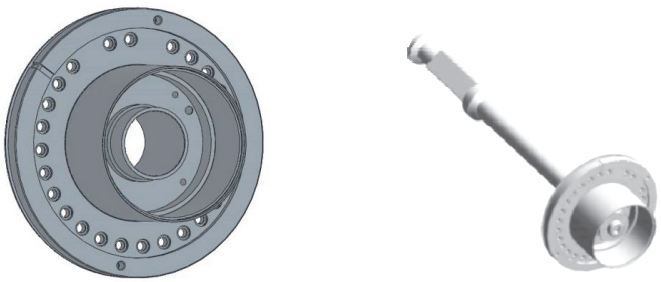
В процессе производства возникают дефекты в виде заусенцев, острых кромок, нестабильного класса шероховатости, которые в процессе доработки исправляют вручную. Особо проблемными местами зачистки заусенцев являются контуры пазов и отверстий.

В качестве наиболее технологичного метода финишной обработки применимо к рассматриваемой детали была выбрана технология MultiFinish. Она применяется для обработки деталей сложной геометрии и с высокими требованиями, в тех случаях, когда контакт обрабатываемых деталей между собой должен быть исключен. При реализации технологии используется принцип планетарного движения.

Для решения поставленной в работе задачи: проведение численного эксперимента по расчету максимальных напряжений и деформаций, возникающих в процессе финишной обработки, использовался современный инженерный программный комплекс ANSYS. Твердотельные модели детали и оснастки были построены с помощью программы SolidWorks.

Разработка и построение твердотельной модели детали и оснастки

С помощью программы SolidWorks были построены твердотельные модели детали и оснастки (рис. 1), соответствующие конструкторской документации.



а *б*

Рис. 1. Модель детали «Корпус» (*а*) и модель оснастки (*б*), построенные в программном комплексе SolidWorks

Разработка физической модели

Для решения поставленной задачи по расчету максимальных напряжений и деформаций, возникающих в корпусной детали при финишной обработке, была сформулирована следующая физическая модель.

- Применялась трехмерная конструкция детали и оснастки.
- Расчет проводился в статической нелинейной постановке с учетом теории малой деформации.

В физической модели принимается следующее:

- При снятии нагрузок деталь не полностью восстанавливает свою исходную форму (учитывается необратимая деформация).
- Максимальное расчетное перемещение значительно меньше корпусной детали.

При формировании физической модели были приняты следующие допущения:

- Не учитывается действие силы трения.
- Влияние напряжения конструкции на ее жесткость не учитывается.
- В зоне контакта оправки и корпусной детали рассматривается полное прилипание (идеальный контакт).
- Стенки конструкции не нагреваются, не поглощают тепло.
- Вращение задавалось с постоянной угловой скоростью относительно оси оправки и корпусной детали *Oz*.
- Упругость наполнителя задавалась через упругую опору, действующую на поверхность корпусной детали.

Разработка математической модели

Разработана математическая модель, которая содержит уравнения равновесия, геометрические соотношения Коши, определяющий закон деформируемой сплошной среды, краевые условия.

Математическое описание упругопластических процессов для каждого из двух контактирующих тел в указанной постановке включает в себя следующие соотношения:

– геометрические соотношения Коши:

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right),$$

где ε_{ij} – компоненты тензора деформаций; u_{ij} – перемещение точки сплошной среды;

– уравнение равновесия:

$$\sigma_{ij,j} + X_i = 0,$$

где X_i – составляющие объемной силы; $\sigma_{ij,j}$ – компоненты тензора напряжений;

– определяющий закон деформируемой сплошной среды:

$$\sigma_{ij} = K\theta\delta_{ij} + 2G\left(1-\omega(e_n)\right)\left(\varepsilon_{ij} - \frac{\theta}{3}\delta_{ij}\right),$$

где K – модуль объемного сжатия; G – модуль упругости; $\omega(e_n)$ – функция, характеризующая отклонение свойств материала от линейных; e_n – интенсивность деформаций (второй инвариант).

Инварианты тензора деформаций имеют вид:

$$\theta = (\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}),$$

$$e_n^2 = \left(\varepsilon_{ij} - \frac{\theta}{3}\delta_{ij}\right)\left(\varepsilon_{ij} - \frac{\theta}{3}\delta_{ij}\right).$$

Данная математическая модель замыкается следующими граничными условиями, схема которых показана на рис. 2:

$$u_i\big|_{S_1} = u_i^0, \quad \sigma_{ij}^{\mathrm{I}}\big|_{S_1} = -P_i^{\mathrm{II}}\Big|_{S_2}, \quad \sigma_{ij}^{\mathrm{I}}\big|_{S_3} = \sigma_{ij}^{\mathrm{II}}\Big|_{S_3}, \quad u_i^{\mathrm{I}}\big|_{S_3} = u_i^{\mathrm{II}}\Big|_{S_3},$$

Рис. 2. Схема граничных условий

где S_1, S_2 – части полной поверхности S, где заданы перемещения u_i^0 и напряжения σ_i^0; n_i – составляющие единичного вектора n, перпендикулярного поверхности S_2; S_3 – контактирующая поверхность; I, II – контактные тела I и II соответственно [1].

Разработка и описание сеточной модели

Было проведено численное моделирование напряженно-деформированного состояния (НДС) оснастки для финишной обработки при задании силы тяжести, скорости вращения, упругости наполнителя. Характеристики задавались через модуль упругости наполнителя.

При расчетах использовался метод конечных элементов (КЭ). Проводилось исследование по подбору формы и размера КЭ, расчет с уточнением элемента в местах закрепления и соединений.

Конечно-элементная модель была построена с помощью автоматического построителя ANSYS Workbench. Для поставленной задачи со сложной геометрией необходимо применение треугольных элементов. Для расчета использовалась структурированная тетраэдальная конечно-элементная сетка (рис. 3). Максимальный размер элемента 0,5 мм, минимальный – 0,001 мм.

Рис. 3. Модель сетки сборки

Мелкая сетка требуется там, где ожидается большой градиент деформаций [2] или напряжений (отверстие, выточка и т.п.). Крупная сетка применялась в зонах с малоизменяющимися относительными деформациями или напряжениями, а также в областях, не представляющих особого интереса для расчета. В связи с этим перед созданием конечно-элементной сетки были выделены предполагаемые области концентрации напряжений. На детали «корпус» концентраторами напряжения являются отверстия, пазы. На оснастке – места закрепления (лыски) и резьбовое соединение.

Расчет оснастки с зафиксированной деталью на НДС проводился при разных типах задания граничных условий. Был выбран тип анализа Static Structural. Проводились следующие виды расчетов.

1. Расчет под действием силы тяжести и с учетом фиксации оснастки.

2. Расчет под действием силы тяжести и вращения оснастки. Расчет проводился с учетом количества оборотов в минуту (10, 60, 100).

3. Расчет при заданной средней скорости вращения под действием силы тяжести, с учетом упругости наполнителя среды для обработки. Средняя скорость вращения зафиксированной детали при финишной обработке – 60 об/мин.

При расчетах учитывались характеристики материалов: для детали «корпус» – сплав 79НМ, для оснастки – углеродистая сталь. Материал наполнителя – керамика, плотность 2,6 г/см^3. Подробные характеристики сплава 79НМ, учитываемые при расчетах: плотность – 8600 кг/м^3, модуль упругости – $2{,}1 \cdot 10^5$ МПа, коэффициент Пуассона – 0,3.

Результаты расчетов указаны ниже:

1. Максимальные напряжения и перемещения под действием силы тяжести: 0,032 694 МПа и $1{,}4903 \cdot 10^{-5}$ мм соответственно.

2. Максимальные напряжения и перемещения при скорости вращения (таблица).

Максимальные напряжения и перемещения при скорости вращения

Переменная	Скорость вращения		
	10 об/мин	60 об/мин	100 об/мин
Напряжения, МПа	0,032 712	0,032 893	0,033 464
Перемещения, мм	$1{,}822\ 4 \cdot 10^{-5}$	$1{,}765 \cdot 10^{-5}$	$1{,}625\ 3 \cdot 10^{-5}$

3. Максимальные напряжения и перемещения с учетом упругости наполнителя при скорости вращения 60 об/мин: 0,031 75 МПа и $1{,}5746 \cdot 10^{-5}$ мм соответственно.

По результатам вычислительных экспериментов получены поля распределения эквивалентных напряжений (рис. 4) и общих перемещений (рис. 5).

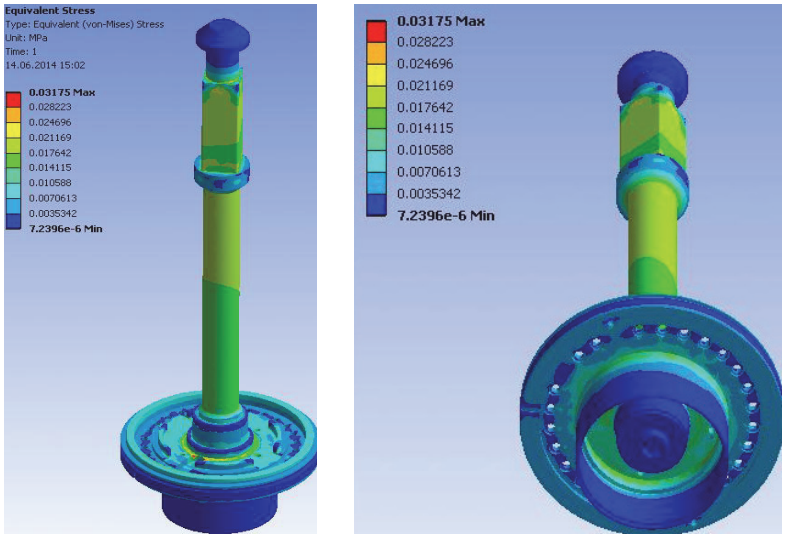


Рис. 4. Визуализация эквивалентных напряжений

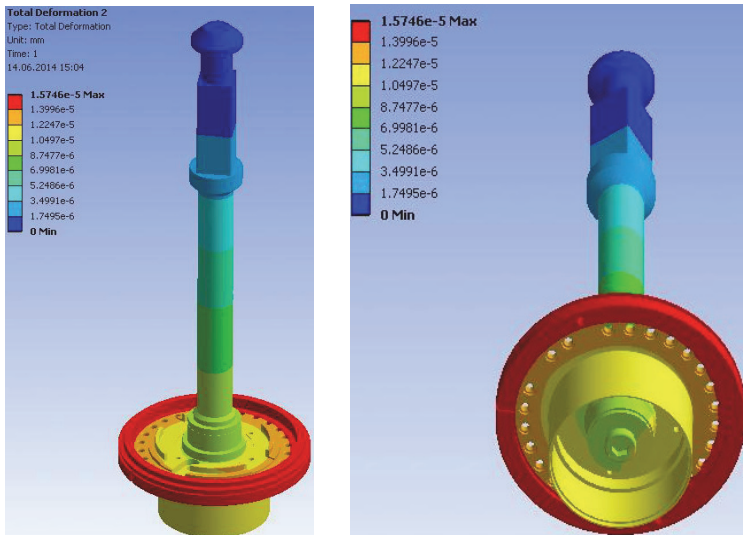


Рис. 5. Показатели общих перемещений

Анализ результатов. Коэффициент запаса прочности более 4, следовательно, оснастка может использоваться при более высоких нагрузках, может служить универсальной. Увеличение скорости вращение приводит к увеличению напряжений и уменьшению перемещений (деформаций). Деформации не выходят за пределы допусков на размеры. По результатам численных расчетов можно сказать, что конструкция оснастки соответствует предъявляемым требованиям.

Результаты работы могут использоваться на предприятии для улучшения деятельности производства при внедрении новых установок для финишной обработки.

Список литературы

1. Писарев П.В., Писарева А.А., Зимин Д.В., Бутымова Л.Н., Гайнутдинова Д.Ф., Модорский В.Я. Моделирование динамического напряженно-деформированного состояния конечномерного цилиндра, нагруженного гидродинамическим потоком жидкости на многопроцессорном вычислительном комплексе. Научный сервис в сети Интернет: экзафлопсное будущее // Труды Международной суперкомпьютерной конференции (19–24 сентября 2011 г., г. Новороссийск). – М.: Изд-во МГУ, 2011. – 643 с.

2. Давыдов Ю.М., Акжолов М.Ж., Модорский В.Я. и др. Численное исследование актуальных проблем машиностроения и механики сплошных и сыпучих сред методом крупных частиц / под ред. Ю.М. Давыдова. Т1-Т5. – М.: НАПН, 1995. – 1658 с.

Д.Ф. Гайнутдинова[1], В.Я. Модорский[1], Г.Ф. Масич[2]

[1]Пермский национальный исследовательский
политехнический университет,
Центр высокопроизводительных вычислительных систем,
[2]Институт механики сплошных сред УрО РАН, Пермь

РАЗРАБОТКА ЭЛЕМЕНТОВ ЭКСПЕРИМЕНТАЛЬНО-ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА НА БАЗЕ СУПЕРЭВМ ДЛЯ АНАЛИЗА ГИДРОУПРУГИХ БЫСТРОПРОТЕКАЮЩИХ ПРОЦЕССОВ

Рассмотрен способ исследования процессов гидроупругого динамического взаимовлияния, сочетающий численное моделирование с натурным экспериментом. Анализ быстропротекающих процессов натурного эксперимента проводится в реальном масштабе времени, с целью управления ими и поиска оптимальных параметров, исключающих процессы кавитации. Оптимизация требует создания технологической платформы на базе высокопроизводительного вычислительного комплекса ПНИПУ, сопряженного с экспериментальной установкой.

Постановка задачи

В ряде случаев процессы гидроупругого динамического взаимовлияния могут явиться причиной возникновения непрогнозируемых опасных эффектов при работе наукоемкого оборудования. В частности, при работе центробежных насосов может наблюдаться непрогнозируемое снижение напора. Это явление объясняется возникновением кавитационных эффектов в каналах переменного сечения при воздействии вибраций стенок конструкции [1–5].

Численное моделирование

Для исследования этого явления решается модельная задача, представляющая собой движение поршня (аналог деформируемой стенки) в замкнутой трубе, заполненной жидкостью. Эмулируется поршень, который совершает движение по заданному синусоидальному закону $V = V_0 \sin \omega t$. В качестве расчетной среды используется система инженерного анализа FlowVision 2.5, которая позволяет решать задачи гидрогазодинамики и кавитации в трехмерной динамической постановке.

Для решения задачи поиска границ возникновения кавитации разработана математическая модель экспериментальной установки. Используется программа многопараметрической оптимизации по результатам измерений, поступающим от установки. Это позволит при соответствующем выборе модели оптимизации существенно снизить время счета на суперкомпьютере и управлять экспериментальной установкой в реальном времени.

Критерием оптимизации в нашем случае будет концентрация воздушных пузырьков C в жидкости, возникающих при движении поршня. Контролируемым параметром будет давление жидкости. Варьируемыми параметрами управления экспериментальной установки будут амплитуда V_0 и частота ω колебаний. Диапазон варьирования амплитуды колебаний V_0 от 0,1 до 10 м/с, частоты колебаний ω от 100 до 10 000 рад/с. В результате получим область и условия возникновения кавитации, которые будут воспроизведены и верифицированы в натурном эксперименте.

Экспериментальная установка

Для исследования кавитационных явлений проектируется экспериментальная установка, позволяющая нагружать гидродинамический объем в замкнутой трубе по заданному закону (рис. 1).

Экспериментальная установка представляет собой конструкцию, состоящую из двух узлов – рабочей камеры и узла нагружения. Рабочая камера состоит из основания (плиты), набора грузов, трубы рабочей длиной

440 мм, общей длиной 480 мм, которая сваривается аргонодуговой сваркой с нижним фланцем. Труба жестко закреплена на массивном основании. В камеру заливается рабочее тело – вода. В качестве узла нагружения планируется использовать пьезоэлектрический акустический трансформатор. Кроме того, предусмотрена возможность крепления регулирующих элементов на верхний фланец экспериментальной установки. Это необходимо для настройки конструкции на резонансные частоты.

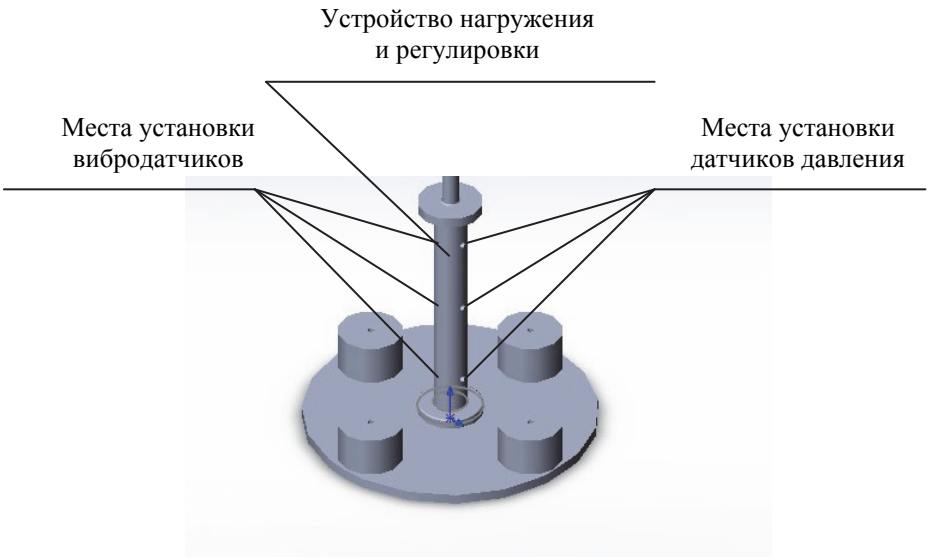


Рис. 1. Экспериментальная установка

Планируется установить датчики вибраций PCB 352C03. Это пьезоэлектрические датчики со встроенным микроэлектронным предусилителем заряда. Для измерения давления планируется использовать датчики давления ДАВ 084-1Б6. Сигналы с датчиков через специальную согласующую плату передаются на оборудование сбора данных [6].

Измерительная система

Регистрация и предварительный анализ быстропротекающих процессов в рабочей камере выполняются программно-аппаратным комплексом (ПАК) на базе National Instruments NI PXI-1050. Измерительная система создана на базе шасси NIPXI\SCXI 1050 с установленным модулем сбора динамических сигналов NIPXI 4472 B. Предусмотрена интеграция ПАК с высокопроизводительным вычислительном комплексом ПНИПУ.

Инфраструктура технологической платформы

Технологическая платформа для экспериментальных и вычислительных исследований быстропротекающих процессов гидроупругости, показанная на рис. 2, состоит из суперкомпьютера, системы хранения данных

(СХД) и источника. Суперкомпьютером является высокопроизводительный вычислительный комплекс ПНИПУ, имеющий счетный интерконнект InfiniBand и интерконнект ввода-вывода Ethernet. Источником является вышеупомянутая измерительная система NIPXI\SCXI 1050, соединенная с изображенной на рис. 1 экспериментальной установкой. Система хранения данных ориентирована на хранение больших объемов экспериментальных данных. Интерконнекты Ethernet вышеперечисленных компонентов технологической платформы соединены друг с другом на скорости 1–10 Гбит/с, они обеспечивают гибкость проводимых исследований согласно двум моделям обработки на суперкомпьютере измеряемых данных и управления экспериментом.

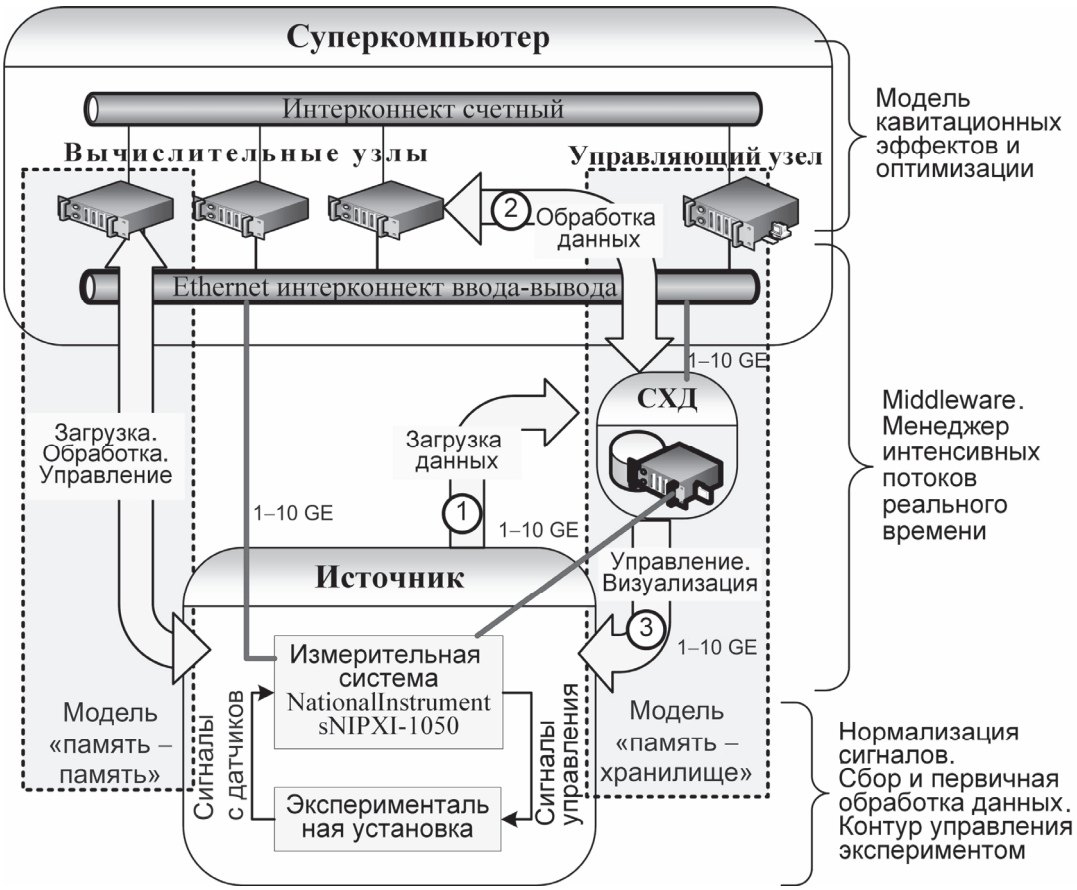


Рис. 2. Инфраструктура технологической платформы

Модель «память – хранилище» проходит три этапа: загрузка данных от источника в СХД (1), обработка размещенных в СХД данных на суперкомпьютере (2), выгрузка результатов обработки с СХД на источник (3). Загрузка (выгрузка) данных в хранилище (этапы 1 и 3) и последующая обработка (этап 2) выполняются либо с использованием протоколов передачи файлов (FTP/GridFTP и SCP), либо в прямом доступе к хранилищу данных при помощи протоколов работы с файловой системой (CIFS и NFS/pNFS). Эта мо-

дель является классической схемой обработки больших данных на суперкомпьютере, в которой процесс измерений и процесс счета разорваны во времени и не соответствуют в ряде случаев требованиям управления экспериментом.

Модель «память – память» ориентирована на обработку на суперкомпьютере интенсивного потока данных от источника в реальном масштабе времени [7]. Идея этой модели основана на прямом вводе интенсивного потока структурированных данных в память вычислительных узлов суперкомпьютера, минуя внешнюю систему хранения данных. Адаптация реализованного комплекса специализированного программного обеспечения middleware модели «память – память» [7, 8] для создаваемой технологической платформы – основной предмет организации взаимодействия источника с суперкомпьютером. Менеджер SciMQ является основной частью реализованного middleware, с которым взаимодействуют приложения источника и суперкомпьютера по протоколу SciMP [8] через API, предоставляемый клиентской библиотекой.

Выводы

Разрабатываемая технологическая платформа будет использована для анализа быстропротекающих процессов в наукоемких изделиях, где требуется управление в реальном масштабе времени и оптимизация по рабочим параметрам. Отличительной особенностью архитектурного решения является соединение математической модели исследуемого процесса на суперкомпьютере с лабораторной установкой, эмулирующей исследуемый процесс.

Работы проводятся при поддержке Российского фонда фундаментальных исследований, Грант РФФИ № 14-07-96003-р_урал_а.

Список литературы

1. Модорский В.Я., Соколкин Ю.В. Газоупругие процессы в энергетических установках. – М.: Наука, 2007.– 176 с.

2. Решение инженерных задач на высокопроизводительном вычислительном комплексе Пермского национального исследовательского политехнического университета: монография / под ред. В.Я. Модорского. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2014. – 314 с.

3. Козлова А.В., Модорский В.Я. Численное моделирование кавитационных эффектов в замкнутой трубе с подвижной стенкой // Науч.-техн. вестник Поволжья. – 2013. – № 2. – С. 132–135.

4. Численное моделирование кавитационных эффектов при действии вибрации / Д.Ф. Гайнутдинова, А.В. Козлова, В.Я. Модорский, Е.В. Мехоношина // Науч.-техн. вестник Поволжья. – 2013. – № 6. – С. 219–223.

5. Исследование влияния вибраций и давления на возникновение кавитации / Е.В. Мехоношина, В.Я. Модорский, А.В. Козлова, Д.Ф. Гайнутдинова // Науч.-техн. вестник Поволжья. – 2013. – № 6. – С. 364–368.

6. Бутымова Л.Н., Модорский В.Я., Соколкин Ю.В. Разработка экспериментальной установки и исследование влияния материала корпуса на резонансные частоты в системе газ–конструкция // Науч.-техн. вестник Поволжья. – 2013. – № 6. – С. 197–200.

7. Щапов В.А., Масич А.Г., Масич Г.Ф. Модель потоковой обработки экспериментальных данных в распределенных системах // Вычислительные методы и программирование. – 2012. – Разд. 2. – С. 139–145

8. Shchapov V., Masich A. Protocol of High Speed Data Transfer from Particle Image Velocimetry System to Supercomputer // Proc. of The 7th International Forum on Strategic Technology (IFOST 2012), September 18–21 / Tomsk Polytechnic University. – Tomsk, 2012. – Vol. 2. – P. 653–657.

Е.А. Глазкова

ООО «Рок Флоу Динамикс», Москва

ГИБРИДНЫЙ ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ЛИНИЙ ТОКА

Предложен параллельный гибридный алгоритм построения линий тока методом трассировки частиц. Описывается программная (MPI+Threads) реализация алгоритма для задачи визуализация результатов гидродинамического моделирования нефтегазовых месторождений. Рассматриваются вопросы балансировки загруженности между потоками.

Введение

При визуализации научных данных одним из распространенных подходов является визуализация векторных полей с помощью линий тока. Линия тока – это кривая, направление касательной в каждой точке которой совпадает с направлением в этой точке вектора скорости. Линии тока широко используются при моделировании нефтегазовых месторождений для решения задач оптимизации заводнения, воспроизведения истории разработки месторождения при построении модели месторождения, визуализации результатов расчета гидродинамических симуляторов [1].

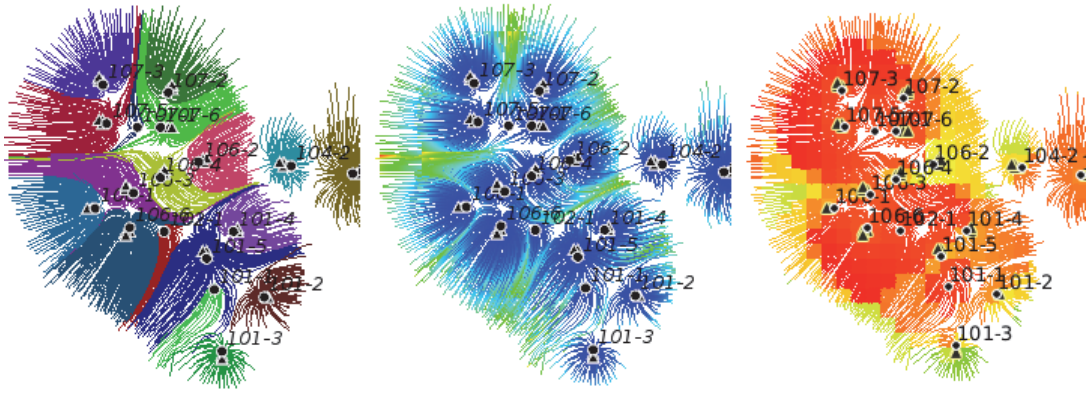
Современные параллельные гидродинамические симуляторы позволяют моделировать большие нефтегазовые месторождения с высокой степенью детализации [2]. Подробная гидродинамическая модель нефтегазового месторождения, которая учитывает все доступные сейсмические, гео-

логические и динамические данные о месторождении, может содержать от нескольких сотен миллионов до миллиарда блоков сетки. Чем больше блоков сетки используется в модели, тем точнее можно построить поле давлений и вычислить скорость фильтрации. Расчет больших гидродинамических моделей возможен только с помощью многопроцессорной высокопроизводительной вычислительной системы и требует использования эффективных параллельных алгоритмов.

После расчета гидродинамической модели необходимо визуализировать результаты расчета на локальной машине пользователя. Для больших моделей месторождений расчет линий тока на локальной машине потребует много времени и оперативной памяти, поэтому линии тока необходимо вычислить на кластере и сохранить для последующей визуализации. Для этого необходимо разработать параллельный алгоритм построения линий тока.

Узлами современных высокопроизводительных систем являются, как правило, многоядерные процессоры, поэтому для создания параллельных приложений для таких систем целесообразно использовать гибридный подход (MPI + Threads) [3]. В этой модели программирования MPI используется для организации взаимодействий между узлами (процессами), а потоки – для распараллеливания задачи внутри узла.

Важным аспектом при создании параллельных алгоритмов является балансировка загруженности [4]. Для эффективного распараллеливания построения линий тока это особенно важно, потому что линии тока распределены неравномерно между разными участками сетки и имеют различную длину (рис. 1), что создает существенный дисбаланс при распределении заданий между потоками и процессами [5].



а *б* *в*

Рис. 1. Использование линий тока для визуализации результатов расчета:
а – линии тока раскрашены в цвет скважины, от которой они трассируются;
б – время пролета частицы, *в* – давление в блоке

В работе предложен параллельный алгоритм, который использует очередь заданий для многопоточного распараллеливания внутри одного процесса и обмен граничными точками линий тока между MPI-процессами.

Построение линий тока методом трассировки частиц

Эффективный и наиболее часто используемый в настоящее время метод построения линий тока был предложен Д. Поллоком в статье [6].

В данном методе частицы трассируются поблочно и применяется кусочно-линейная аппроксимация скорости по значениям заданным на границах блоков сетки. Пусть $(x_{\text{enter}}, y_{\text{enter}}, z_{\text{enter}})$ – точка входа частицы в блок сетки. Скорость частицы внутри блока сетки и градиент скорости в направлении x задаются формулой

$$\begin{aligned} v_x &= v_{x0} + g_x(x - x_0), \\ g_x &= \frac{v_{x0+\Delta x} - v_{x0}}{\Delta x}, \end{aligned} \tag{1}$$

где v_{x0} и $v_{x0+\Delta x}$ – значения скорости, заданные на противоположных гранях блока сетки. Аналогичные соотношения можно записать для скорости частицы вдоль направлений y и z. Соотношение для скорости частицы (1) можно проинтегрировать по времени, чтобы получить время, через которое частица покинет блок сетки через одну из граней.

$$\Delta t_x = \frac{1}{g_x}\ln\left(\frac{v_{x0} + g_x\left(x_{\text{exit}} - x_0\right)}{v_{x0} + g_x\left(x_{\text{enter}} - x_0\right)}\right), \Delta t_y = \frac{1}{g_y}\ln\left(\frac{v_{y0} + g_y\left(y_{\text{exit}} - y_0\right)}{v_{y0} + g_y\left(y_{\text{enter}} - y_0\right)}\right),$$

$$\Delta t_z = \frac{1}{g_z}\ln\left(\frac{v_{z0} + g_z\left(z_{\text{exit}} - z_0\right)}{v_{z0} + g_z\left(z_{\text{enter}} - z_0\right)}\right), \Delta t_m = \min\left(\Delta t_x, \Delta t_y, \Delta t_z\right).$$

Частица покинет блок сетки через грань, для которой время полета минимально, и точка выхода частицы из блока может быть вычислена по формуле

$$x_{\text{exit}} = \frac{1}{g_x}\left(v_{x_{\text{enter}}} e^{x\Delta tm} - v_{x0}\right) + x_{0,}\, y_{\text{exit}} = \frac{1}{g_y}\left(v_{y_{\text{enter}}} e^{y\Delta tm} - v_{y0}\right) + y_0,$$

$$z_{\text{exit}} = \frac{1}{g_z}\left(v_{z_{\text{enter}}} e^{z\Delta tm} - v_{z0}\right) + z_0.$$

На следующем шаге алгоритма точка $\left(x_{\text{exit}}, y_{\text{exit}}, z_{\text{exit}}\right)$ рассматривается как точка входа частицы в следующий блок. Сетки, использующиеся для расчетов в гидродинамических симуляторах, имеют непрямоугольные блоки,

следовательно, для них вначале выполняется изопараметрическое преобразование координат. Алгоритм останавливается, когда время пролета частицы становится больше заданного пользователем максимального значения.

Параллельный алгоритм

Параллельная реализация алгоритма включает два уровня параллелизма: очередь заданий для многопоточной обработки внутри одного MPI-процесса и обмен граничными точками линий тока между MPI-процессами задачи (рис. 2).

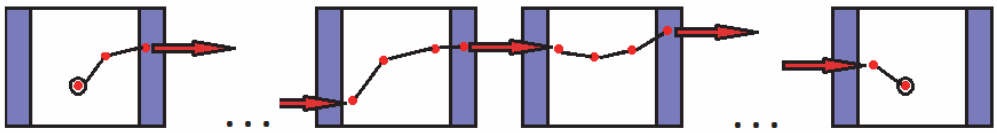


Рис. 2. Схема трассировки линии тока, проходящей через блоки сетки, принадлежащие нескольким MPI-процессам

Описание алгоритма:

1. Сначала каждый MPI-процесс определяет начальные точки линий тока, выходящих из принадлежащих ему блоков сетки. В качестве блоков сетки, из которых начинают трассироваться линии тока, обычно выбираются блоки, в которых находятся интервалы перфорации скважин, активных на данном расчетном шаге симулятора. Количество линий тока прямо пропорционально дебиту или приемистости скважины.

2. После определения начальных точек линий тока происходит формирование заданий для очереди заданий внутри MPI-процесса. Количество линий тока, содержащихся в одном задании, не может быть слишком большим, потому что это приведет к дисбалансу загруженности между потоками, но маленький размер задания приведет к большим накладным расходом на синхронизацию процессов. Заранее фиксированное количество линий тока в качестве размера задания тоже нельзя выбрать, потому что в разных моделях количество линий тока может очень сильно отличаться, поэтому количество заданий в очереди и количество линий тока внутри одного задания определяются по формуле

$$\text{Task_size} = \frac{N_{\text{streamlines}}}{N_{\text{threads}}}, N_{\text{tasks}} = \frac{N_{\text{streamlines}}}{\text{Task_size}}, \quad (2)$$

где $N_{\text{streamlines}}$ – количество линий тока; N_{threads} – количество потоков; N_{tasks} – количество заданий в очереди; Task_size – количество линий тока внутри одного задания.

Таким образом, чем больше число потоков внутри MPI-процесса, тем больше количество заданий в очереди и меньше размер одного задания.

3. На каждом MPI-процессе хранятся «теневые грани» – блоки сетки, принадлежащие соседним MPI-процессам. После инициализации очереди заданий, каждый поток в цикле берет из очереди новую порцию линий тока и трассирует каждую линию тока, пока время пролета частицы не превысит заданную пользователем величину либо частица не достигнет блока из «теневой грани». В этом случае линия тока помечается для последующей передачи MPI-процессу, которому принадлежит этот блок сетки.

4. Когда все задания внутри узла выполнены, MPI-процессы собирают с других процессов данные о неоттрассированных частицах, попавших в их локальную сетку. После этого происходит формирование заданий для обработки внутри каждого MPI-процесса и процесс трассировки возобновляется (см. п. 3).

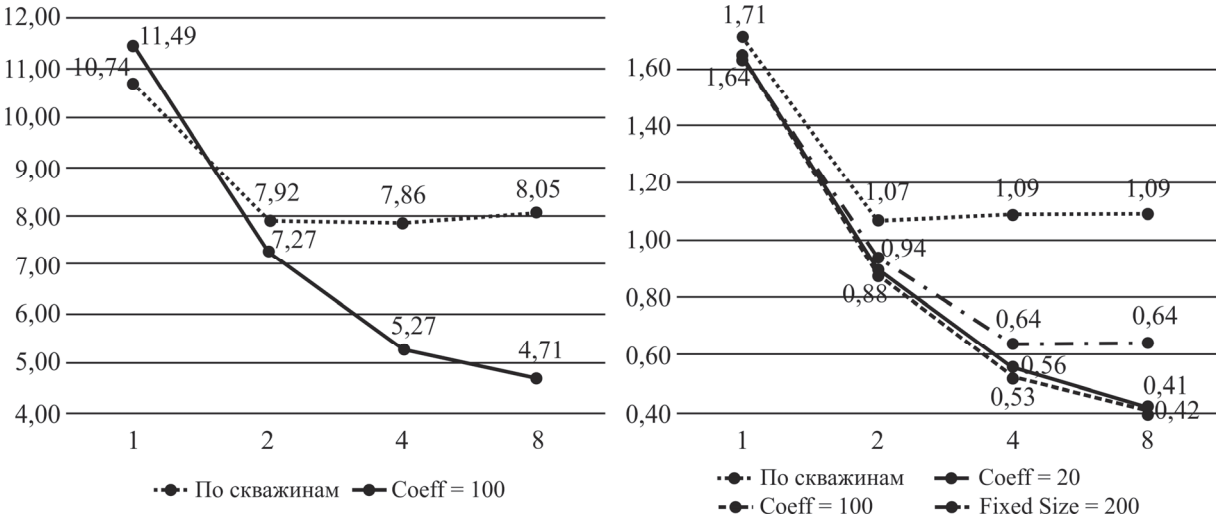
5. Алгоритм останавливается, когда ни у одного из MPI-процессов не осталось неоттрассированных частиц с временем пролета меньше заданной пользователем величины.

Вычислительный эксперимент

В первом эксперименте исследовалось влияние способа формирования заданий очереди и числа потоков на время выполнения многопоточной версии. Эксперимент проводился на двух моделях месторождений: в первой модели несколько сотен тысяч линий тока примерно одинаковой длины, во второй модели примерно 1,5 тыс. линий тока, которые имеют сильно отличающуюся длину (длина колеблется от 20 до 400 точек). Первый вариант формирования заданий, когда в одно задание попадают все линии тока, принадлежащие одной скважине, оказался неэффективным на обоих моделях, потому что количество линий тока, идущих от одной скважины, прямо пропорционально дебиту скважины, а он сильно отличается у разных скважин. Фиксированный размер задания (200 линий тока) на второй модели, начиная с 8 потоков, оказался хуже, чем распределение заданий по формуле (2). Для формулы (2) были протестированы значения коэффициента 20 и 100. Лучший результат получился при значении коэффициента, равном 100, что соответствует более мелкому размеру задания.

Во втором эксперименте исследовалось время выполнения гибридной MPI+Threads версии на кластере, вычислительными узлами которого являются два 10-ядерных процессора Intel Xeon CPU E5-2680 с тактовой частотой 2,80 ГГц, соединенных сетью Infiniband 56 Гб/с, объем оперативной памяти на одном узле – 128 Гб. Эксперимент проводился на гидродинамической модели, содержащей 43 млн блоков сетки. В эксперименте

считались первые 10 шагов модели. На момент последнего шага было 20 тыс. линий тока со средней длиной 140 точек. Из-за того, что в начале расчета модели скважины распределены по сетке неравномерно, разница между первым и вторым процессами незначительна, но начиная с четырех процессов время расчета линий тока начинает ускоряться (рис. 3–5).



а *б*

Рис. 3. Зависимость времени вычисления линий тока многопоточной версией от числа потоков и способа формирования заданий: *а* – модель 1; *б* – модель 2

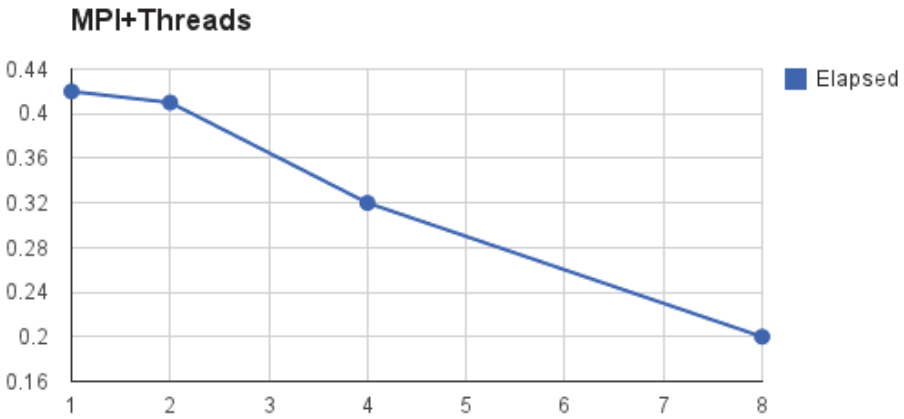


Рис. 4. Время выполнения гибридной версии на кластере

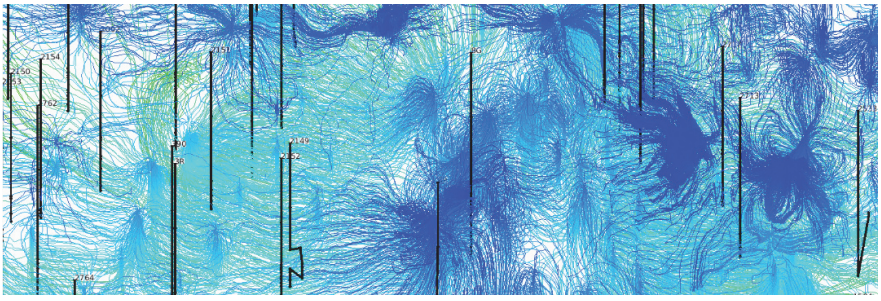


Рис. 5. Линии тока модели из третьего эксперимента

В дальнейшем планируется заменить в реализации коллективные взаимодействия на асинхронные обмены сообщениями.

Заключение

Предложен параллельный гибридный алгоритм построения линий тока. Была выполнена программная реализация алгоритма на языке программирования C++ с использованием библиотеки MPI. Описанный метод был реализован в составе российского гидродинамического симулятора tNavigator.

Список литературы

1. Thiele M.R., Gerritsen M.G., Blunt M.J. Streamline simulation / Society of Petroleum Engineers. – 2011.

2. Advances in modeling of giant reservoirs / V.I. Dzyuba [et al.] // Mathematical Methods in Fluid Dynamics and Simulation of Giant Oil and Gas Reservoirs. – 2012.

3. Трехуровневый MPI+NUMA+Threads метод построения параллельных программ для расчета задач гидродинамического моделирования на кластерных системах с узлами с неоднородным доступом к памяти / К.Ю. Богачев [и др.] // Вычислительные методы и программирование. – 2013. – Т. 14. – С. 375–382.

4. Балансировка загруженности узлов кластера при расчете задачи фильтрации / К.Ю. Богачев [и др.] // Вычислительные методы и программирование. – 2011. – Т. 12. – С. 70–73.

5. A study of parallel particle tracing for steady-state and time-varying flow fields / T. Peterka [et al.] // Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. – 2011. – P. 580–591.

6. Pollock D.W. Semianalytical computation of path lines for finite difference models // Groundwater. – 1988. – Vol. 26. – № 6. – C. 743–750.

Е.Н. Головченко, Е.Ю. Дорофеева

Институт прикладной математики им. М.В. Келдыша РАН, Москва

РАЗБИЕНИЕ ГРАФОВ МИКРОДОМЕНОВ

Задача рациональной декомпозиции расчетных сеток возникает при численном моделировании на высокопроизводительных вычислительных системах проблем механики сплошных сред, импульсной энергетики, электродинамики и др. Число процессоров, на котором будет считаться вычислительная задача, как правило, заранее неизвестно, поэтому имеет

смысл предварительно однократно разбить сетку на большое число микродоменов, а потом формировать из них домены. Проведены вычислительные эксперименты по сравнению различных разбиений графов микродоменов на домены, а также разбиений сразу на домены. На задаче моделирования распространения ударной волны от приземного взрыва проведено тестирование различных разбиений графов микродоменов и разбиения сразу на домены, полученных параллельным инкрементным алгоритмом созданного комплекса программ GRIDSPIDERPAR.

Введение

Задача рациональной декомпозиции расчетных сеток возникает при численном моделировании на высокопроизводительных вычислительных системах проблем механики сплошных сред, импульсной энергетики, электродинамики и многих других. При распараллеливании подобных вычислительных приложений используется метод геометрического параллелизма, при котором сетка, аппроксимирующая расчетную область, распределяется между процессорами по геометрическому признаку. В ходе расчета каждый процессор обрабатывает свою часть сетки. Эффективность работы многопроцессорной вычислительной системы определяется тем, насколько равномерно распределена сетка по процессорам и насколько минимизированы затраты на передачу данных между процессорами. Объем передаваемых между процессорами данных зависит от числа связей между распределенными по процессорам доменами (частями сеток).

Число процессоров, на котором будет считаться вычислительная задача, как правило, заранее неизвестно, поэтому имеет смысл предварительно однократно разбить сетку на большое число микродоменов, а потом формировать из них домены. Количество микродоменов на несколько порядков меньше числа вершин, поэтому многократное разбиение микродоменов на домены быстрее многократного разбиения всей сетки.

Задача сбалансированного разбиения сетки на домены сводится к более общей задаче разбиения графа на домены. Методы разбиения графов параллельных пакетов PARMETIS, JOSTLE, PT-SCOTCH и ZOLTAN основываются на иерархических алгоритмах, состоящих из следующих частей: поэтапное огрубление графа, декомпозиция самого маленького из полученных графов и отображение разбиения на предыдущие графы с периодическим локальным уточнением границ доменов. Недостатком таких алгоритмов является образование доменов, границы которых состоят из неоптимальных наборов сегментов, в частности несвязных доменов. Такое ухудшение качества доменов для некоторых задач является критичным. На доменах с длинными границами или сложной конфигурацией алгоритмы решения систем линейных уравнений сходятся за большее число ите-

раций. В алгоритме композиции подобластей [1] у несвязных подобластей длиннее приграничные полосы, в которых требуется повторное вычисление значений, а на узких приграничных полосах возникают проблемы с применимостью метода. Несвязные домены с оторванными ячейками являются неприемлемыми, например, для распараллеливания методики ТИМ-2D, решения задач механики сплошной среды [2] на нерегулярных многоугольных сетках произвольной структуры.

Другим недостатком указанных пакетов является получение сильно несбалансированных разбиений. В частности, в разбиениях, получаемых пакетом PARMETIS, числа вершин в доменах могут отличаться в два раза. К тому же разбиения больших сеток на большое число микродоменов не всегда удается получить методами существующих пакетов разбиения графов.

Вышесказанное обусловило разработку комплекса программ декомпозиции больших сеток – GRIDSPIDERPAR, в который вошли два алгоритма: параллельный алгоритм геометрической декомпозиции сеточных данных и параллельный инкрементный алгоритм декомпозиции графов [3, 4]. Параллельный алгоритм геометрической декомпозиции сеточных данных (GeomDecomp) основан на методе рекурсивной координатной бисекции. Параллельный инкрементный алгоритм декомпозиции графов (IncrDecomp) основан на последовательном инкрементном алгоритме декомпозиции графов [5]. Достоинством инкрементного алгоритма является формирование преимущественно связных доменов.

Результаты разбиения графов микродоменов на домены различными методами

Вычисления проводились на кластерах МВС-100К (227,94 TFlop/s) и «Ломоносов» (1700 TFlops).

Проведены вычислительные эксперименты по сравнению различных разбиений графов микродоменов на домены, а также разбиений сразу на домены (рис. 1). По разбиениям четырех тетраэдральных сеток (10^8–$2{,}7\cdot10^8$ вершин, $7\cdot10^8$–$1{,}6\cdot10^9$ тетраэдров, $8\cdot10^8$–$1{,}9\cdot10^9$ ребер) на 25 600 микродоменов методами пакета PARMETIS и созданного комплекса программ GRIDSPIDERPAR были составлены графы связей между микродоменами с весами вершин, соответствующими количеству вершин в микродоменах. Графы связей были разбиты на 512 доменов на одном процессоре методами PartGraphRecursive (PGR) и PartGraphKway (PGrK) пакета METIS, методом PartKway (PK) пакета PARMETIS и методом IncrDecomp (I) созданного комплекса программ GRIDSPIDERPAR, запущенными на одном процессоре. Проведено сравнение различных вариантов разбиений микродо-

менов на домены между собой и с разбиениями сразу на домены методами PartKway (PK), PartGeomKway (PGK) и PartGeom (PG) пакета PARMETIS, GeomDecomp (G) созданного комплекса программ GRIDSPIDERPAR, диффузионным алгоритмом пакета PT-SCOTCH и методами RCB, RIB и HSFC пакета ZOLTAN.

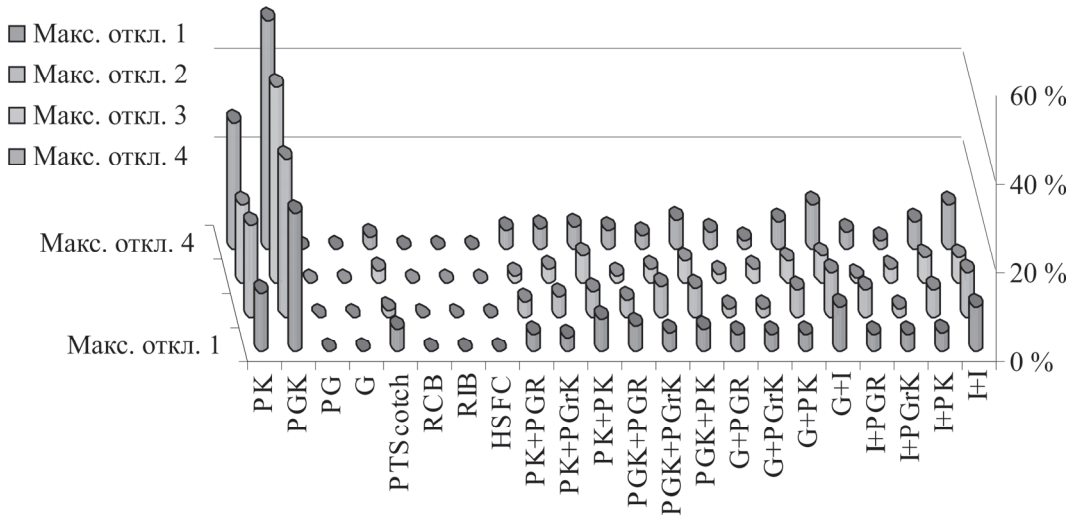


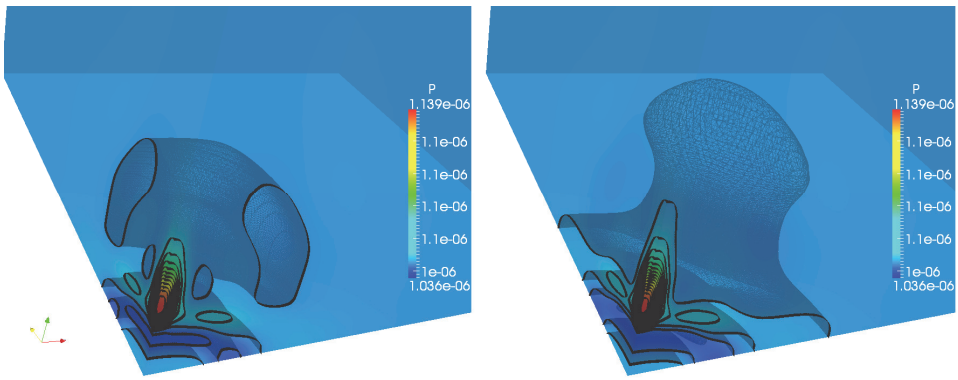
Рис. 1. Процентное отношение максимального модуля отклонения (макс. откл.) от среднего арифметического числа вершин в домене в разбиениях тетраэдральных сеток на 512 доменов (слева приведены разбиения сразу на домены, справа – разбиения графов микродоменов, методы разбиения на микродомены и разбиения графов на домены объединены знаком «+»)

Результаты показали, что дисбаланс числа вершин в доменах, сформированных из микродоменов, не зависит от дисбаланса числа вершин в микродоменах. Видимо, это связано с недостаточной чувствительностью алгоритмов разбиения графов к весам вершин. Лучшие разбиения графов микродоменов на домены получены пакетом METIS. Наиболее качественные разбиения на домены получены методами GeomDecomp (параллельный алгоритм геометрической декомпозиции) созданного комплекса программ GRIDSPIDERPAR, RCB пакета ZOLTAN и пакетом PT-SCOTCH. Оценивались дисбаланс числа вершин в доменах, число разрезанных ребер и число несвязных доменов.

Результаты тестирования разбиений графов микродоменов на физической задаче

На задаче моделирования распространения ударной волны от приземного источника энергии взрывного типа (рис. 2) проведено тестирование различных разбиений графов микродоменов и разбиения сразу на домены, полученных параллельным инкрементным алгоритмом (IncrDecomp)

комплекса программ GRIDSPIDERPAR. Сравнивалась эффективность параллельного счета рассматриваемой физической задачи пакетом MARPLE3D при распределении сетки по ядрам в соответствии с различными разбиениями. Параллельный программный комплекс MARPLE3D создан в ИПМ им. М.В. Келдыша РАН, и его предметной областью являются задачи двухтемпературной радиационной магнитной гидродинамики.



а *б*

Рис. 2. Апроксимация изоповерхностей давления
на расчетную сетку P (10^{11} Па) в момент времени t = 1000 мс
с шагом dP = 691 Па (*а*) и dP = 518 Па (*б*)

Дуальный граф boomL, содержащий $1{,}2 \cdot 10^8$ вершин и $1{,}0 \cdot 10^9$ ребер, был разбит на различное число микродоменов (от 24 576 до 196 608) и сразу на 3072 домена алгоритмом IncrDecomp. Составлены графы связей микродоменов с весами вершин, соответствующими количеству вершин в микродоменах. Графы микродоменов были разбиты алгоритмом IncrDecomp на 3072 домена.

В соответствии с разбиениями дуального графа boomL были получены разбиения вершин сетки. Для расчета физической задачи на всех разбиениях выделялось одинаковое машинное время (5 ч). Были получены числа шагов по времени, до которых досчитала задача.

В таблице представлены результаты тестирования разбиений. Под дисбалансом подразумевается процентное отношение максимального модуля отклонения от среднего арифметического числа вершин в домене.

Как видно из таблицы, чем больше микродоменов, тем меньше дисбаланс получаемых разбиений, тем меньше максимальное число соседних доменов, но тем больше общее число разрезанных ребер. Увеличение общего числа разрезанных ребер объясняется тем, что при составлении графов микродоменов не учитывались веса ребер между доменами. Максимальное число соседних доменов влияет на количество обменов между процессорами, обрабатывающими данные домены. С увеличением числа

микродоменов увеличивается число шагов по времени, полученных на разбиениях. Это говорит о том, что для задачи моделирования распространения ударной волны от приземного взрыва равномерность распределения вычислительной нагрузки по процессорам и количество обменов между процессорами критичнее, чем объем передаваемых данных.

Результаты тестирования разбиений графа boomL (116 214 272 гексаэдров) на 3072 домена, полученных алгоритмом IncrDecomp

Микро-домены	Микро-домены в домене	Дисбаланс, %	Разрезанные ребра	Соседние домены (max)	Несвяз-ные до-мены	Шаги по времени
3072	1	9,1	53 140 207	28	0	1107
24576	8	62,5	64 611 859	25	0	833
49152	16	37,5	66 566 874	25	0	880
98304	32	18,7	68 841 339	23	0	949
196608	64	7,9	68 207 798	21	0	999

При сравнении разбиения сразу на 3072 домена и разбиений графов микродоменов заметно, что в разбиении, составленном из 196 608 микродоменов, дисбаланс числа вершин в доменах меньше, чем в разбиении сразу на домены, и меньше максимальное число соседних доменов. Это объясняется тем, что при разбиении на определенное количество доменов не всегда удается получить требуемый дисбаланс. Например, при разбиении данного графа на 4096 доменов получаемый дисбаланс составлял 0,03 %, что значительно меньше 9,1 %, полученных при разбиении на 3072 домена. Число шагов по времени, полученных на разбиении, составленном из 196 608 микродоменов, не намного меньше, чем полученных на разбиении сразу на домены.

Таким образом, можно сделать вывод, что при достаточном количестве микродоменов в доменах разбиения графов микродоменов не уступают по качеству разбиению сразу на домены, что подтверждается малым уменьшением скорости счета рассматриваемой физической задачи. К тому же на декомпозицию графа микродоменов при массовых расчетах требуется меньше процессоро-часов.

Работа выполнена при поддержке РФФИ (гранты 13-01-12073 офи_м, 14-01-00663 А, 14-07-00712 А, 14-01-31154 мол_а).

Список литературы

1. Илюшин А.И., Колмаков А.А., Меньшов И.С. Построение параллельной вычислительной модели путем композиции вычислительных объектов // Математическое моделирование. – 2011. – Т. 23, № 7. – С. 97–113.

2. Воропинов А.А. Декомпозиция данных для распараллеливания методики ТИМ-2D и критерии оценки ее качества // Вестник ЮУрГУ. Математическое моделирование и программирование. – 2009. – № 37 (170), вып. 4. – С. 40–50.

3. Головченко Е.Н. Параллельный пакет декомпозиции больших сеток // Математическое моделирование. – 2011. – Т. 23, № 10. – С. 3–18.

4. Numerical Experiments with New Algorithms for Parallel Decomposition of Large Computational Meshes / E. Golovchenko, E. Dorofeeva, I. Gasilova, A. Boldarev // Parallel Computing: Accelerating Computational Science and Engineering (CSE). Advances in Parallel Computing. – 2014. – Vol. 25. – P. 441–450.

5. Якобовский М.В. Инкрементный алгоритм декомпозиции графов // Вестник Нижегородского университета им. Н.И. Лобачевского. Математическое моделирование и оптимальное управление. – 2005. – Вып. 1 (28). – С. 243–250.

С.Ю. Городецкий

Нижегородский государственный университет им. Н.И. Лобачевского

О МАТЕМАТИЧЕСКИХ ПРИНЦИПАХ УПРАВЛЯЕМОГО СИНТЕЗА КЛАССОВ МНОГОЭКСТРЕМАЛЬНЫХ ЗАДАЧ С ОГРАНИЧЕНИЯМИ ДЛЯ ПРЕДСТАВИТЕЛЬНОГО ТЕСТИРОВАНИЯ МЕТОДОВ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ

Представлены математические принципы псевдослучайного синтеза классов тестовых многоэкстремальных задач со сложными ограничениями при известном положении условного глобального минимума в каждой задаче. Свойства, сложность и вид функций тестовых классов определяются небольшим числом параметров при высоком разнообразии порождаемых задач. Обеспечивается повторяемость тестовой выборки при фиксированном наборе параметров. Данные алгоритмы синтеза необходимы для объективного представительного тестирования новых параллельных многомерных методов глобальной оптимизации с ограничениями.

Введение

Параллельные методы многоэкстремальной оптимизации являются важным инструментом решения прикладных задач, требующих оптимального выбора параметров. Современные возможности параллельных вычислений приводят к новым нетривиальным параллельным реализациям, позволяющим увеличивать размерность решаемых задач (см., например,

[1–3]). Представительное объективное исследование новых методов требует средств генерации многомерных многоэкстремальных задач управляемой сложности с заранее известными решениями. Подобные генераторы разработаны для задач без функциональных ограничений [4, 5] однако для синтеза тестовых задач со сложными ограничениями аналогичные удовлетворительные средства отсутствуют. Новые методы условной глобальной оптимизации для задач вида:

$$f^{**} = f(x^{**}) = \min\{f(x) : x \in G\},$$
$$G = \{x \in D : g_s(x) \le 0, (s = 1, ..., m)\},$$
$$D = \{x \in R^N : a \le x \le b\} \qquad (1)$$

часто исследуются лишь на отдельных тестовых задачах (см., например, [3]). Предложенные математические принципы синтеза порождают псевдослучайные классы задач (1) с управляемой сложностью структуры функции f и допустимого множества G.

Структура многоэкстремальных функций, используемых при синтезе задач

Предлагаемый подход к управляемому синтезу задач вида (1) основан на использовании функций специальной структуры, первоначально предложенных и исследованных в работе [6]. Функции такого вида были применены в [4] при разработке широко используемого генератора многоэкстремальных функций, называемого GKLS-генератором [5]. Эти функции всегда непрерывны, могут иметь требуемый порядок гладкости (0, 1 и 2-го порядка). Вид каждой функции полностью определяется конечным набором параметров, часть из которых зависит от псевдослучайного выбора.

В предлагаемом подходе при синтезе каждой задачи используется $(k+1)$-я функция такой же структуры, что и в GKLS-генераторе, но при измененных правилах порождения набора параметров функций. Функция f_0 с номером 0 выбирается в качестве минимизируемой, а остальные функции $f_s(x)$, $(s = 1, ..., k)$ используются при построении ограничений-неравенств вида $g_j(x) \le 0$, $(j = 1, ..., m \le 2k)$. Выбор параметров для f_0 и остальных функций происходит по отличающимся правилам.

Общей является структура функций. Каждая $f_s(x)$ с номером $s \in \{0, 1, ..., k\}$ строится в заданном гиперпараллелепипеде D в виде параболоида $P_s(x) = \| x - T_s \|^2 + t_s$, который переопределяется по правилам [4, 5] в областях в виде попарно непересекающихся для разных i шаров S_i^s с

центрами в точках $w_i^s \in D$ и радиусами ρ_i^s $(i = 0, 1, ..., \ell_s)$. При этом они не пересекаются также с заданной окрестностью вершины $T_s = w_1^s$ параболоида. Переопределение происходит таким образом, что в точках w_i^s образуются локальные минимумы функции $f_s(x)$ со значениями C_i^s, а на границах шаров происходит полиномиальное сопряжение нужного порядка гладкости с поверхностью параболоида P_s [5, 6]. При этом $w_1^s = T_s$, $C_1^s = t_s$. Точка w_0^s и значение C_0^s всегда выбираются по особому правилу до выбора остальных точек локального минимума w_i^s, $(i = 2, ..., \ell_s)$ таким образом, чтобы w_0^s стала точкой x_*^s безусловного глобального минимума функции f_s. Для этого гарантируется $C_0^s < C_i^s$ $(i = 1, 2, ..., \ell_s)$. Положение точки $w_0^s = x_*^s$ для функции f_s выбирается случайно в D на расстоянии r_*^s от центра T_s параболоида. Радиус ρ_0^s шара S_0^s задается равным величине ρ_*^s, которую можно условно трактовать как радиус «области притяжения» безусловного глобального минимума для f_s. Положение остальных локальных минимумов w_i^s для $(i = 2, ..., \ell_s)$ выбирается случайно по правилам, модифицированным для $s \geq 1$ по отношению к [4, 5], а затем – по специальному алгоритму [4, 5] вычисляются радиусы ρ_i^s шаров S_i^s $(i = 2, ..., \ell_s)$.

Заметим, что для минимизируемой функции, т.е. при $s = 0$, положение вершины T_0 параболоида в соответствии с [4, 5] выбирается случайно во множестве D, и случайное размещение точек локального минимума w_i^0 для $(i = 2, ..., \ell_0)$ также выполняется по правилам [4, 5]. При формировании $f_s(x)$ с $s \geq 1$ выбор T_s происходит по иному принципу, а также изменяются правила расположения точек w_i^s для $i \geq 2$.

Основные принципы порождения многоэкстремальных задач со сложными ограничениями, два типа задач

В предлагаемом подходе можно синтезировать 100 задач с ограничениями, где каждая задача с номером t может иметь одну из двух форм $\text{form}_t = -1$ или $\text{form}_t = +1$, выбираемых случайно с заданными вероятностями p_f и $1 - p_f$ (рис. 1). В задачах, имеющих форму (–1), безусловный глобальный минимум $x^* = x_*^0 \in G$ – допустимому множеству, т.е. совпадает с условным глобальным решением x^{**}. В задачах формы (+1) $x^{**} \neq x^* \notin G$. В структуре любой задачи всегда присутствует точка A с управляемым

набором активных в этой точке ограничений. В задачах формы (+1) $A = x^{**}$ совпадает с условным глобальным минимумом, а в задачах формы (–1) A является условным локальным минимумом со значительной областью притяжения. Точка A выбирается на отрезке $[T_0, x^*]$ по правилам, зависящим от формы задачи: либо случайно в некоторой окрестности точки T_0, либо в ρ_*^0 – окрестности точки x^* как решение уравнения

$$f_o(A) = f^{**},\ A \in \{x : x = x^* + (T_0 - x^*)\tau;\ \tau \in [0, \rho_*^0 / r_*^0]\}. \qquad (2)$$

Из описания структуры функции f_0 и правила выбора точки A следует, что антиградиент этой функции в точке A вне зависимости от формы задачи всегда сонаправлен нормированному вектору $V_* = \text{form}_t(x^* - T_0) / \| x^* - T_0 \|$.

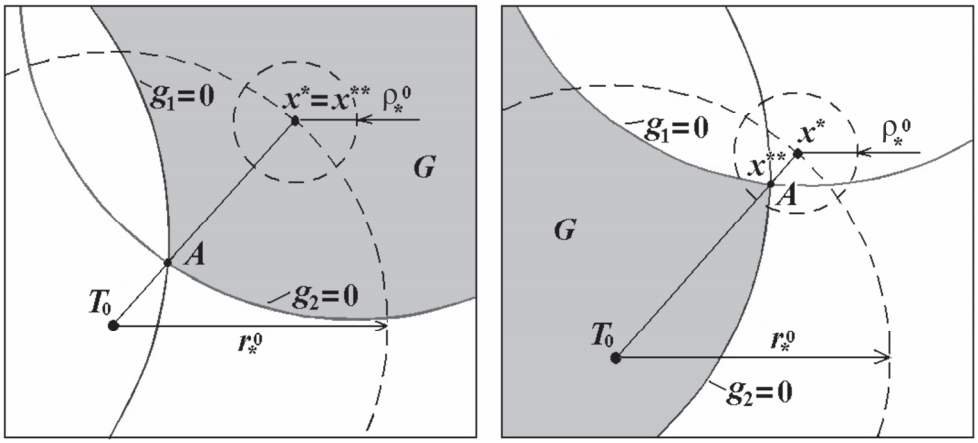


Рис. 1. Локальная структура фрагментов границ
в задачах двух форм: (–1) и (+1) с $A \neq x^{**} = x^*$ и с $A = x^{**} \neq x^*$

Число ограничений m_A, активных в точке A, выбирается случайно в заданных пределах от $m_A^{\min} \geq 1$ до $m_A^{\max} \leq N$. Указанное выше значение $k = m_A$. Для каждого такого ограничения $g_s(x) \leq 0$, $(s = 1, ..., k)$ случайно (с заданными вероятностями) выбирается его тип: $\text{type}_s = +1$ – локально выпуклое, $\text{type}_s = +1$ – локально вогнутое. Основные ограничения $g_s(x) \leq 0$ строятся на основе базовых функций f_s, описанных выше:

$$g_s(x) = (f_s(x) - f_s(A)) \cdot \text{type}_s \cdot 10^{p_s} \leq 0, \qquad (3)$$

где степени p_s случайно выбираются из заданного диапазона $[-p, p]$ для размасштабирования значений функций ограничения.

Ограничения из (3) назовем основными. Заметим, что для обеспечения правильной локальной ориентации в окрестности точки A гладких фрагментов границы множества G достаточно при случайном выборе по-

ложений всех локальных минимумов w_i^s, $(i = 0, 1, ..., \ell_s)$ функции f_s не размещать их в заданных окрестностях точек A и x^{**}.

Изменения в генерации минимизируемой функции

При генерации минимизируемой функции f_0 используются правила построения, описанные в п. 1. Перед их применением должно быть определено значение f_0^* в безусловном глобальном минимуме этой функции. Оно выбирается в зависимости от заданного пользователем желаемого значения f^{**} этой функции в условном глобальном минимуме x^{**} задачи (при этом $f^{**} < t_0$). Если задача имеет форму $\mathrm{form}_t = -1$, полагается $C_0^0 = f_0^* = f^{**}$, если же $\mathrm{form}_t = 1$, на промежутке $(0, (t_0 - f^{**})\nu_1]$, $(0 < \nu_1 < 1/4)$ случайно выбирается число Δf_0^*, после чего полагается $C_0^0 = f_0^* = f^{**} - \Delta f_0^*$. Кроме того, накладываются новые требования на выбор C_i^0, $(i = 2, .., \ell_s)$ – значений f_0 в ее локальных минимумах. А именно, требуется $C_i^0 \geq f^{**} + \delta f_0^*$, где δf_0^* случайно выбирается из промежутка $(0, (t_0 - f^{**})\nu_2]$, $(0 < \nu_2 << 1)$. Для задач формы (+1) после численного определения положения точки E_l из решения уравнения (2) выполняется коррекция ранее заданного значения f^{**} на значение, соответствующее точке A: $f^{**} = f_0(A)$.

Принципы генерации базовых функций для построения ограничений

Под базовыми функциями понимаются функции f_s из формулы (3), $(s = 1, ..., k = m_A)$. В силу замечания, сделанного в конце п. 2, внешние нормали N_s в точке A к порождаемым ими гладким фрагментам границы допустимого множества G определяются как

$$\mathrm{N}_s = (A - T_s)\mathrm{type}_s . \qquad (4)$$

Из теоремы Куна – Таккера следует, что для локальной оптимальности точки A в задаче (1) необходимо, чтобы ранее построенный вектор V_* принадлежал конусу, натянутому на совокупность внешних нормалей к фрагментам активных в точке A ограничений: $V_* \in \{ v : v = \sum_{s=1}^{m_A} \lambda_s \mathrm{N}_s, \lambda_s \geq 0 \}$. Поэтому построение набора функций f_s, $(s = 1, ..., m_A)$ начинается с построения случайной реализации с заданными свойствами набора линейно независимых векторов $d_1, ..., d_{m_A}$, образующих конус K, содержащий вектор V_*:

$$\mathrm{K} = \{ v : v = \sum_{s=1}^{m_A} \lambda_s d_s, \lambda_s \geq 0 \}. \quad (5)$$

Далее положение точек T_s выбирается так, чтобы векторы N_s из формулы (4) совпадали по направлению с векторами d_s. Длины векторов N_s принимаются равными параметрам генерации r_*^s. При этом глобальный минимум каждой базовой функции f_s размещается от вершин T_s ее параболоида на том же расстоянии, что и точка A. Априори границы $r_*^{\min}$, $r_*^{\max}$ случайного выбора значений r_*^s задаются как параметры генерации. Однако для задач формы (–1) эти границы специальным образом корректируются, чтобы исключить возможность отсечения глобального минимума минимизируемой функции x_*^0 одной из функций ограничений из формулы (3). Формулы коррекции получены. После вычисления r_*^s определяются значения $T_s = A - \mathrm{type}_s r_*^s d_s / \| d_s \|$, после чего по описанным в п. 1–2 правилам вычисляются: положение глобального минимума каждой функции f_s, значение в нем $C_0^s < t_s$, положение точек остальных локальных минимумов w_i^s, а также настраиваются радиусы шаров S_i^s («области притяжения»), генерируются значения в локальных минимумах $C_i^s > C_0^s$.

Построение конуса K из формулы (5) выполняется специальными алгоритмами, включающими построение деформированного симплекса управляемого размера с m вершинами в случайно ориентированном $(m-1)$-м подпространсве $(N-1)$-размерного линейного многообразия, ортогонального вектору V_*.

Дополнительные ограничения и возможность настройки относительной меры допустимой области

По устанавливаемым правилам синтеза тестовых задач общее число ограничений m должно быть больше числа основных ограничений m_A, но не превышать $2m_A$. Дополнительные ограничения строятся на основе тех же базовых функций $f_1, ..., f_{m_A}$, что и основные. При этом ограничители на их значения накладываются с противоположной стороны. Несколько упрощая описание, примем, что при этом используются первые $n = m - m_A$ базовых функций. Для возможности регулировки относительной меры допустимого множества значения ограничителей для f_s сделаем изменяющимися от $f_s(A)$, как в формуле (3), до другого предельно возможного значения $\hat{f}_s$, в зависимости от некоторого коэффициента смешивания $\beta_s(\alpha) = \min\{\alpha; \alpha_s^{\max}\}$, где $\alpha \in [0, 1]$.

Для базовых функций с $\mathrm{type}_s = +1$ (локально выпуклые) $\hat{f}_s = f_s(x_*^s) = C_0^s$, для базовых функций с $\mathrm{type}_s = -1$ (локально вогнутые) величина $\hat{f}_s$ определяется как максимальное значение функции $f_s(x)$ на множестве вершин D. Дополнительные ограничители $\alpha_s^{\max}$ на α необходимы только для задач формы (–1), поскольку для них при уменьшении меры допустимой области возможен выход точки $x^{**} = x^*$ из допустимого множества. Определено, что для задач с $\mathrm{form}_t = +1$ при $\mathrm{type}_s = -1$ и при $\mathrm{type}_s = +1$ значения $\alpha_s^{\max}$ соответственно равны:

$$\alpha_s^{\max} = \gamma(\hat{f}_s - t_s - \| x^* - T_s \|^2) / (\hat{f}_s - f_s(A)),$$
$$\alpha_s^{\max} = \gamma(t_s + \| x^* - T_s \|^2 - C_0^s) / (f_s(A) - C_0^s),$$

где $\gamma = 0{,}99$. Для задач с $\mathrm{form}_t = +1$ всегда $\alpha_s^{\max} = 1$.

Дополнительные ограничения строятся в виде:

$$g_{s+\ell}(x) = -(f_\ell(x) - R_\ell(\beta_\ell(\alpha)) \cdot \mathrm{type}_\ell \cdot 10^{p_{s+\ell}} \le 0,\ \ell = 1, ..., n, \qquad (6)$$

где $R_\ell(\beta) = \hat{f}_\ell + (f_\ell(A) - \hat{f}_\ell)\beta$. В совокупности с основными ограничениями они определяют допустимое множество $G(\alpha)$, зависящее от выбора α. Далее методом Монте-Карло приближенно определяются на сетке $0 = \alpha_1 < \alpha_2 < ... < \alpha_q = 1$ значения относительной меры $\mu(\alpha)$ множества $G(\alpha)$ и затем по желаемому значению μ^* относительной меры вычисляется $\alpha = \alpha^*$, как значение, минимизирующее невязку в уравнении $\mu(\alpha) = \mu^*$. Окончательно в дополнительных ограничениях (6) используется $\alpha = \alpha^*$.

6. Примеры имитации синтеза многоэкстремальных задач в R^2

Имитация работы предложенных алгоритмов подтверждает разнообразие и управляемую сложность порождаемых задач (рис. 2).

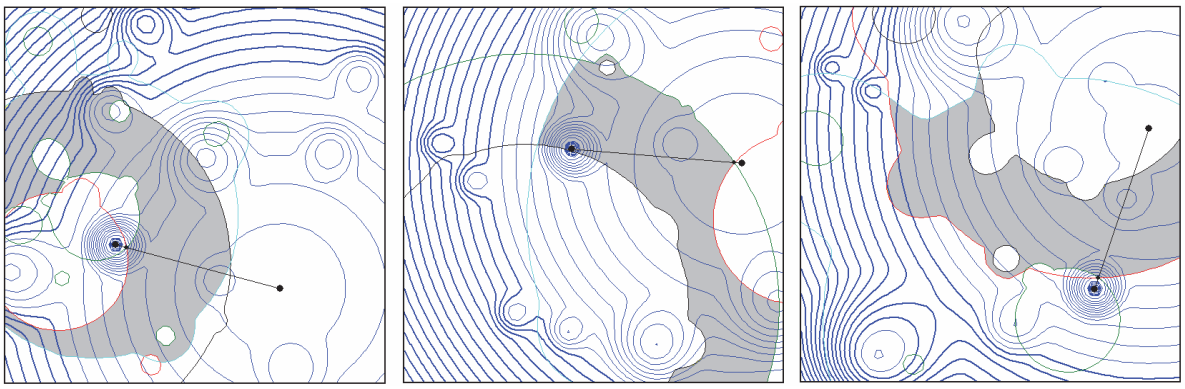


Рис. 2. Примеры видов задач в R^2, порождаемых предложенными алгоритмами, при определенных значениях параметров генерации, включая $\mu^* = 0{,}2$

При этом значение условного глобального минимума x^{**}, f^{**} каждой построенной задачи и остальные параметры всегда известны.

Спсок литературы

1. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. – Dordrecht: Kluwer Academic Publishers, 2000.

2. Параллельные вычисления в задачах глобальной оптимизации: монография / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов. – М.: Изд-во МГУ, 2013. – 280 с.

3. Сидоров С.В. Параллельный двухуровневый индексный алгоритм глобальной оптимизации // Математическое моделирование. Оптимальное управление. Вестник ННГУ. – 2012. – № 5 (2). – С. 206–211.

4. Software for generation of classes of test functions with known local and global minima for global optimization / M. Gaviano, D.E. Kvasov, D. Lera, Ya.D. Sergeyev // ACM Transactions on Mathematical Software. – 2003. – Vol. 29 (4). – P. 469–480.

5. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008.

6. Gaviano M., Lera D. Test functions with variable attraction regions for global optimization problems // Journal of Global Optimization. – 1998. – Vol. 13 (2). – P. 207–223.

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

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

МОДИФИЦИРОВАННЫЙ МЕТОД МОНТЕ-КАРЛО ДЛЯ МОДЕЛИРОВАНИЯ РАСПРОСТРАНЕНИЯ ЗОНДИРУЮЩЕГО ИЗЛУЧЕНИЯ В ЗАДАЧАХ ОПТИЧЕСКОЙ ДИФФУЗИОННОЙ СПЕКТРОСКОПИИ

Описан алгоритм моделирования распространения зондирующего излучения методом Монте-Карло, адаптированный для решения задач оптической диффузионной спектроскопии. Описан предложенный авторами модифицированный метод Монте-Карло, предназначенный для более эффективного моделирования в системе источник – детекторы. Приведены результаты решения задачи моделирования распространения излучения от источника к детектору в голове человека обоими методами. Показано превосходство модифицированного метода при решении данного класса задач.

Введение

В настоящее время в медицинских исследованиях существует потребность в развитии новых безопасных и доступных методов диагностики, поскольку используемые традиционные методы (МРТ, КТ, УЗИ) имеют ряд ограничений. Например, МРТ и КТ требуют наличия дорогостоящего оборудования для проведения диагностики, а УЗИ характеризуется плохим пространственным разрешением, и, как следствие, ограниченным кругом решаемых задач. Классом наиболее перспективных методов диагностики, которые могут применяться как в сочетании с существующими методами, так и в некоторых случаях вместо них, являются оптические методы. Эти методы безопасны для человека, не требуют применения дорогостоящего оборудования и могут применяться для решения широкого класса задач.

Одним из примеров таких методов является метод оптической диффузионной спектроскопии (ОДС), применяемый для диагностики биологических тканей на больших глубинах (десятки миллиметров). Использование метода ОДС позволяет решать такие задачи, как диагностика раковых опухолей, в частности рака груди; мониторинг активности зон коры головного мозга; планирование фотодинамической терапии; мониторинг состояния пациента при хирургическом вмешательстве; определение состояния кожных покровов и др.

Для применения метода ОДС на практике используется система источников и детекторов зондирующего излучения, устанавливаемых на теле пациента (рис. 1). Излучение от источника, проходя через целевую область биоткани (например, опухоль), дает определенный сигнал на детекторах.

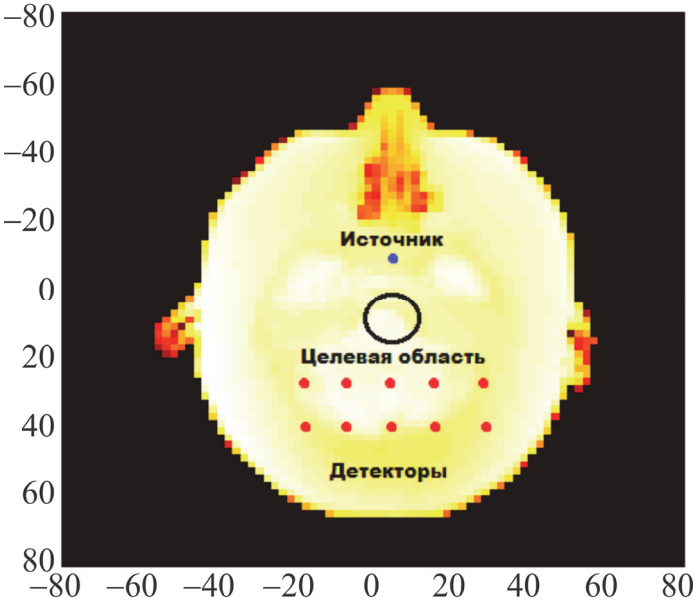


Рис. 1. Система источников и детекторов излучения
на голове пациента

При изменении статуса целевой области этот сигнал меняется, позволяя делать определенные диагностические выводы. Отметим, что исследуемые биоткани являются в общем случае гетерогенными средами со сложной геометрией.

Наиболее широко применяемым подходом для решения данного класса задач являются методы Монте-Карло – моделирования переноса зондирующего излучения в сложных гетерогенных средах [1]. Формально применения этих методов достаточно для решения описанного класса задач, однако данные методы характеризуются значительной вычислительной трудоемкостью, что не позволяет использовать их для решения ряда прикладных задач на практике. Таким образом, актуальной является проблема разработки новых, более эффективных с точки зрения быстродействия методов и алгоритмов моделирования ОДС.

Моделирование распространения излучения с использованием стандартного метода 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 три класса – диффузионно-отраженные, поглощенные и прошедшие (рис. 2).

В задачах оптической диффузионной спектроскопии дополнительно вводится понятие детектора – некоторой области на внешней границе среды, которая способна улавливать прошедшие через нее фотоны. И наибо-

лее интересными являются траектории диффузионно-отраженных фотонов, прошедших через тот или иной детектор [2]. Полученная информация позволяет определить взаимное расположение источника и детекторов с тем, чтобы траектории фотонов, попавших в эти детекторы, проходили через интересующую исследователя область.

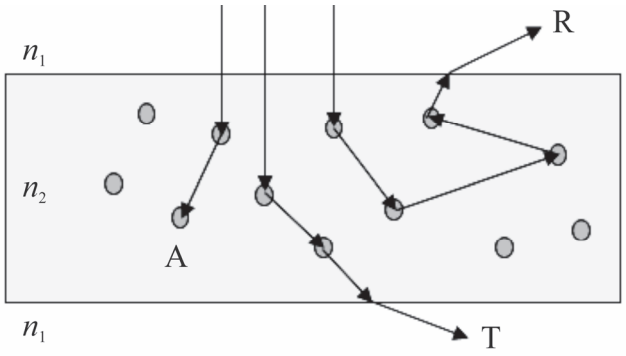


Рис. 2. Траектории движения и типы фотонов:
R – диффузионно-отраженный фотон; A – поглощенный фотон;
T – прошедший фотон; *n* – коэффициент отражения

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 [3].

Модифицированный метод Монте-Карло для моделирования распространения излучения в системе источник – детекторы

Основной недостаток стандартного метода Монте-Карло в данной задаче состоит в том, что полезными здесь являются только те фотоны, которые попали на детектор. Причем обычна ситуация, когда из 10^6 пакетов на целевой детектор попало только 100. А значит, для получения достаточного количества фотонов на детекторе необходимо проводить трассировку значительно большего их числа, что занимает существенное время.

С целью сокращения объема обрабатываемой информации возможно применение метода существенной выборки. А именно, нужно выбирать новое направление движения фотона близким к направлению на детектор. При этом изменение исходной функции распределения компенсируется уменьшением веса пакета фотонов.

Данная идея предлагалась ранее, однако разработанные на ее основе методы поддерживали работу только с однородным плоскопараллельным и полубесконечным слоем [4, 5]. Авторами же данной работы *предложен но-*

вый метод, позволяющий проводить моделирование гетерогенных многослойных объектов с произвольной геометрией слоев [6].

Применение метода существенной выборки сосредоточено на шаге вычисления нового направления движения фотона. Вводится коэффициент выживаемости

$$S\left(v_f, v_a\right) = \left[0{,}1 + 0{,}9\frac{1 + v_f v_a}{2} \right]^{\varepsilon},$$

где v_f – новое направление движения фотона; v_a – вектор притяжения (направление, приводящее фотон на детектор); ε – коэффициент притяжения. Векторы v_f и v_a имеют единичную норму.

Генерируется направление движения фотона v_f стандартным способом. После этого генерируется значение равномерно распределенной на интервале $[0,1)$ случайной величины ξ. Если $S\left(v_f, v_a\right) > \xi$, тогда сгенерированное направление v_f принимается. Иначе генерируется новое направление и процесс повторяется.

Для того чтобы сохранить правильность итоговых результатов моделирования, необходимо выполнять корректировку веса пакета фотонов одновременно с выбором нового направления.

Вектор притяжения должен соответствовать такому направлению движения фотона из текущей позиции, которое бы приводило фотон на детектор с наибольшей вероятностью. Предлагается следующая схема его выбора: вектор притяжения выбирается как направление движения фотона, с наибольшей вероятностью приводящее фотон из текущей позиции на детектор с точки зрения диффузионной теории.

Следует отметить, что авторами предложено также *использование диффузионной модели, в дополнение к рассеянию учитывающей еще и поглощение среды* при выборе вектора притяжения [6].

Результаты применения стандартного и модифицированного методов Монте-Карло в задаче моделирования распространения зондирующего излучения в голове человека

Основное достоинство модифицированного метода Монте-Карло состоит в том, что для получения достаточного объема статистики для целевого детектора требуется выполнять трассировку значительно меньшего числа фотонов, чем при использовании стандартного метода Монте-Карло.

На рис. 3 представлены результаты моделирования траекторий движения фотонов в голове человека от источника к целевому детектору (в двумерном сечении). Цветом отмечена частота посещения области: че-

рез более светлые области проходит большее число фотонов. Используется логарифмическая цветовая шкала (белый – желтый – красный – черный). Точечный источник излучения располагается в начале координат. Целевой детектор находится справа от него на расстоянии в 30 мм. Белыми линиями обозначены границы тканей головы.

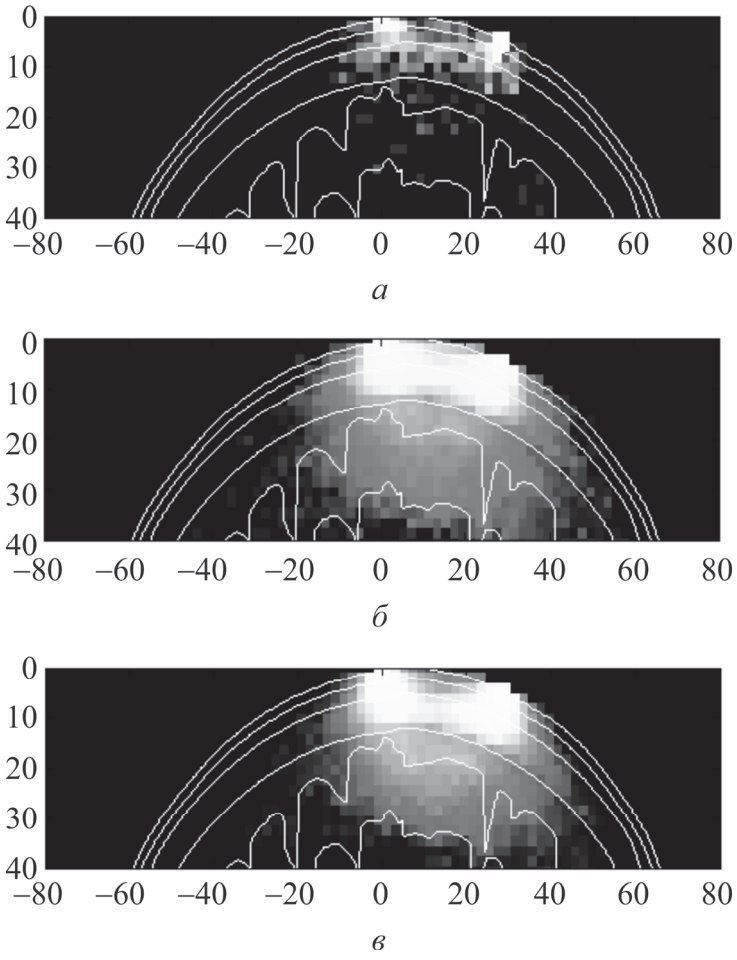


Рис 3. Двумерные сечения траекторий движения фотонов от источника излучения к детектору для стандартного метода с 10 млн (*а*) и 1 млрд (*б*) фотонов, и модифицированного метода Монте-Карло с 10 млн фотонов (*в*)

В результате трассировки 10 млн фотонов модифицированным методом Монте-Карло удалось получить 8597 фотонов на целевом детекторе. Примерно такое же количество фонов попадает на целевой детектор только при моделировании 1 млрд фотонов стандартным методом (9302 фотона). При этом результаты моделирования сигнала на детекторах достаточно хорошо совпадают. А при моделировании 10 млн фотонов стандартным методом Монте-Карло хорошо заметен недостаток статистики. Отметим, что число фотонов, попавших на целевой детектор, в этом случае равно 94.

При этом *время моделирования 1 млрд фотонов стандартным методом Монте-Карло составляет 54 ч 35 мин, а применение модифицированного метода позволяет решить поставленную задачу путем моделирования 10 млн фотонов, сокращая это время до 1 ч 12 мин*.

Для вычислений использовался кластер на базе CPU Intel Xeon L5630 2,13 ГГц (4 ядра), на каждом узле кластера установлено 2 CPU, 24 ГБ RAM, операционная система MS Windows Server HPC Edition. Для решения данной задачи использовалось 10 узлов кластера.

Заключение

В данной работе приведено описание предложенного авторами модифицированного метода Монте-Карло для решения задач моделирования распространения зондирующего излучения в системе источник – детекторы. Описаны результаты решения задачи моделирования распространения излучения от источника к детектору в голове человека обоими методами. Показано, что использование модифицированного метода позволяет сократить время моделирования для указанного класса задач на порядок.

Работа выполнена в рамках гранта РФФИ № 14-02-31549 (мол_а) при организационной поддержке Лаборатории информационных технологий ННГУ.

Список литературы

1. Wang L., Jacques S.L., Zheng L. MCML – Monte Carlo modeling of light transport in multi-layered tissues // Comput. Methods Programs Biomed. – 1995. – Vol. 47, № 2. – P. 131.

2. 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.

3. 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.

4. Chen N.G., Bai J. Estimation of quasi-straightforward propagating light in tissues // Phys. Med. Biol. – 1999. – Vol. 44, № 7. – P. 1669.

5. Chen N. Controlled Monte Carlo method for light propagation in tissue of semi-infinite geometry // Appl. Opt. – 2007. – Vol. 46, № 10. – P. 1597.

6. Горшков А.В., Кириллин М.Ю., Гергель В.П. Улучшенный метод Монте-Карло для моделирования распространения зондирующего излучения в задачах оптической диффузионной спектроскопии // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2014. – № 1. – С. 239.

К.И. Дикарев

Нижегородский государственный университет им. Н.И. Лобачевского

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ В ЗАДАЧАХ ПОИСКА ОПТИМАЛЬНЫХ РЕЖИМОВ НАСОСНЫХ СТАНЦИЙ

Приводятся результаты адаптации численного алгоритма решения задач оптимального функционирования насосных станций для случая параллельной вычислительной среды с использованием функций библиотеки MPI. В качестве примера рассмотрена задача оптимизации работы насосной станции второго подъема с пятью рабочими насосными агрегатами. Показано, что работа алгоритма в параллельном режиме является эффективной.

Введение

На современном этапе развития систем водо- и теплоснабжения и водоотведения населенных пунктов и промышленных предприятий, актуальной является задача обеспечения эффективных режимов их работы с точки зрения затрат электроэнергии, потребляемых насосными станциями в целях перекачки жидкой среды. Сюда, прежде всего, относятся насосные станции I и II подъема, а также группы насосов теплофикационных установок ТЭЦ.

В настоящее время в практику эксплуатации предприятий водоснабжения и водоотведения постепенно внедряются автоматизированные системы мониторинга и управления водозаборными узлами, позволяющие существенно повысить эффективность их функционирования. Одной из важных функций подобных систем является оперативное расчетное определение режимов работы станционных насосных групп, оптимальных с точки зрения энергозатрат. С целью обеспечения адекватного реагирования системы автоматизированного управления на изменение параметров в управляемых объектах, указанные вычисления должны обладать не только достаточной степенью точности, но и выполняться за короткие промежутки времени. Один из подходов к сокращению времени получения решений сложных задач неизменно связан с привлечением многопроцессорных вычислительных технологий.

В соответствии с требованиями ряда нормативных документов вновь проектируемые промышленные насосные станции следует укомплектовывать центробежными насосами с регулируемыми приводами [1]. Возможность варьирования рабочих частот вращения для валов центробежных насосов позволяет производить подбор оптимальных и согласованных режимов работы оборудования.

В рамках данной работы алгоритм расчетного определения оптимальных с точки зрения потребляемой мощности гидравлических режимов насосных станций, предложенный в [2, 3], расширен для случая применения многопроцессорной вычислительной среды.

Математическая постановка задачи поиска оптимального режима насосной станции

Требование согласования режимных параметров для системы водоснабжения населенного пункта по критерию минимизации суммарных затрат можно сформулировать следующим образом: необходимо при заданных значениях требуемых напоров и расходов у конечных абонентов водопроводной сети определить такие «управления» для насосных агрегатов станции, при которых данные значения будут обеспеченны с наименьшими суммарными затратами мощности.

При оптимизации процесса функционирования отдельной насосной станции необходимо учитывать функциональную зависимость, связывающую потребляемую электроприводом каждого насосного агрегата мощность с подачей перекачиваемой жидкой среды при условии, что все прочие параметры, такие как требуемый напор в водопроводах и плотность водной среды, фиксированы.

В настоящей работе, как и в работе [3], указанная нелинейная зависимость аппроксимируется посредством конечного числа кусочно-линейных участков, при этом весь интервал аппроксимации по подаче насосов разбивается на n подинтервалов, с равномерным шагом, и заменой на каждом из данных интервалов исходной зависимости линейной функцией [3]. Будем также считать, что все насосные агрегаты на станции соединены по параллельной схеме.

Множество всех насосных агрегатов разобьем на подмножества, которые будем называть группами, предполагая, что каждая группа содержит множество насосных агрегатов с близкими характеристиками. Пусть $i = \overline{1,m}$ – номера групп агрегатов насосной станции, $j = \overline{1,n}$ – номера подынтервалов, на которые дискретизированы возможные значения подачи воды для всех групп насосных агрегатов. Обозначим через m_i^- и m_i^+ соответственно минимально возможное и максимально допустимое число насосных агрегатов i-й группы, которые могут быть задействованы, $i = \overline{1,m}$; через Q_{ij}^- и Q_{ij}^+ соответственно – минимально возможный и максимально допустимый объемы подачи воды, которые могут быть обеспечены i-м насосным агрегатом, если его производительность будет соответство-

вать j-му интервалу, $0 \le Q_{ij}^{-} \le Q_{ij}^{+}$, $i=\overline{1,m}$, $j=\overline{1,n}$; a_{ij} и b_{ij} – коэффициенты линейной функции, определяющей зависимость потребляемой мощности насосного агрегата i от его подачи при функционировании в рамках подинтервала j допустимых объемов подач, $i=\overline{1,m}$, $j=\overline{1,n}$; $Q^{\text{вых}}$ – плановый объемный расход воды, которую должна обеспечить насосная станция.

Обозначим через x_i количество насосных агрегатов *i*-й группы, которые будут работать на насосной станции в планируемом периоде, $i=\overline{1,m}$; y_i – объем подачи воды, который будет обеспечен насосным агрегатом *i*-й группы, $i=\overline{1,m}$; $z_{ij}=1$, если насосный агрегат *i*-й группы будет работать в *j*-м интервале допустимых объемов подач, и $z_{ij}=0$, в противном случае $i=\overline{1,m}$, $j=\overline{1,n}$.

Тогда математическая модель насосной станции включает соотношения:

$$\sum_{i=1}^{m} x_i y_i = Q^{\text{вых}},\ m_i^{-} \le x_i \le m_i^{+},\ i=\overline{1,m},\ \sum_{j=1}^{n} z_{ij} Q_{ij}^{-} \le y_i \le \sum_{j=1}^{n} z_{ij} Q_{ij}^{+},$$

$$i=\overline{1,m},\ \sum_{j=1}^{n} z_{ij}=1,\ i=\overline{1,m},\ x_i - \text{целые},\ i=\overline{1,m};\ z_{ij} \in \{0,1\},\ i=\overline{1,m},\ j=\overline{1,n}.$$

Ограничения модели включают в себя условия обеспечения требуемого расхода воды на выходе станции согласно графику, ограничения по возможному количеству используемых насосных агрегатов, а также по возможной подаче воды каждым из них, условие существования только одного рабочего интервала производительности по объему подачи воды для каждого насоса, естественные условия на переменные.

Критерием оптимизации для рассматриваемой задачи минимизации затрат на функционирование участка системы отопления является функционал

$$F(\vec{x},\vec{y},Z) = \sum_{i=1}^{m}\sum_{j=1}^{n} (a_{ij} y_i + b_{ij}) z_{ij} x_i \to \min,$$

т.е. суммарная мощность, потребляемая при работе всех насосных агрегатов станции второго подъема, должна иметь минимальное значение. Решение поставленной задачи определяет, какие насосные агрегаты и с какими объемами подачи воды должны работать в рамках насосной станции.

Задача является задачей частично-целочисленного математического программирования и относится к классу NP-трудных задач.

Алгоритм решения задачи

Для решения поставленной выше задачи не существует точных алгоритмов, в общем случае отличных от полного перебора. Можно предложить различные эвристические методы, позволяющие находить их приближенные решения. Показано [2, 3], что для решения подобных задач можно находить точные планы, используя предлагаемую ниже переборную схему решения.

Для нахождения решения необходимо рассмотреть $n^m \cdot \prod_{i=1}^{m}(m_i^+ - m_i^- + 1)$ задач линейного программирования с номерами $s = 1, 2, ...,$ $n^m \cdot \prod_{i=1}^{m}(m_i^+ - m_i^- + 1)$ вида

$$\sum_{i=1}^{m} k_i y_i = Q^{\text{вых}},\ Q_{ij}^- \le y_i \le Q_{ij}^+,\ i = \overline{1,m},$$

$$F(\vec{y}) = \left(\sum_{i=1}^{m} a_{ij_s} y_i + b_{ij_s} \right) k_i \to \min.$$

Для решения каждой такой задачи достаточно рассмотреть 2^m точек, «подозрительных» на допустимость, для каждой точки проверить ее допустимость (подставив значения координат точки в уравнение неразрывности $\sum_{i=1}^{m} k_i y_i = Q^{\text{вых}}$), и далее среди допустимых точек найти точку с минимальным значением критерия. Здесь k_1, k_2, …, k_m – произвольный допустимый набор, в котором k_i – количество насосных агрегатов *i*-й группы, которые будут использованы в рассматриваемом периоде функционирования, $i = \overline{1,m}$.

Пример постановки задачи оптимизации режимов для насосной станции

Рассмотрим конкретный пример, параметры которого соответствуют гипотетической проектируемой насосной станции второго подъема, предназначенной для подачи воды в распределительную сеть. С учетом графика суточного водопотребления станция должна обеспечивать в напорном водоводе максимальный расчетный расход воды 1700 м^3/ч, при расчетном напоре потока 41,6 м. Для функционирования в машинном зале станции предусмотрено 5 насосных агрегатов, при этом три насоса имеют тип Д 500-36, а остальные два насоса – тип Д 320-50 (число типов $m = 2$). Требуется найти такие режимы работы насосов станции, при которых указан-

ные параметры расчетной производительности и напора на выходе станции обеспечивались бы с минимальными затратами мощности.

Для поставленной задачи с применением вышеописанного алгоритма может быть получено следующее оптимальное решение:

– количество работающих насосов типа Д 500-36 равно 3; количество работающих насосов типа Д 320-50 равно 1;

– подача каждого насоса типа Д 500-36 равна 475,54 $м^3$/ч; подача каждого насоса типа Д 320-50 составляет 273,39 $м^3$/ч;

– частота вращения рабочих колес насосов Д 500-36 составляет 1008,20 об/мин., а частота вращения рабочих колес насосов Д 320-50 составляет 1311,30 об/мин;

– суммарная мощность, потребляемая насосами станции водоснабжения, равна 232,23 кВт.

Адаптация алгоритма для параллельной вычислительной среды

Предлагаемый выше алгоритм решения задачи сводится к перебору вариантов решений по сокращенной схеме. Вместе с тем для поставленной выше задачи оптимизации насосной станции с насосами двух типов для обеспечения достаточной точности решения (число подынтервалов разбиения зависимости мощности насосных установок от расхода принималось $n = 4700$) требуется перебор $1{,}06 \cdot 10^9$ вариантов. Для ускорения решения целесообразно прибегнуть к применению параллельной вычислительной среды.

Алгоритм был реализован в виде консольного приложения на языке C++ в среде Visual Studio 2008. Для реализации параллельных вычислений с распределенной памятью был использован стандартный интерфейс MPI. Блок-схема программного алгоритма для случая двух типов насосов на станции приведена на рис. 1. Ввод исходных данных и характеристик насосов осуществляется в однопроцессорном режиме. Перебор произвольных допустимых наборов насосов и рабочих интервалов зависимостей «объем подачи насоса – потребляемая мощность» с выбором оптимального решения осуществляется в многопроцессорном режиме, с равномерным распределением переборных вариантов между процессами. Преимуществом указанного варианта декомпозиции является отсутствие потребности обменов между процессами во время параллельных вычислений. Обмен с использованием функций MPI_Send() и MPI_Recv() осуществляется только на последней стадии, при передаче процессу с нулевым рангом оптимальных значений суммарных мощностей, вычисленных по результатам перебора вариантов другими процессами. Процесс с нулевым рангом находит минимальное значение суммарной мощности из конечного множества передаваемых в него значений.

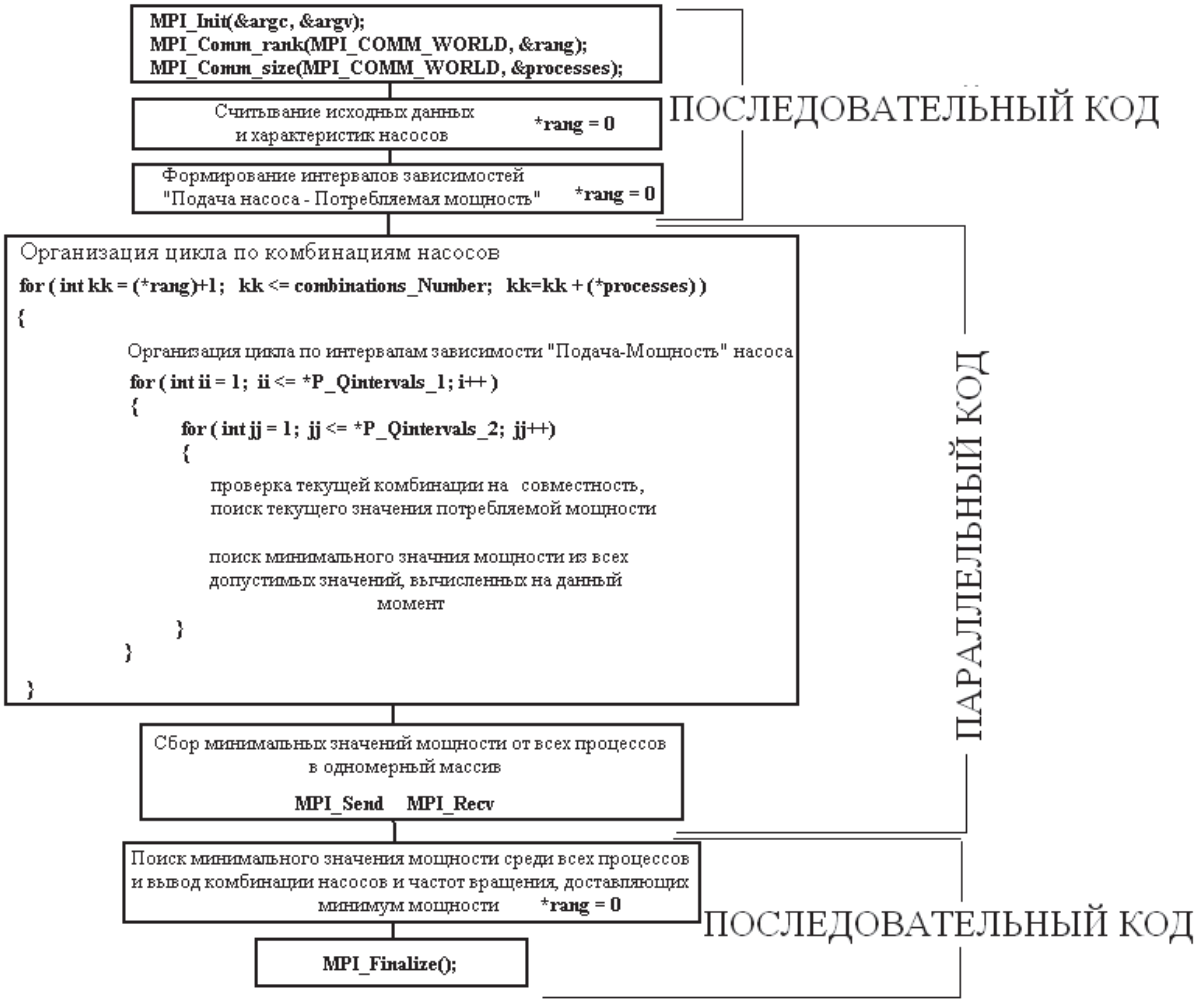


Рис. 1. Блок-схема алгоритма поиска оптимального режима насосной станции

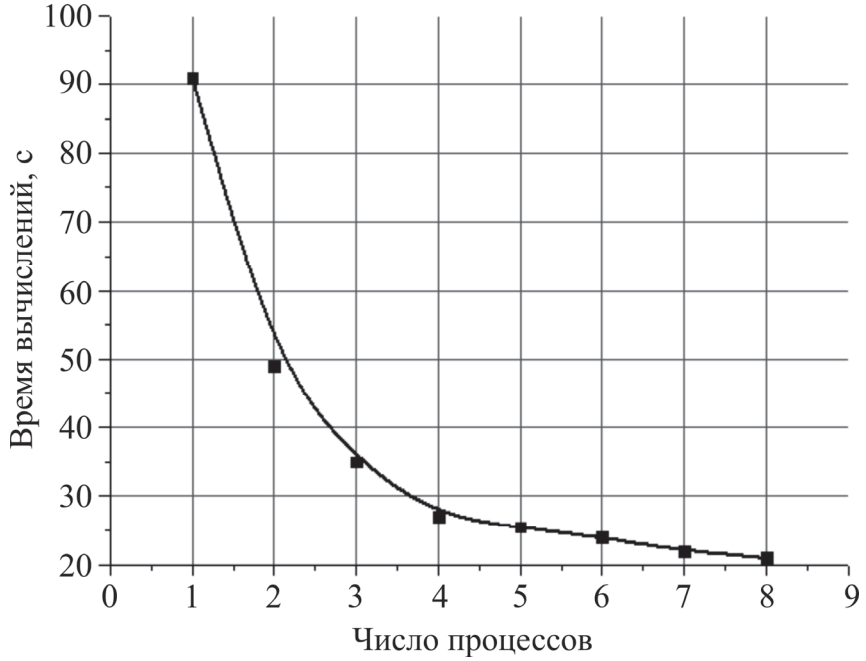


Рис. 2. Зависимость времени поиска оптимального решения
от количества процессов

На рис. 2 показана зависимость времени работы алгоритма для решения нашего примера постановки задачи от количества процессов. Апробация выполнялась на ПЭВМ с 8-ядерным процессором Intel Core(TM) i7-2600K 3,40ГГц, ОЗУ 8,00 ГБ. Из указанных результатов видно, что применение параллельного подхода существенно увеличивает эффективность вычислений при числе процессов ≤ 4. При этом эффективность параллельного алгоритма составляет не менее 0,81.

Заключение

В работе описан подход, позволяющий выполнять решение задач поиска оптимальных режимов для насосных станций водоснабжения, на которых функционируют насосные агрегаты, соединенные по параллельной схеме. Предложен алгоритм решения указанного класса задач, и показана эффективность его параллельной реализации с использованием интерфейса MPI.

Список литературы

1. Садовский С.И., Шаварин В.Н. Энергосбережение на основе регулируемого электропривода: проблемы и решения // Промышленная энергетика. – 2004. – № 9. – С. 2–9.

2. Прилуцкий М.Х., Дикарев К.И. Распределение ресурсов в иерархических системах с активными элементами // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2012. – № 5 (2). – С. 181–189.

3. Костюков В.Е., Прилуцкий М.Х. Распределение ресурсов в иерархических системах. Оптимизационные задачи добычи, транспорта газа и переработки газового конденсата: учеб. пособие. – Нижний Новгород: Изд-во Нижегородского университета, 2010. – 78 с.

К.А. Дробных, А.Н. Половинкин

Нижегородский государственный университет им. Н.И. Лобачевского

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ ГЛУБОКОГО ОБУЧЕНИЯ ДЛЯ УЛУЧШЕНИЯ ТОЧНОСТИ АЛГОРИТМОВ ОЦЕНКИ ВОЗРАСТА ПО ФОТОГРАФИИ ЛИЦА ЧЕЛОВЕКА

Рассматривается задача определения возраста по фотографии лица с использованием активных моделей внешнего вида. Предлагается использовать алгоритмы глубокого обучения для улучшения точности предсказания. Применение глубокого обучения позволило уменьшить ошибку алгоритма на 4,5 %.

Введение

В настоящей работе рассматривается задача оценки возраста по фотографии лица человека. Все алгоритмы решения данной задачи можно представить в виде четырех этапов: выделение лица на изображении, получение признакового описания, уменьшение размерности и применение алгоритмов машинного обучения. В данной работе рассматривается построение признакового описания. Существует множество подходов к решению этой задачи: среди наиболее распространенных можно отметить активные модели внешнего вида [1], BIF (Bio-Inspired Features [2]) и ряд других. Одним из современных подходов к построению признакового описания является использование алгоритмов глубокого обучения, которые находят лучшим образом подходящее для решения конкретной задачи признаковое описание объектов. Выделяют два основных типа алгоритмов глубокого обучения: с учителем (классическим примером являются нейронные сети [3] и без учителя (например, Coates и Ng предложили использовать алгоритм кластеризации *k*-средних [4]).

В данной работе предлагается использовать глубокое обучение для улучшения точности решения задачи. Рассматривается новый алгоритм глубокого обучения с учителем, основанный на применении ансамбля деревьев решений [5], приводятся результаты вычислительного эксперимента.

Базовый алгоритм

В качестве базового алгоритма использовались активные модели внешнего вида. Данные модели содержат статистические модели формы и текстуры объекта, которые могут быть обобщены практически для любого допустимого примера. Для генерации статистической модели формы используется набор изображений с отмеченными ключевыми точками. Перед построением статистической модели необходимо, чтобы формы были независимы от положения, ориентации и масштаба, для чего обычно применяется прокрустов анализ, с помощью которого формы выравниваются таким образом, чтобы минимизировать отклонение от средней.

Для построения статистической модели текстуры необходимо деформировать каждое заданное изображение таким образом, чтобы его контрольные точки совпадали с контрольными точками средней формы (значения интенсивностей пикселей при этом находятся с помощью интерполяции). Текстура представляется в виде вектора значений цвета пикселов соответствующей текстуры, содержащихся в ограниченной средней формой области. Если изображение многоканальное, то формируются векторы пикселов для каждого из каналов, а потом выполняется их конкатенация (как в случае с точками формы). Для того чтобы минимизировать влияние условий освещения, векторы центрируются и нормируются.

Для определения среднего значения нормализованных данных применяется итеративный процесс. За начальную оценку средней текстуры принимается какая-нибудь из текстур, остальные текстуры выравниваются к ней, определяется новая оценка средней с использованием выровненных текстур и т.д.

Далее применяется метод главных компонент для уменьшения размерности.

Использование алгоритмов глубокого обучения

Входные данные для обучения словаря представляют собой множество маленьких патчей (частей) входных изображений (рис. 1).

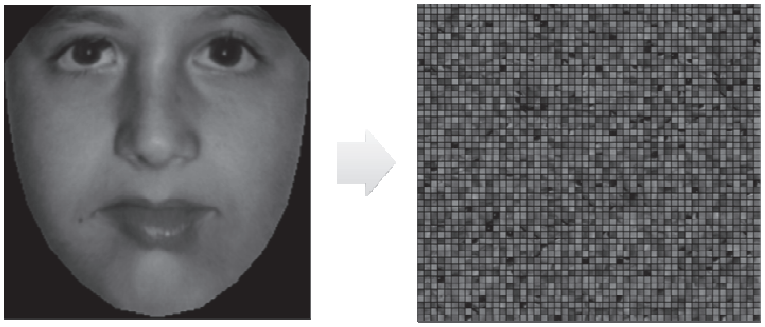


Рис. 1. Входное изображение и извлеченные патчи

Используются патчи размером $n \times n$ пикселов, представленные в виде векторов размером n^2, содержащих значения интенсивности пикселов. Эти патчи собираются с каждого входного изображения. Например, можно собрать по k патчей с каждого изображения (обычно патчи охватывают все изображение с некоторым шагом). Перед тем как применять алгоритм к входным данным, полезно нормализовать яркость и контрастность патчей.

В данной работе используется два варианта глубокого обучения:

1. Глубокое обучение без учителя. К собранным патчам применяется алгоритм кластеризации k-средних, в результате чего получается m кластеров. Для каждого патча, извлеченного из входного изображения с номером j, вычисляется центр кластера, среднеквадратичное расстояние до которого минимально. Далее вычисляется гистограмма, содержащая доли патчей на изображении, принадлежащих каждому кластеру. Полученные гистограммы используются в качестве входных признаков, вместо изображений.

2. Глубокое обучение с учителем. В качестве признаков используются гистограммы, которые строятся следующим образом: на полученных с изображений патчах строится случайный лес (вход – вектор признаков, выход – выход исходной задачи для изображения, с которого был получен вектор признаков). Для каждого патча x_s определяется подмножество тер-

минальных вершин случайного леса, которых достиг вход: $T_s = \{t_{s,i_1}, ..., t_{s,i_K}\}$. Гистограмма, связанная с каждым изображением – доли патчей на изображении, достигших каждой терминальной вершины.

Результаты экспериментов

Для проведения эксперимента использовалась база данных фотографий лиц FG-NET [5]. Ключевые точки для этих фотографий заранее известны, используется 68 точек. База данных содержит 1002 фотографии лиц 82 людей в возрасте от 0 до 69 лет. В качестве алгоритма машинного обучения используется множество бинарных классификаторов, каждый из которых отвечает на вопрос: старше ли данный человек заданного возраста: $f_i(x) = \begin{cases} 1, \text{age}(x) > i \\ 0, \text{age}(x) \le i \end{cases}$.

Возраст оценивается как сумма ответов классификаторов: $f(x) = \sum_{i=0}^{68} f(x)$.

В качестве бинарных классификаторов используется алгоритм машинного обучения GBT (Gradient Boosting Trees [6]), 1000 деревьев.

Глубокое обучение применялось к выровненным изображениям текстур, полученным в результате построения активных моделей внешнего вида.

Использовались следующие параметры для алгоритмов глубокого обучения: патчи 11×11, 1500 патчей с каждого изображения, в случае обучения без учителя 1000 кластеров, с учителем – 40 деревьев глубины 6.

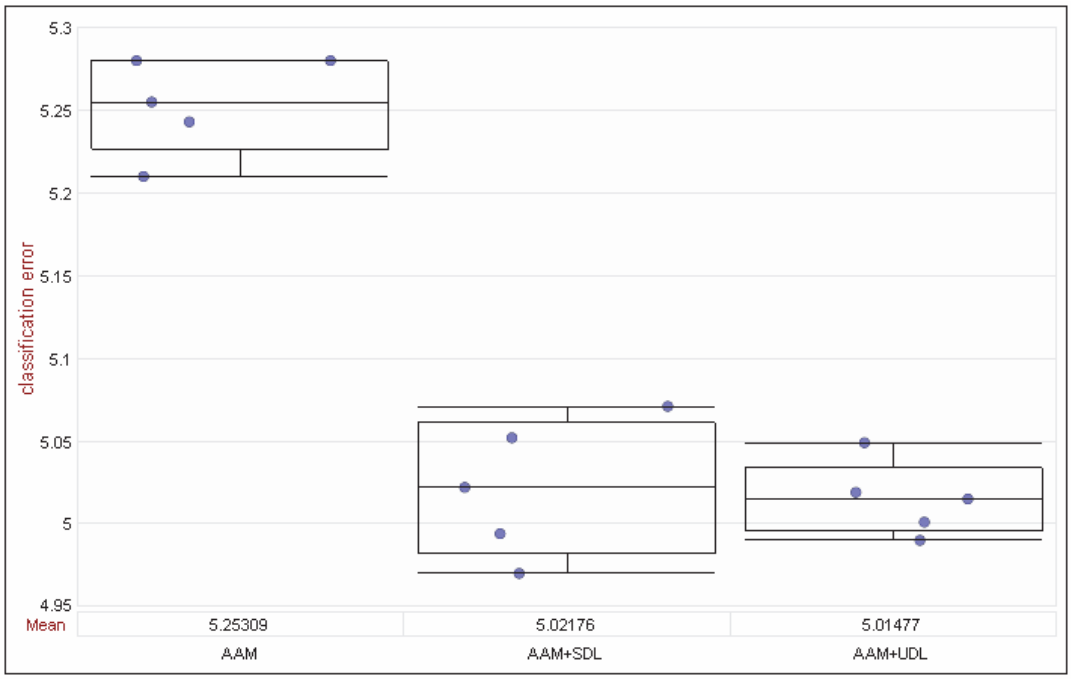


Рис. 2. Численные результаты экспериментов: AAM – активные модели внешнего вида; AAM + SDL – активные модели + глубокое обучение с учителем; AAM + UDL – активные модели + глубокое обучение без учителя

Ошибка вычислялась методом LOPO (Leave One Person Out): тестовая выборка для человека с номером l представляет собой признаки, вычисленные для фотографий только этого человека, обучающая – все остальные, полученные ошибки на тестовой выборке усредняются.

Из графиков видно, что с помощью алгоритмов глубокого обучения удалось уменьшить ошибку на 4,5 %.

Список литературы

1. Cootes T., Edwards G., Taylor C. Active Appearance Models // Proc. European Conference on Computer Vision. – 1998. – P. 484–498.
2. Human age estimation using bio-inspired features / G. Guo, G. Mu, Y. Fu, T. Huang // Computer Vision and Pattern Recognition. – 2009. – P. 112–119.
3. Krizhevsky A., Sutskever I., Hinton G.E. Image Net Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems 25. – 2012. – P. 1097–1105.
4. Coates A., Ng A.Y. Learning Feature Representations with K-means // Neural Networks: Tricks of the Trade, 2012.
5. Martyanov V., Polovinkin A., Tuv E. Image classification with codebook based on decision tree ensembles // Proceedings of the 9th International Conference “Intelligent Information Processing”. – 2012. – P. 480–482.
6. FG-NET Aging Database. – URL: http://www.fgnet.rsunit.com.

М.Р. Еникеев, И.М. Губайдуллин

Институт нефтехимии и катализа РАН, г. Уфа

ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ АНАЛИЗА ПИТТИНГОВОЙ КОРРОЗИИ

Рассматривается задача исследования процесса коррозии путем визуального проводимого эксперимента. В ходе решения задачи обработки последовательности кадров возникла необходимость организации параллельных вычислений с целью ускорения процесса расчета анализируемых поверхностей.

Введение

В последние годы в области изучения поверхностных процессов нашли широкое применение методы визуального контроля: оптическая микроскопия in situ, конфокальная микроскопия, лазерная сканирующая мик-

роскопия и т.д. При всех преимуществах визуальных методов все они имеют один существенный недостаток – трудность в количественном описании полученного изображения в целом. Коллективом авторов проводится изучение механизма коррозионных процессов посредством исследования морфологии поверхности металла с использованием методов компьютерного зрения.

В данной работе рассматривается проблема изучения зарождения питтинговой коррозии. Питтинговая коррозия – коррозия металлов, ведущая к образованию питтингов, т.е. язв, полостей в металле, начинающихся с его поверхности. Существует множество работ по изучению появления локальных коррозионных дефектов (питтингов) на пассивных металлах, и разработано большое количество конкурирующих моделей, описывающих их возникновение. Множество методов было привлечено для изучения процессов возникновения и развития питтингов.

Однако, несмотря на большое количество проведенных исследований, динамика процесса все еще не ясна, и ведутся споры относительно того, каким образом происходит зарождение питтинга под воздействием агрессивных ионов и как эти коррозионные дефекты растут [1]. Обычно для исследования локального дефектообразования в металлах используются электрохимические методы. Недостатком этих методов является отсутствие информации о точных координатах дефектов. Получить такую информацию можно при одновременном использовании оптических методов, являющихся простым и неразрушающим средством наблюдения in situ за динамическими изменениями поверхности электрода. Задача визуального контроля за экспериментом решается методами компьютерного зрения. Компьютерное зрение – теория и технология создания машин, которые могут производить обнаружение, слежение и классификацию объектов.

В компьютерном зрении существует множество подходов к детектированию дефектов поверхности, они могут различаться в зависимости от конкретных задач.

Можно выделить общую схему поиска повреждений поверхности:

1. Предварительная обработка изображения.
2. Сегментация изображения.
3. Анализ параметров дефектов.

Методы компьютерного зрения в задаче поиска объектов

Для решения задачи поиска и оценки дефектов материала был разработан специальный вычислительный модуль, обрабатывающий входные данные, предоставленные пользователем.

Тестовое задание состояло в анализе записей поверхности алюминия в 0,1 М растворе NaCl (pH 11) под действием электрического поля (рис. 1). Задача поиска объектов на изображении сводилась к обнаружению пузырьков выделяемого водорода, по серии снимков необходимо было определить интенсивность этого выделения. Также этот процесс необходимо автоматизировать вне зависимости от кадра изображения, так как один эксперимент включает в себя примерно 150–200 фотографий, и вручную подбирать параметры обработки практически невозможно

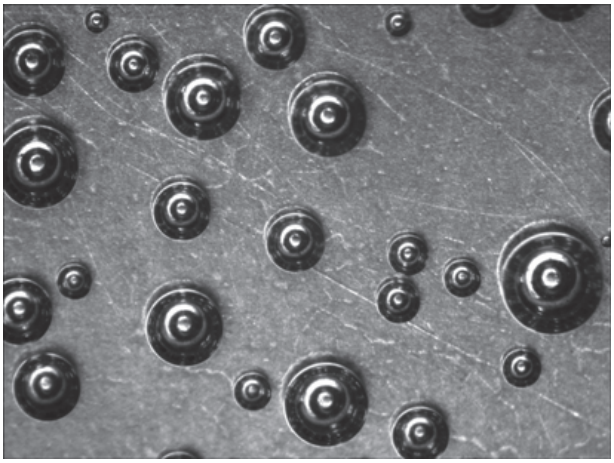


Рис. 1. Кадр из тестового задания по анализу
выхода пузырьков водорода после применения
операции «приведения к серому»

Поиск дефектов можно осуществлять разными способами. В данном случае границы интересующих нас пузырей сливаются с тенью, поэтому могут возникнуть дополнительные трудности при использовании SURF-(Speeded up Robust Features) детектора. Также можно привести тестовые изображения к бинарному виду и разделить объекты по типу: пузырек водорода и фон. Было решено исследовать пути интеллектуальной бинаризации с элементами template matching (сравнение по образцу).

В данной задаче цвет не несет полезной информации и для упрощения обработки изображение приводится к представлению в оттенках серого.

Исследуемые изображения получены с помощью фотокамеры, и это неизбежно сопряжено с рядом проблем. Помимо того, что изображение объекта, как правило, не совпадает по размеру и масштабу с изображениями в эталонной базе данных системы распознавания, оно также подвержено разного рода яркостным искажениям. Для подавления такого рода искажений производится выравнивание гистограммы яркости. В ходе исследования необходимо автоматическое определение параметров пузырей, поэтому было решено использовать метод эквализации гистограммы, при

котором гистограмма изображения подчиняется равномерному закону распределения. Результат применения данного алгоритма к тестовому изображению представлен на рис. 2.

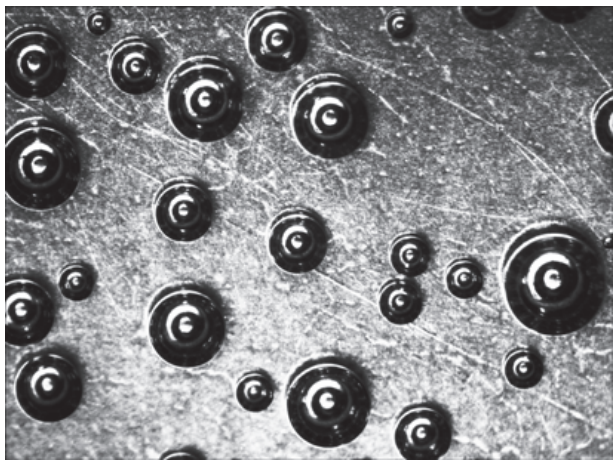


Рис. 2. Изображение после применения
метода эквализации гистограммы

После улучшения контрастности можно приступать к процедуре бинаризации, т.е. операции порогового разделения, которая в результате дает бинарное изображение. На рис. 2 яркость пикселов границы пузырьков и тень пузырька практически одинаковая, следовательно, при бинаризации не удастся сразу отделить пузырьки от фона.

Для извлечения информации о пузырьке и отделения его от тени воспользуемся аппаратом математической морфологии. Основные операции: наращивание, эрозия, замыкание и размыкание [2]. После применения морфологических операций остается найти все замкнутые внутри белого фона темные регионы – это и будут границы предполагаемых пузырей (рис. 3).

Рис. 3. Результат применения морфологических операций

В качестве метода бинаризации был выбран метод Отсу [3], так как при его использовании перед пользователем не стоит проблема ручного выбора порога бинаризации. Смысл метода Отсу заключается в следующем: диапазон яркостей [0; L] (в нашем случае L = 255) изображения делится на две части пороговым значением T. В алгоритме Отсу минимизация внутриклассовой дисперсии эквивалентна максимизации межклассовой дисперсии.

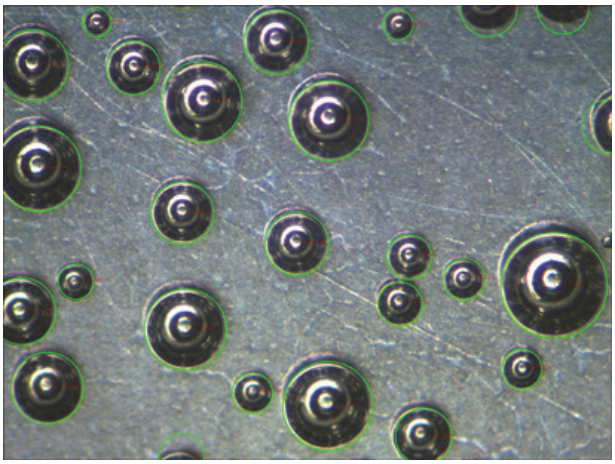


Рис. 4. Работа детектора пузырьков
на поверхности металла

После того как необходимые контуры найдены, следует определить параметры пузырьков. Поскольку контур пузырька представляет собой окружность, то, используя метод наименьших квадратов для окружности, найдем необходимые значения радиуса и центра пузырька. На рис. 4 приведен пример работы детектора пузырьков на поверхности металла. Идентификация пузырей прошла успешно, кроме граничных областей, для пузырьков, которые вошли не целиком, это связано с недоработкой алгоритма проведения окружности.

После того как операция детектирования была проведена на тестовом образце, было решено провести обработку всего набора кадров. При этом встал вопрос о скорости обработки, обработка одного изображения занимает около 40 с, таким образом для анализа одного эксперимента уходит порядка 2 ч. Так как таких экспериментов несколько десятков, то стал вопрос об использовании параллельных вычислений для данной задачи.

Параллельные вычисления в процессе работы детектора

Обоснованное решение проблемы отображения задач обработки изображений на архитектуру распределенных вычислительных систем с различными типами параллелизма должно опираться на исследование различ-

ных вариантов организации хранения и обработки данных применительно к архитектуре вычислительной системы и программному обеспечению.

Выделяются следующие основные подходы к организации программного обеспечения параллельной обработки изображений [4]: паралельная обработка на многоядерных процессорах в рамках модели общей памяти; паралельная обработка с использованием гетерогенных многопроцессорных вычислительных систем (CPU+GPU); параллельная обработка на распределенных вычислительных системах, в том числе кластерных.

Рассмотрим возможности к распараллеливанию данной задачи на уровне параллелизма данных.

Название модели «параллелизм данных» происходит оттого, что параллелизм заключается в применении одной и той же операции к множеству элементов данных. В нашем случае этой моделью удобно пользоваться для набора кадров. Для параллельной обработки кадров удобно использовать такую технологию программирования, как OpenMP. Естественный при обработке изображений параллелизм, основанный на декомпозиции данных, позволяет достаточно просто адаптировать последовательные реализации широкого круга алгоритмов обработки изображений для их выполнения на многоядерных системах. Однако с увеличением размеров входных данных время их обработки все более определяется временем, затрачиваемым на операции ввода и вывода.

Следующий уровень – это распараллеливание отдельных процедур и алгоритмов. Сюда можно отнести алгоритмы фильтрации изображения, обработки, деления объектов изображения по типам и т.д. На этом уровне абстракций удобно использовать такую технологию, как OpenCV в связке с GPU.

Для параллельной обработки GPU модуль OpenCV содержит в себе несколько уровней функциональности. На нижнем уровне находятся реализации служебных операций, таких как инициализация и управление GPU, работа с памятью, механизм асинхронных вызовов. Уровнем выше реализован широкий набор базовых функций обработки изображений: различные методы фильтрации, поиск максимума, аффинные преобразования, вычисление разности изображений в различных нормах и т.д. Список этих алгоритмов постоянно обновляется, в целом они призваны облегчить разработку алгоритмов компьютерного зрения самого высокого уровня.

В OpenCV контейнером для хранения данных (в том числе изображений) служит класс Mat. По аналогии в GPU-модуле реализован класс GpuMat, практически с той же функциональностью, но хранящий данные в видеопамяти. Библиотека OpenCV предоставляет удобные средства обмена информацией между GPU и CPU. Основную часть GPU-модуля составля-

ют функции, имеющие интерфейс, идентичный CPU части OpenCV, с тем лишь отличием, что они принимают на вход объект типа GpuMat. Программный интерфейс модуля (API) создан максимально близким к интерфейсу CPU части OpenCV, что облегчает перенос существующего кода на GPU. Однажды загрузив изображение на видеокарту, можно использовать различные функции обработки, которые будут производиться целиком на видеокарте.

На данный момент проводится расчет для серии экспериментов, с использованием приведенных технологий (рис. 5).

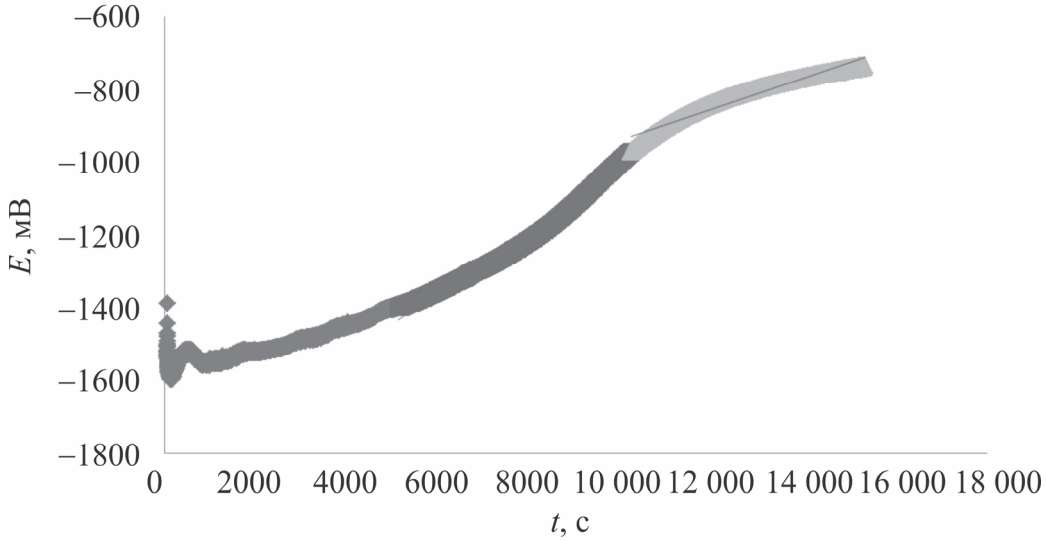


Рис. 5. Зависимость потенциала от времени
при проведения опыта на поверхности алюминия в 0,1 М растворе NaCl
под действием электрического поля

Результаты данных расчетов будут представлены на конференции и проанализированы с учетом изменения электрического поля во времени.

Список литературы

1. Szklarsa-Smialowska Z. Pitting corrosion of metals / NACE. – Houston, 1986.

2. Шапиро Л., Стокман Дж. Компьютерное зрение. – М.: Бином. Лаборатория знаний, 2006. – 752 с.

3. Otsu N. A threshold selection method from gray-level histograms // IEEE Trans. Sys., Man., Cyber. – 1979. – № 9. – P. 62–66.

Л.В. Еникеева[1], Н.Ф. Мурзашева[2], И.М. Губайдуллин[1]

[1]Институт нефтехимии и катализа РАН, Уфа
[2]Башкирский государственный университет, Уфа

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ ПРИ ИССЛЕДОВАНИИ КИНЕТИКИ РЕАКЦИИ ОКИСЛЕНИЯ 4-ТРЕТБУТИЛФЕНОЛА

Работа посвящена поиску кинетических параметров реакции окисления 4-третбутилфенола путем решения обратной кинетической задачи. В качестве алгоритма программы используется генетический алгоритм; для выполнения численных расчетов и анализа данных разработана программа на языке программирования Java с использованием пакета Octave. На данный момент ведется работа по распараллеливанию программ расчета с помощью готовых пакетов программы Octave.

Введение и постановка задачи

Данная работа продолжает исследования в области моделирования реакции окисления фенолов водными растворами пероксида водорода в присутствии различных между собой образцов металлосиликатов, проводимые в Институте нефтехимии и катализа РАН. Создание теории подбора катализаторов невозможно без всестороннего и детального изучения каталитических процессов, и прежде всего их кинетики. Исследование кинетики реакции позволяет дать наиболее строгую и точную оценку активности катализатора[1].

Авторами работы под руководством ст. науч. сотрудника лаборатории математической химии Института нефтехимии и катализа РАН, д-ра физ.-мат. наук И.М. Губайдуллина ведется разработка информационно-вычислительного электронного ресурса «Web-лаборатория математической химии» (http://mathchem.ru), который служит хранилищем созданных вычислительных модулей. Целью настоящей работы является описание подхода к оптимизации исследования кинетики реакции окисления 4-третбутилфенола водными растворами пероксида водорода, основанного на эволюционных вычислениях, и его распараллеливанию.

Основные структуры данных и генетический алгоритм

Генетический алгоритм основан на моделировании процесса естественного отбора в популяции особей, каждая из которых представлена точкой в

[1] Топчиева К., Романовский Б. Определение адсорбционных коэффициентов кинетическим методом. Адсорбционные коэффициенты воды, эфира и этилена на окиси алюминия // Кинетика и катализ. 1960. Т. 1, № 1. С. 233–238.

пространстве решений задачи оптимизации. Особи представлены структурами данных – Gen-хромосомами, включающими свободные (неопределенные) параметры *pk* обратной задачи химической кинетики S: Gen = {*P*}= {*p*1, *p*2, ..., *pk*}, $k \geq 0$. Эти параметры определяют предэкспоненциальные множители k_0 и энергии активаций Ea химических реакций S(Gen). Каждая популяция является множеством структур данных Gen и определяет множество параметров S(Gen).

Основная идея алгоритма синтеза состоит в эволюционном преобразовании множества хромосом (параметров обратной задачи химической кинетики) в процессе естественного отбора с целью выживания «сильнейшего». В нашем случае этими особями являются параметры, имеющие наименьшее значение целевой функции. Алгоритм начинается с генерации начальной популяции. Все особи этой популяции создаются случайно, затем отбираются наилучшие особи и запоминаются. Для создания популяции следующего поколения новые особи формируются с помощью генетических операций селекции (отбора), мутации, кроссовера и добавления новых элементов.

Целевой функцией служит минимальное значение среднеквадратичного отклонения расчетных кривых от экспериментальных данных:

$$\mathrm{F} = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(x_{ij}^{p} - x_{ij}^{э}\right)^2} \rightarrow \min,$$

где x_{ij}^{p} – расчетные значения концентраций наблюдаемых веществ, мольные доли; $x_{ij}^{э}$ – экспериментально полученные значения концентраций наблюдаемых веществ, мольные доли; N – количество точек эксперимента; M – количество наблюдаемых веществ реакции ($M = 2$).

Целью алгоритма является поиск параметров, при которых достигается минимум целевой функции F.

Распараллеливание генетического алгоритма и экспериментальные результаты

Для выполнения численных расчетов и анализа данных в Java выбрана программа Octave (http://www.gnu.org/software/octave/) – это высокоуровневый интерактивный язык программирования, предназначенный прежде всего для решения численных задач; возможна и пакетная (batch) работа. Как каждый проект организации GNU, программа Octave поставляется с полным исходным текстом и полностью бесплатна (условия GPL). Из расчетных модулей Java вызывается программа Octave, в которой выполняется решение прямой и обратной задач химической кинетики.

Одним из способов вызова внешнего скрипта Octave из Java-программы является использование стандартных потоков ввода-вывода для обработки передачи параметров и вывода. Тем не менее передача данных через стандартный поток ввода-вывода в Octave довольно медленная. Поэтому если размеры входных параметров и выходных результатов велики, то использование стандартных потоков ввода-вывода будет сильно замедлять работу программы Java.

На данный момент вычисления производятся в системе Octave и предполагают значительный объем вычислений.

Полученные нами данные при использовании генетического алгоритма приведены в табл. 1–3; δ – относительная погрешность отклонения, %.

Таблица 1

Расчетные данные для генетического алгоритма при температуре 35 °С

Номер эксперимента	Размер популяции	*F*	Время, с	*k*1	*k*2	*k*3	*k*4	δ
1	70	0,0073	1717	0,2601	0,0568	0,0004	2,8447	6,7794
2	140	0,0072	1404	0,2601	0,0568	0,0004	2,844	6,7794
3	300	0,0072	1697	0,2603	0,0575	0,0001	2,8670	6,6245

Таблица 2

Расчетные данные для генетического алгоритма при температуре 50 °С

Номер эксперимента	Размер популяции	*F*	Время, с	*k*1	*k*2	*k*3	*k*4	δ
1	70	0,0266	3217	1,2405	0,3156	0,0001	2,9460	8,7917
2	140	0,0264	23532	1,1765	0,3050	0,0000	2,7978	8,7385
3	300	0,0259	4899	0,9930	0,2723	0,0000	2,3643	8,6252

Таблица 3

Расчетные данные для генетического алгоритма при температуре 75 °С

Номер эксперимента	Размер популяции	*F*	Время, с	*k*1	*k*2	*k*3	*k*4	δ
1	70	0,0223	11074	1,3323	0,3643	0,0001	2,8604	9,9523
2	140	0,0223	4300	1,3372	0,3660	0,0005	2,8689	9,9681
3	300	0,0215	2843	0,9893	0,3009	0,0001	2,1288	9,6945

Основными решениями по организации распределенных вычислений в среде Octave являются библиотеки bcMPI[2], MPI Toolbox for Octave (MPITB)[3], octave-parallel_mpi[4] и parallel (ранее octave-parallel)[5], предназначенные для кластерных систем[6], и модуль parcellfun библиотеки general, предназначенный обеспечивать параллельные вычисления в масштабе «одной машины» (т.е. на многоядерных и предположительно на массивно-параллельных системах). Первые три библиотеки обеспечивают интерфейс к наличной инсталляции библиотеки MPI и, насколько возможно, транслируют в Octave (bcMPI – также и в MATLAB) синтаксис подмножества вызовов MPI. Соответственно, в этих случаях вычисления распределены лишь настолько, насколько это подготовлено программирующим, и обеспечиваются в рамках возможностей системной инсталляции MPI (рисунок).

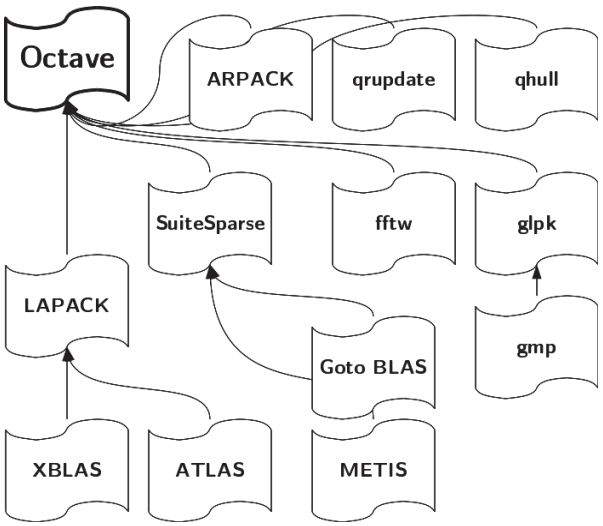


Рис. Схема основных зависимостей при сборке пакета Octave

Библиотека parallel, напротив, не требует наличия среды типа MPI или PVM, а обеспечивает распределенность с помощью внутренних средств пакета Octave.

На данный момент ведется работа по распараллеливанию программ расчета с помощью готовых пакетов программы Octave.

---

[2] Разработана в Центре супервычислений Огайо, г. Колумбус (США), см. http://www.osc.edu/bluecollarcomputing/applications/bcMPI/.

[3] Разработана в Университете Гранады (Испания), см. http://atc.ugr.es/~javier/mpitb.down.html.

[4] Разработана в Университете Тохоку, Япония, см. http://www.aoki.ecei.tohoku.ac.jp/octave/.

[5] Изначально разработана в Университете Тохоку (Япония), далее разрабатывается как часть проекта Octave Forge.

[6] Известно, например, что MPI Toolbox перенесена на платформу IBM BlueGene, см. http://www.pdc.kth.se/systems_support/computers/bluegene/bgloctave/mpitb-octave-on-bgl; http://www.ichec.ie/training_support/tutorials/bg_port.pdf.

А.В. Ермаков, И.К. Марчевский, Г.А. Щеглов

Московский государственный технический университет им. Н.Э. Баумана

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ВЫНУЖДЕННЫХ КОЛЕБАНИЙ БАЛКИ В ПРОСТРАНСТВЕННОМ ПОТОКЕ МЕТОДОМ ВИХРЕВЫХ ЭЛЕМЕНТОВ

Рассмотрен алгоритм численного моделирования вынужденных колебаний стержня, вызываемых его нестационарным пространственным обтеканием. Динамика стержня рассчитывается методом разложения по собственным формам; процесс вихреобразования моделируется методом вихревых элементов, программная реализация которого использует технологию распараллеливания MPI. Приведены результаты тестирования алгоритма на модельной задаче. Показано влияние податливости конструкции на лобовое сопротивление.

Введение

Стержень, взаимодействующий с потоком несжимаемой среды, является распространенной расчетной схемой, применяемой при математическом моделировании широкого класса технических систем. В случае протяженного жесткого стержня постоянного кругового сечения, помещенного в поперечный поток, гидродинамические нагрузки определяются сходом вихрей Кармана и близки к моногармоническим с безразмерной частотой $\mathrm{Sh} \approx 0,2$ [1]. Однако если рассматриваемый стержень является упругим, для правильного описания его вынужденных колебаний в потоке требуется решение сопряженной задачи гидроупругости. Данная задача рассматривалась многими авторами, однако на сегодняшний день прямое численное моделирование пространственного обтекания податливого стержня с использованием сеточных методов связано со значительными затратами вычислительных ресурсов.

В настоящей работе в качестве альтернативного подхода предлагается использовать в указанной задаче бессеточный лагранжев метод вихревых элементов. Данный подход, основанный на рассмотрении завихренности как первичной расчетной величины, в ряде случаев позволяет существенно снизить вычислительную трудоемкость задачи. При этом возможно рассмотрение пространственных течений в неограниченных областях.

Отличием предлагаемой модификации вихревого метода от известных панельных методов и метода дискретных вихрей [2] является отсутствие необходимости явного задания линии отрыва потока от поверхности обтекаемого тела за счет использования гипотезы потока завихренности [3].

К настоящему времени авторы имеют опыт применения различных модификаций вихревых методов для моделирования обтекания неподвижных жестких тел [4]. Целью настоящей работы является построение и тестирование алгоритма решения сопряженной задачи гидроупругости.

Математическая модель

Рассматривается прямолинейный стержень длиной L, имеющий постоянное поперечное сечение в форме кольца радиуса R с толщиной стенки h. Стержень имеет погонную массу γ, материал стержня изотропный с модулем упругости E и коэффициентом Пуассона ν. Концы стержня неподвижно закреплены и заглушены. Стержень помещен в поток несжимаемой среды постоянной плотности ρ, имеющей скорость $\vec{V}_\infty$ (рис. 1).

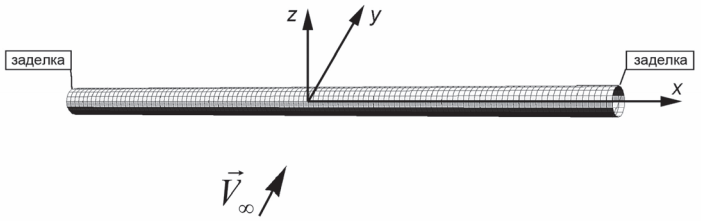


Рис. 1. Расчетная схема стержня в поперечном потоке

Вязкость среды μ считается малой, что позволяет применить подход Прандтля и рассматривать влияние вязкости только как причину генерации завихренности на обтекаемой поверхности. Внешнее течение считается идеальным.

Математическая модель сопряженной задачи гидроупругости имеет вид:

$$EJ\frac{\partial^4\vec{u}}{\partial x^4}+\gamma\frac{\partial^2\vec{u}}{\partial t^4}+\kappa J\frac{\partial^5\vec{u}}{\partial x^4\partial t}=\vec{Q}_{\text{н}}[p],$$

$$\nabla\cdot\vec{V}=0,$$

$$\frac{\partial\vec{V}}{\partial t}+(\vec{V}\cdot\nabla)\vec{V}=\frac{\mu}{\rho}\nabla^2\vec{V}-\nabla\left(\frac{p}{\rho}\right),$$

где J – момент инерции; $\vec{u}$ – вектор перемещения профиля в сечении стержня; $\vec{V}$ – поле скоростей жидкости; p – давление. Граничные условия соответствуют заделке концов стержня, прилипанию жидкости к обтекаемой поверхности $\vec{r}_{\text{к}}=\vec{r}_{0\text{к}}+\vec{u}(x,t)$ и затуханию возмущений на бесконечности:

$$\vec{u}\big|_{x=0,L}=\left.\frac{\partial\vec{u}}{\partial x}\right|_{x=0,L}=0,\ \vec{V}(\vec{r}_{\text{к}},t)=\dot{\vec{r}}_{\text{к}},\ \vec{V}\underset{r\to\infty}{\to}\vec{V}_\infty,\ p\underset{r\to\infty}{\to}p_\infty.$$

Начальные условия соответствуют равновесному положению стержня в покоящейся среде.

Таким образом, сопряжение упругой и гидродинамической подсистем осуществляется за счет удовлетворения граничных условий на подвижной границе $\vec{r}_{\text{к}}(x,t)$, перемещение которой определяется действующими гидродинамическими нагрузками.

Численный метод

Для моделирования гидродинамической подсистемы применяется метод вихревых элементов [3]. Поле завихренности моделируется симметричными вортонами-отрезками [4] и замкнутыми вортонными рамками на поверхности тела, при этом поле скоростей рассчитывается с использованием закона Био – Савара. Для восстановления давления используется аналог интеграла Бернулли и Коши – Лагранжа [5]. Выполнение граничного условия прилипания обеспечивается генерацией вортонов на всей обтекаемой поверхности; данные вортоны формируют вихревой след за телом.

Алгоритм метода вихревых элементов допускает эффективное распараллеливание: в разработанном программном комплексе, основанном на использовании технологии MPI, доля параллельного кода доведена до 98 %, что позволяет производить расчеты на вычислительных кластерах, содержащих десятки вычислительных ядер [6, 7].

Для моделирования упругой подсистемы используется метод разложения по собственным формам колебаний; при этом на шаге интегрирования гидродинамическая нагрузка принимается постоянной, что позволяет использовать аналитическое решение уравнения малых колебаний стержня:

$$\{\ddot{\varphi}\} + \delta[\omega]_{\text{diag}}\{\dot{\varphi}\} + [\omega^2]_{\text{diag}}\{\varphi\} = \{f_H\},$$

где $\{\varphi\}$ – вектор главных нормальных координат; δ – коэффициент модального демпфирования; $[\omega]_{\text{diag}}$ – диагональная матрица собственных частот; $\{f_{\text{н}}\}$ – постоянный на данном шаге интегрирования вектор нагрузки, приведенный к главным координатам.

Собственные частоты и формы колебаний определяются методом конечных элементов с помощью пакета MSC.Nastran, с использованием решателя SOL103. Сетка на поверхности стержня, используемая как для конечно-элементного расчета, так и для формирования вортонных рамок, строится в препроцессоре Patran. Вектор перемещения узлов сетки на поверхности $\{q\}$ связан с вектором главных координат $\{\varphi\}$ при помощи матрицы собственных форм: $\{q\} = [A]\{\varphi\}$.

Тестирование алгоритмов производилось путем сравнения с известными экспериментальными результатами М.Б. Гафурова, М.А. Ильгамова [8].

Вычислительный эксперимент

В качестве модельной задачи рассматривается стержень с параметрами $L = 1{,}1$ м, $R = 0{,}025$ м, $h = 0{,}001$ м, изготовленный из изотропного материала ($E = 2 \cdot 10^9$ Па, $\nu = 0{,}3$, $\gamma = 1{,}1$ кг/м). Коэффициент модального демпфирования принимался $\delta = 0{,}05$. Конечно-элементная модель содержала 1792 элемента типа QUAD4, что в методе вихревых элементов соответствовало 16 панелям в окружном направлении, 110 панелям вдоль образующей и 32 панелям на торцах. Скорость набегающего потока принималась равной 1 м/с.

При разложении по собственным формам использовались 6 низших тонов колебаний. В силу симметрии стержня соответствующие им собственные частоты попарно совпадают:

$$\omega_{1,2} = 26{,}6 \text{ Гц}, \ \omega_{3,4} = 70{,}8 \text{ Гц}, \ \omega_{5,6} = 132{,}7 \text{ Гц}.$$

Был проведен расчет переходного режима от момента времени $t = 0$ до $t = 1$ с с шагом $\Delta t = 0{,}001$ с. Вихревой след за стержнем в момент окончания расчета представлен на рис. 2, где точками обозначены положения вихревых элементов.

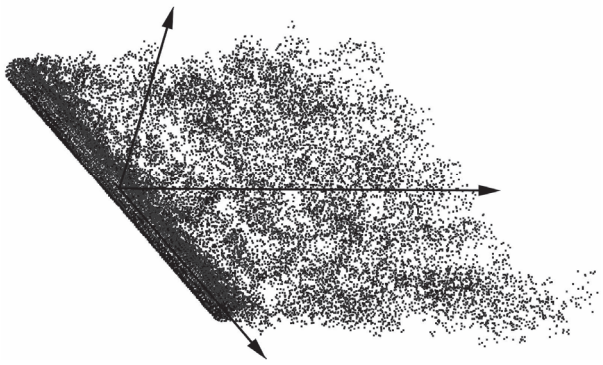


Рис. 2. Вихревой след за стержнем в момент времени $t = 1$ с

В среднем сечении след за стержнем имеет характерный вид вихревой дорожки Кармана (рис. 3).

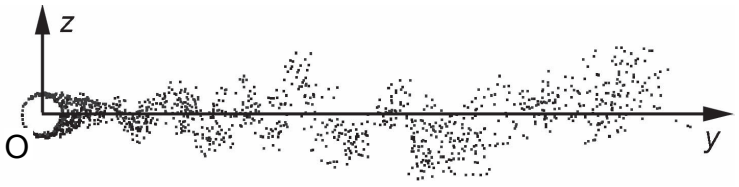


Рис. 3. Дорожка Кармана за стержнем в сечении *Oxz*

В ходе численного моделирования установлено, что под действием набегающего потока стержень в плоскости *Oxy* изгибается в основном по первой собственной форме (рис. 4), тогда как в плоскости *Oxz* проявляется вторая собственная форма. В результате движение сечений имеет сложную траекторию (рис. 5).

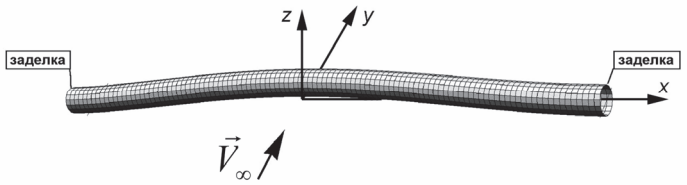


Рис. 4. Деформированная форма стержня в момент времени $t = 0{,}1$ с
(перемещения увеличены в 30 раз)

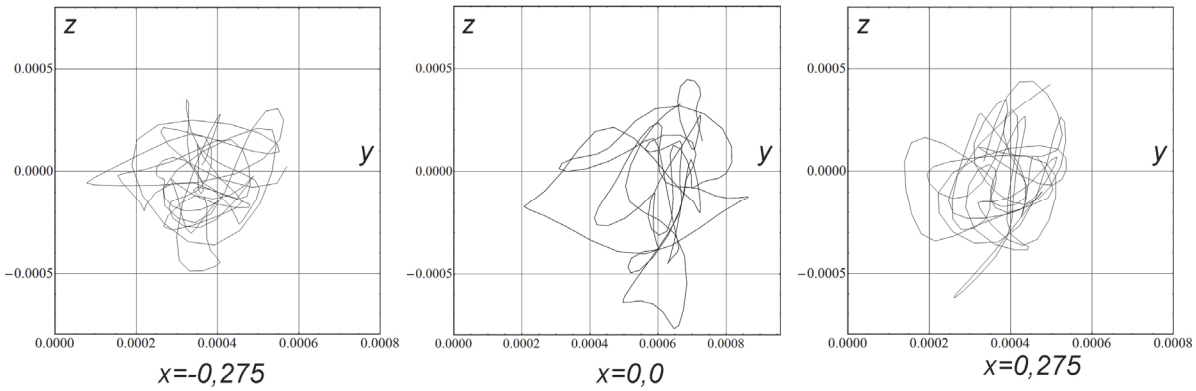


Рис. 5. Траектории центров сечений стержня при $0{,}8 < t < 1{,}0$,
соответствующих пучностям 1-й и 2-й собственных форм

Анализ спектра поперечных перемещений сечений показал, что после окончания переходного режима (при $t > 0{,}3$ с) устанавливается режим вынужденных колебаний, в котором преобладают частоты, близкие ко 2-й собственной частоте и частоте схода вихрей Кармана. На рис. 6 показан спектр перемещений среднего сечения, в котором частоты приведены к числам Струхаля.

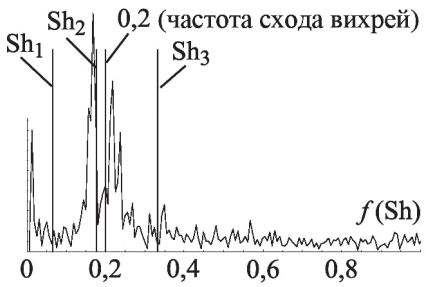


Рис. 6. Спектр перемещений центра
среднего сечения стержня в направлении оси *Oz*

Также установлено, что вынужденные колебания стержня приводят к некоторому снижению лобового сопротивления. На рис. 7 показаны зависимости силы лобового сопротивления для жесткой (rigid) и упругой (elastic) моделей.

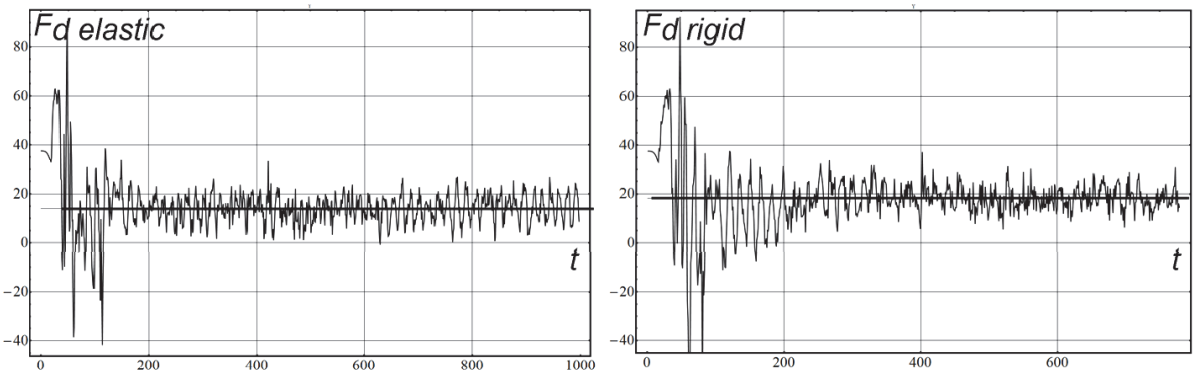


Рис. 7. Лобовое сопротивление упругого и абсолютно жесткого стержней

Среднее значение силы лобового сопротивления, показанное на рис. 7 сплошной линией, снижается при этом примерно на 15–20 %.

Заключение

Разработанный алгоритм позволяет исследовать взаимодействие элементов конструкций с набегающим потоком, при этом форма и упругая модель могут быть достаточно сложными. Для определения собственных частот и форм колебаний используется широко распространенный комплекс пакетов Nastran-Patran, а для расчета течения жидкости – метод вихревых элементов.

Авторы благодарят Центр компетенций компании MSC Software при МГТУ им. Н.Э. Баумана. Работа выполнена при частичной финансовой поддержке гранта Президента РФ (проект МК-3705.2014.8).

Список литературы

1. Жукаускас А., Улинскас Р., Катинас В. Гидродинамика и вибрации обтекаемых пучков труб. – Вильнюс: Мокслас, 1984. – 312 с.

2. Нелинейная теория крыла и ее приложения / Т.О. Аубакиров, С.М. Белоцерковский, А.И. Желанников, М.И. Ништ. – Алматы: Гылым, 1997. – 448 с.

3. Cottet G.-H., Koumoutsakos P.D. Vortex Methods: Theory and Practice. CUP, 2008.

4. Модель симметричного вортона-отрезка для численного моделирования пространственных течений идеальной несжимаемой среды / И.К. Марчевский, Г.А. Щеглов, И.К. Марчевский, Г.А. Щеглов // Вестник МГТУ им. Н.Э. Баумана. Сер. Естественные науки. – 2008. – № 4. – С. 62–71.

5. Андронов П.Р., Гувернюк С.В., Дынникова Г.Я. Вихревые методы расчета нестационарных гидродинамических нагрузок. – М.: Изд-во МГУ, 2006. – 184 с.

6. Марчевский И.К., Щеглов Г.А. Параллельная реализация метода вихревых элементов с использованием модели симметричного вортона-отрезка // ПаВТ'2013: тр. междунар. науч. конф. – Челябинск, 2013. – С. 203–214.

7. Марчевский И.К., Щеглов Г.А. Применение параллельных алгоритмов при решении задач гидродинамики методом вихревых элементов // Вычислительные технологии и программирование. – 2010. – Т. 11. – С. 105–110.

8. Ермаков А.В., Щеглов Г.А. Моделирование методом вихревых элементов динамики цилиндрической оболочки в пространственном потоке жидкости // Изв. вузов. Машиностроение. – 2014. – № 3. – С. 35–41.

Е.Б. Замятина[1], А.А. Митраков[2]

[1]Национальный исследовательский университет
«Высшая школа экономики» (Пермский филиал),
[2]Пермский государственный национальный исследовательский университет

АЛГОРИТМЫ СИНХРОНИЗАЦИИ, ОСНОВАННЫЕ НА ЗНАНИЯХ О МОДЕЛИ, В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ ИМИТАЦИИ

Рассматриваются алгоритмы синхронизации в распределенных системах имитации и способы их оптимизации. Приводятся описания двух типов систем имитации: традиционной (событийно-ориентированная парадигма) Triad.Net и агентно-ориентированной. Предлагается использовать метод оптимизации алгоритмов синхронизации, основанный на знаниях. Приводятся результаты, подтверждающие важность использования знаний о конкретной имитационной модели для оптимизации алгоритмов синхронизации.

Введение

Метод имитационного моделирования широко применяется для исследования сложных динамических систем, каковыми являются, в частности, компьютерные сети, параллельные и распределенные вычислительные системы (РВС), к которым можно отнести сетевые организации, виртуальные предприятия, социальные сети и т.д. Еще одной интересной задачей имитационного моделирования является исследование масштабируемых алгоритмов, выполняющихся на базе параллелльных и распределенных

систем. Исследователи, применяя метод имитационного моделирования, пытаются определить адекватность поведения параллельных и распределенных алгоритмов при значительном изменении количества вычислительных узлов (проиграть их функционирование в вычислительной среде, которой пока не существует).

Очень часто количество вычислительных узлов в РВС таково, что имитационный эксперимент является затратным по времени. По этой причине возникает необходимость в сокращении времени выполнения имитационного эксперимента. Одним из решений этой проблемы является использование высокопроизводительной техники. Система имитации в этом случае использует ресурсы нескольких вычислительных узлов для прогона распределенной имитационной модели. Распределенная (параллельная) имитационная модель представляет собой совокупность логических процессов, выполняющихся на различных вычислительных узлах и обменивающихся сообщениями друг с другом. Управляют функционированием распределенной имитационной модели алгоритмы синхронизации. Для того чтобы распределенный имитационный эксперимент был эффективным, необходимо оптимизировать алгоритм синхронизации. Вторым источником оптимизации является балансировка логических процессов распределенной имитационной системы. В системе моделирования Triad.Net реализована мультиагентная система балансировки, описание и результаты можно посмотреть в [1].

Два класса алгоритмов синхронизации и способы их оптимизации

Алгоритмы синхронизации логических процессов распределенной имитационной модели обычно разделяют на два класса: консервативные и оптимистические [2].

Консервативные алгоритмы обычно «сдерживают» продвижение системного времени, очередное событие выполняется, если можно убедиться, что другой логический процесс не передаст текущему сообщение, помеченное меньшим штампом времени, чем рассматриваемое (текущее) событие. Для того чтобы расширить временные горизонты процессов и не попасть в тупиковую ситуацию, каждый логический процесс по всем выходным связям при изменении своего времени посылает сообщение – либо реальное, запрограммированное в модели, либо фиктивное или «нулевое».

Классическим оптимистическим алгоритмом является алгоритм Time Warp [3]. Алгоритм позволяет обрабатывать любые события (не только безопасные), но при поступлении сообщения с временной меткой T’ (используют еще термин «штамп времени»), которое меньше текущего времени T, происходит откат на момент времени T’. Для реализации отката

необходимо восстановить состояние текущего логического процесса на момент времени T’. Кроме того, необходимо отменить сообщения, разосланные другим логическим процессам. Для этого используют механизм антисообщений: для каждого обработанного сообщения хранится противоположное ему. Для отмены сообщения посылают «антикопию». При получении антисообщения процесс либо удаляет положительное сообщение из очереди, если оно не было еще обработано, либо производит откат до момента времени, которым помечено антисообщение.

Два подхода к организации распределенных систем имитационного моделирования

В настоящее время создаваемые распределенные системы имитации разделяются на два класса: традиционные, соответствующие основным парадигмам имитационного моделирования: процессо-ориентированные, событийно-ориентированные, объектно-ориентированные; и агентно-ориентированнные. Если проектирование имитационной модели в первом случае ведется сверху вниз, то в агентно-ориентированных моделях взаимодействующие агенты совместно отображают реальносуществующую систему, ситуацию, процесс (проектирование снизу вверх). Агентно-ориентированные системы изначально представляют собой распределенную систему.

Распределенная система моделирования Triad.Net и алгоритм синхронизации TriadRule

Распределенная система имитации Triad.Net [4] была разработана на базе системы автоматизированного проектирования и моделирования Triad [5] и предназначалась для разработки и исследования вычислительных систем. Система Triad имеет такие отличительные характеристики, как трехслойное представление модели, включающее слой структур, слой рутин и слой сообщений. Описание структуры вычислительной системы, обычно представлено графом

$$P = (V, W, E),$$

где V – множество вершин, соответствующих объектам моделируемой системы; W – множество входных и выходных полюсов; E – множество дуг, соединяющих эти вершины.

Каждая вершина может быть описана составляющими ее компонентами (графом более низкого уровня). Слой рутин состоит из последовательности событий, планирующих друг друга. Эта последовательность событий описывает поведение каждой из вершин. Слой сообщений необхо-

дим для описания сообщений сложной структуры. В качестве сообщения может быть рассмотрена и описана программа, которая выполняется на описанной модели вычислительной системы. Описание структуры, рутин, сообщения представляет собой параметризованные процедуры. Управляющая программа (симулятор) продвигает время от события к событию.

В распределенной модели отдельные вершины *P*-графа могут располагаться на различных вычислительных узлах. Во время имитационного эксперимента распределенная модель представляет собой совокупность логических процессов, описанных рутинами, которые обмениваются сообщениями друг с другом. Для продвижения времени в распределенной системе имитации реализован оптимистический алгоритм Time Warp. Time Warp не ограничивает выполнение логических процессов, что приводит к большому количеству откатов, которые сигнализируют о значительных тратах вычислительных ресурсов впустую. Модельное время из-за откатов также продвигается медленно, поэтому возникает необходимость в оптимизации алгоритма. Существует ряд работ [2], в которых используются знания об имитационной модели для сокращения времени выполнения распределенного имитационного эксперимента. Примерами являются классические алгоритмы: lookahead используют в консервативных алгоритмах, lookback – в оптимистических, циклы – в алгоритме Work Flow и т.д.

При реализации алгоритма синхронизации в Triad.Net [6] также были использованы знания о модели. Проблема алгоритма состоит в том, что необходимо на каждом шаге определить, является ли обработка следующего локального события безопасной, т.е. можно ли гарантировать отсутствие отката. Анализ текущей последовательности событий (в рамках одного прогона) или результаты предыдущих выполнений системы, а также знания эксперта позволяют выявить зависимости между событиями и, следовательно, определить безопасное событие. Наиболее адекватным решением является применение продукционной экспертной системы. Правила определяются не только знаниями эксперта о конкретной модели, но и генерируются автоматически: в ходе статического анализа кода модели перед выполнением, а также в ходе выполнения: при откатах необходимо создавать правила, предотвращающие появление откатов в будущем. Например, если мы обработали событие покупки билета вторым клиентом (E2), а затем произошел откат из-за события покупки билета первым клиентом (E1), в системе генерируется связь (E1→E2).

Моделирование проводилось на кластере с использованием нескольких простых моделей. Для сравнения производительности были выбраны классический консервативный и оптимистический алгоритмы. Наиболее показательные результаты были получены на модели вычислителя IllIAC (рис. 1, 2).

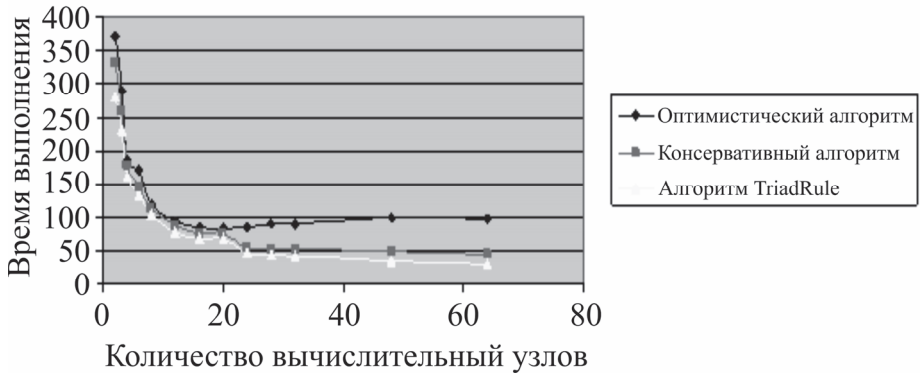


Рис. 1. Время выполнения имитационного эксперимента
в зависимости от количества вычислительных узлов

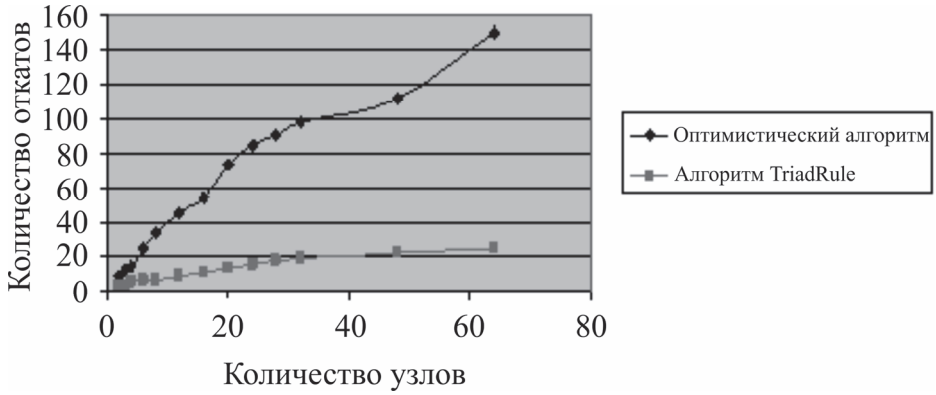


Рис. 2. Сравнение количества откатов

Алгоритм TriadRule показывает лучшую производительность, чем оба традиционных алгоритма, причем при увеличении количества вычислительных узлов разница в производительности становится более заметна. Происходит это потому, что увеличивается количество логических процессов, а следовательно, и общее количество внешних связей в системе. Для консервативного алгоритма это приводит к уменьшению быстродействия, так как временной горизонт продвижения становится меньше, а для оптимистического – приводит к увеличению количества откатов.

Итак, использование знаний о конкретной модели позволило улучшить характеристики алгоритма синхронизации в традиционных системах имитационного моделирования. Посмотрим, можно ли применить аналогичный подход в мультиагентных системах имитационного моделирования.

Мультиагентная система имитационного моделирования и алгоритм синхронизации, основанный на знаниях о модели

Настоящая версия мультиагентной системы моделирования выполнена на языке Scala [7]. Язык Scala был выбран в качестве языка программирования для реализации агентной платформы моделирования. Язык Scala пред-

ставляет собой кросспарадигменный объектно-функциональный язык программирования, совмещающий в себе объектно-ориентированное программирование и функциональное программирование и специализирующийся на создании легкомасштабируемого компонентного программного обеспечения (Scala была создана в 2004 г. под руководством Мартина Одерски в Университете EPFL (Lausanne, Switzerland)). Среди ключевых особенностей языка можно отметить: единую объектную модель, наличие примесей (traits), технику сопоставления с образцом, λ-исчисление, виды (type views bounds), линеаризацию типов (type linearization), параметрический и функциональный полиморфизм, вариантность типов, кейс-классы (case classes), вывод типов, поддержка хвостовой рекурсии, наличие многочисленных инструментов по созданию новых языковых конструкций и обработку списков. Обобщенная структурная схема симулятора изображена на рис. 3.

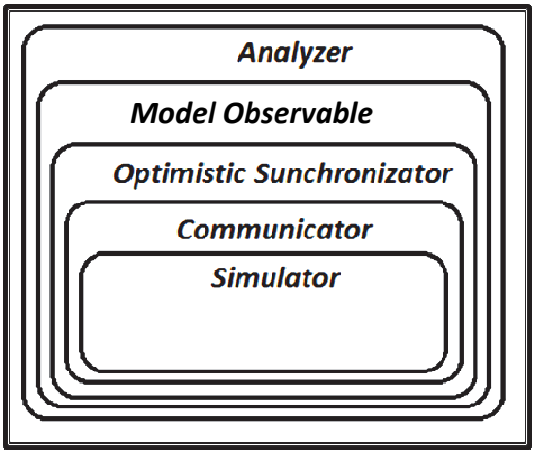


Рис. 3. Многослойная архитектура симулятора

Таким образом, архитектура симулятора является не многокомпонентной, а многослойной. Достигается это за счет того, что к модулю, реализующему некоторую базовую функциональность, примешиваются другие компоненты, расширяющие структуру, поведение и семантику. Трейт Simulator является базовым модулем, к которому подмешиваются другие элементы, уточняя его поведение. Фактически, Simulator, – это каркас для логического процесса. Трейт Communicator представляет собой модуль для работы с акторской системой. Он содержит в себе экземпляр актора, который занимается посылкой/приемом сообщений. Трейт Optimistic Synchronizator реализует оптимистический алгоритм синхронизации Time Warp. Трейт Model Observable является реализацией концепции информационных процедур, принятых в области имитационного моделирования для организации сбора статистики (информационные процедуры – программные компоненты, накладываемые на модель с целью сбора статисти-

ки в ходе имитационного прогона). Трейт Analyzer добавляет «интеллектуальность» алгоритмам синхронизации, используемых для реализации распределенного моделирования.

В системе реализован оптимистический алгоритм синхронизации, основанный на знаниях об имитационной модели. Использование знаний об агентной имитационной модели, извлекаемые из онтологий, дали хорошие результаты при реализации алгоритмов синхронизации в агентной системе имитации [8]. Были проведены эксперименты с одной и той же моделью многократно с одними и теми же параметрами. Общая длительность моделирования составляла (300 ± 15) единиц модельного времени.

Эксперименты показали, что метрики, характеризующие скорость проведения распределенного эксперимента, управляемого модифицированным алгоритмом синхронизации (алгоритм KBASA, основанный на знаниях о модели), существенно сократились по сравнению с метриками классического оптимистического алгоритма Time Warp. Действительно, количество откатов сократилось с 71,2 до 3,8.

Общее число неглубоких откатов сократилось практически до нуля (в среднем 0,1–0,2 против 7–11). Число глубоких откатов также сократилось с 19,4 до 3,2. Количество отправленных антисообщений сократилось с 108,4 до 12,8.

Заключение

Итак, был рассмотрен подход, который использует знания о модели для сокращения времени выполнения имитационного эксперимента для распределенной как традиционной, так и агентной имитационной модели. Очень часто при оптимизации алгоритмов синхронизации разработчики стремятся к универсализации алгоритма, а результаты исследований, приведенные выше, показали, что целесообразно еще и извлекать знания о конкретной модели. Этот же подход целесообразно применять при проведении балансировки вычислительных узлов, на которых проводится распределенный имитационный эксперимент.

Работа выполнена при финансовой поддержке гранта РФФИ № 12-07-00302-а, гранта РФФИ № 13-07-96506_юг_а.

Список литературы

1. Миков А.И., Замятина Е.Б., Козлов А.А. Мультиагентный подход к решению проблемы равномерного распределения вычислительной нагрузки // Natural and Artificial Intelligence, ITHEA. – Sofia, Bulgaria, 2010. – P. 173–180.

2. Fujimoto R.M. Distributed Simulation Systems / S. Chick, P.J. Sánchez, D. Ferrin, and D.J. Morrice, eds. The 2003 Winter Simulation Conference 7–10 December 2003. The Fairmont New Orleans. – New Orleans, LA. – P. 124–134.

3. Jefferson D.R. Virtual Time // ACM Transactions on Programming Languages and Systems. – 1985. – № 7 (3). – P. 404–425.

4. Замятина Е.Б., Миков А.И. Программные средства системы имитации Triad.Net для обеспечения ее адаптируемости и открытости // Информатизация и связь. – 2012. – № 5. – С. 130–133.

5. Mikov A.I. Simulation and Design of Hardware and Software with Triad// Proc.2nd Intl.Conf. on Electronic Hardware Description Languages. – Las Vegas, USA, 1995. – P. 15–20.

6. Замятина Е., Ермаков С. Алгоритм синхронизации объктов распределенной имитационной модели в TRIAD.Net // Applicable Information Models. ITHEA, Sofia, Bulgaria, 2011. – P. 211–220.

7. Odersky M. The Scala Programming Language [Электронный ресурс]. – URL: http://www.scala-lang.org/node/25 (дата обращения: 06.06.2013).

8. Замятина Е.Б., Каримов Д.Ф., Митраков А.А. Архитектура агентно-ориентированной системы имитации с агентами, основанными на нейронных сетях // Информатизация и связь. – 2014. – № 2. – С. 89–97.

А.В. Захарчук, П.А. Смирнов, Д.А. Насонов, С.В. Ковальчук

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

ДИНАМИЧЕСКИЙ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫЙ ЯЗЫК ДЛЯ ЗАДАЧ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ В СРЕДАХ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Данная работа представляет инструмент для высокоуровневого описания BigData задач в сфере e-Science, в основе которого находится динамический предметно-ориентированный язык программирования. Динамическую структуру языка определяет наличие предметных библиотек, подключаемых в зависимости от решаемой задачи. Такие библиотеки описывают подходы к обработке, анализу и моделированию данных из конкретной предметной области. Разрабатываемая технология интегрирована в состав облачной платформы CLAVIRE и совместно с технологией Virtual Simulation Objects используется для разработки и сервиса workflow в распределенной облачной среде, сочетая в себе решение вычислительно трудоемких задач с технологией обработки данных MapReduce.

Введение

Современные задачи в области e-Science для своего решения требуют приложений, оперирующих с большими объемами данных и входящих в состав композитных приложений [1]. В настоящее время огромное количество данных доступно для научных исследований, например, полученные с помощью моделирования, от различных датчиков, собранные во время наблюдений, при краулинге сетей и т.д. Рост количества и разнообразия таких данных совместно с увеличивающимися вычислительными мощностями является одной из причин появления области BigData [2]. Для этой области уже создано множество решений, включающих в себя модели, методы и технологии для более эффективного управления большими массивами данных. Например, модель MapReduce [3] сегодня является практически стандартом для обработки распределенных данных. Но некоторые вопросы все же остаются открытыми до сих пор. Один из них – разработка высокоуровневого, простого в использовании средства описания задач [4]. Сейчас для описания комплексных предметно-ориентированных e-Science задач часто используется модель workflow (WF) [5]. Но такой метод чаще всего подразумевает под собой подход data-to-code (данные передаются к исполнителю), в то время как BigData основан на подходе code-to-data (исполнитель передается к данным и выполняется локально) [6]. Это различие делает довольно сложным интеграцию BigData в системы управления workflow, и зачастую модуль, отвечающий за BigData, является отдельным специальным сервисом workflow, что нарушает целостность архитектуры [7]. С другой стороны, разработка workflow требует определенных технических знаний, которыми зачастую специалисты предметных областей, являющиеся конечными пользователями таких систем, не обладают [8], поэтому есть необходимость в более дружественном инструменте, не требующем глубоких знаний программирования.

В данной работе описывается технология, позволяющая конечным пользователям (специалистам предметных областей) создавать высокоуровневое описание решаемых задач, использующая оба подхода – code-to-data и data-to-code – при поддержке формализованных знаний предметной области [9]. В основе такой технологии лежит разработанный динамический предметно-ориентированный язык (DSL), который может адаптироваться к различным предметным областям и быть интегрированным в состав облачной платформы для описания BigData-задач. В работе представлены главные особенности разработанного языка и его применение в составе композитных приложений.

Детали реализации

Разработанное решение основывается на облачной вычислительной среде CLAVIRE [10], которая использует технологию базы знаний для решения научных задач. CLAVIRE поддерживает набор DSL: для описания доступных прикладных пакетов (EasyPackage, реализованный на базе языка Ruby), и для высокоуровневого описания композитных приложений в форме абстрактных workflow (EasyFlow, реализованный с использованием технологии ANTLR). Кроме того, используется технология Virtual Simulation Objects (VSO) для поддержки высокоуровневого семантического описания исследуемой системы. И с ее помощью модель системы может быть отображена на структуру абстрактного WF и выполнена в облачной среде [11]. Платформа использует распределенное хранилище данных (dStorage), которое было расширено до поддержки модели MapReduce [12]. В данной работе был разработан динамический DSL, который служит в качестве инструмента для высокоуровневого описания задач по анализу данных и использует вышеописанные технологии. Цель разработки состоит в расширении возможностей CLAVIRE в создании гибридных композитных приложений с помощью подходов code-to-data и data-to-code. Язык был разработан с использованием платформы Java, что позволяет легко интегрировать его с dStorage, также написанном на Java, и технологии ANTLR, для более простой совместимости с EasyFlow.

Распределенная обработка данных платформой CLAVIRE

Текущая версия языка была разработана для платформы CLAVIRE, и в частности для его распределенного хранилища dStorage. Архитектура распределенной обработки данных в CLAVIRE и dStorage представлена на рис. 1.

Обработка данных внутри хранилища производится по фиксированной схеме MapReduce. Сначала для каждого файла, хранящегося в хранилище, применяется функция Map, затем ко всем результатам на одном агенте применяется функция Reduce, и к этим результатам агент, выбранный главным, применяет снова функцию Reduce. Результат последнего шага является результатом выполнения задачи. Функции Map и Reduce описываются внутри обработчика (Data processor). Обработчик написан с помощью языка программирования Java и является .jar файлом. Он загружается в хранилище и рассылается всем агентам, на которых хранятся обрабатываемые файлы.

Для реализации высокоуровневого описания различных задач был разработан интерпретатор динамического DSL, который является составной частью обработчика данных. Помимо интерпретатора, в зависимости от задачи,

к обработчику подключаются необходимые библиотеки предметной области. В итоге задача описывается в виде скрипта на DSL и передается как параметр при запуске обработчика. Функции Map и Reduce при помощи ANTLR разбирают скрипт и строят абстрактное синтаксическое дерево (AST), при обходе которого выполняют все необходимые функции.

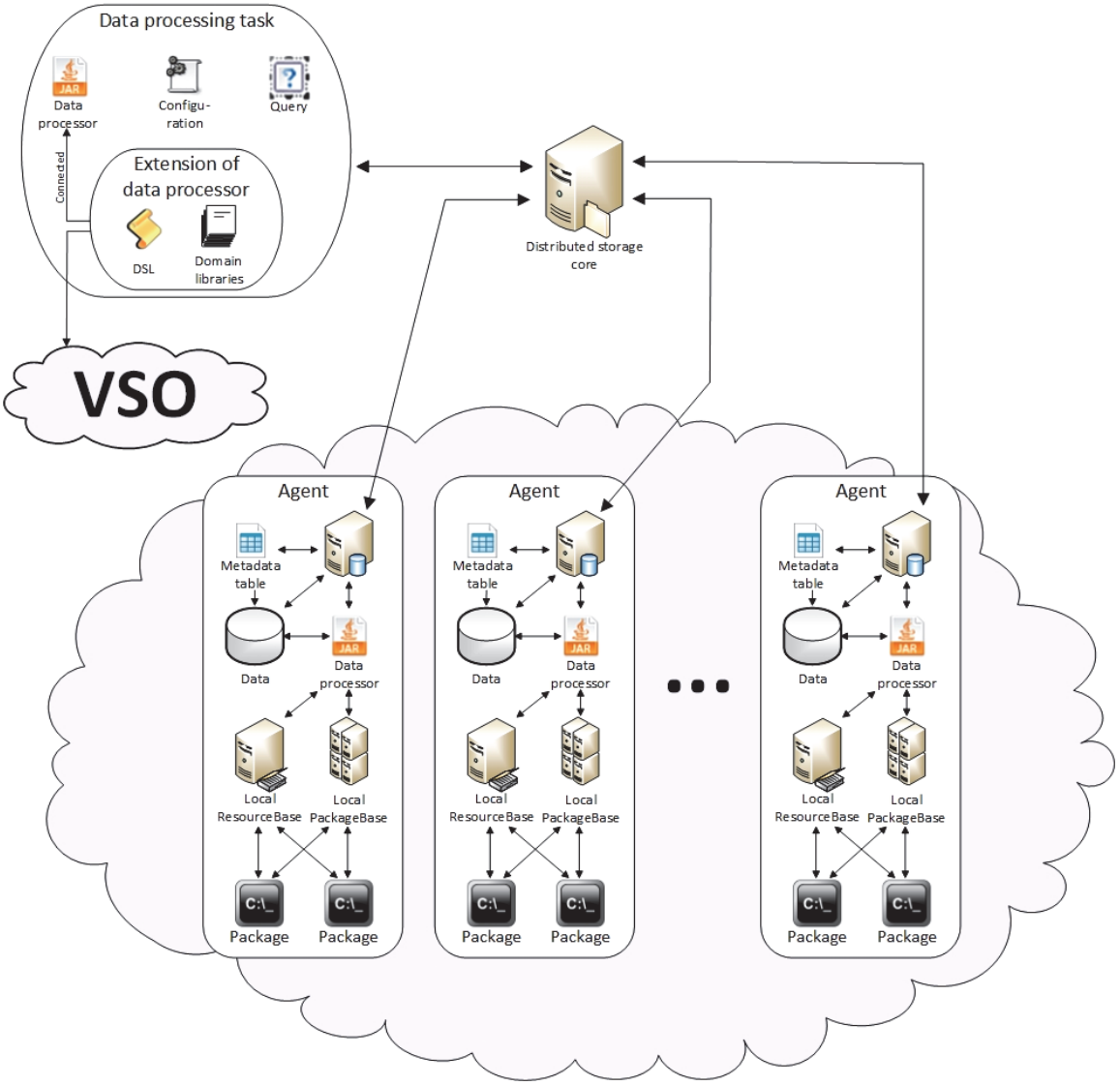


Рис. 1. Распределенная обработка данных в платформе CLAVIRE

Для поддержки высокоуровневого семантического описания задач используется технология VSO, являющаяся расширением платформы CLAVIRE. Технология VSO обеспечивает доступ к формализованным знаниям предметной области, содержащим высокоуровневые семантические абстракции, позволяющие формировать описание исследуемой системы в терминах предметной области. Также интеграция с CLAVIRE позволяет использовать абстрактные WF в виде EasyFlow и доступные прикладные пакеты с помощью EasyPackage.

Интерпретаторы и функции языка

Чтобы конечным пользователям не приходилось явно описывать функции Map и Reduce, а описать только задачу в общем виде, используя предметные примитивы, и не задумываться о том, что выполнение будет производиться в распределенном хранилище, было решено создать два интерпретатора. Один из них должен интерпретировать запрос для выполнения функции Map, второй – для функции Reduce.

Интерпретаторы грамматики основных функций языка выполняют следующие действия:

select : SELECT domainName (whereCond)? ;

create : CREATE domainName FROM domainNameFrom (whereCond)?;

simulate

: SIMULATE PACKAGE packageName WITH domainEntityName (LSQUARE auxiliaries RSQUARE)? execParam

| SIMULATE VSO vsoName FROM domainEntityNameFrom (LSQUARE auxiliaries RSQUARE)? TO domainEntityNameTo;

out : OUT name ((DOT functionList)? | (DOT variable)?);

1. Select. В функции Map создает объект из библиотеки предметной области и, используя унифицированные алгоритмы чтения, наполняет его данными из файлов хранилища. В Reduce используется для объединения разных частей одного data-сета. Если объект полностью собран и прошел проверку на целостность, то проверяются условия, описанные в виде вызова функций анализа из библиотеки, по отбору необходимых данных. Использует описание объектов и функции, применяемые к ним, из предметных библиотек.

2. Create. Аналогично функции select, но данные для наполнения новых объектов берутся не из файлов, а из уже существующих объектов, например считанных ранее из хранилища.

3. Simulate. При выполнении задачи может потребоваться необходимость вызова прикладного пакета либо исполнения абстрактного WF, именно это и делает данная функция.

3.1. Если во время выполнения запроса появляется необходимость выполнить прикладной пакет, то интерпретатор обращается к сервису PackageResource для получения всех необходимых сведений об этом пакете. Затем делает локальный вызов прикладного пакета с параметрами, введенными пользователем, тем самым обеспечивая подход code-to-data.

3.2. При необходимости использования VSO интерпретатор делает вызов сервиса VSO, указывая название сценария, какие параметры имеются, и какой параметр необходимо получить. Сервис строит абстрактный

WF, при исполнении которого все прикладные пакеты будут обработаны локально, в рамках подхода code-to-data.

В связи с тем, что многие прикладные пакеты на вход требуют файлы, хранящиеся в хранилище, то выполнение данной функции на стадии Reduce могло бы привести к тому, что эти файлы пришлось бы пересылать по сети, что противоречит концепции BigData, поэтому было решено исполнять данную функцию только на стадии Map и игнорировать ее на стадии Reduce.

3.4. Out. Указывает, что именно должно являться результатом выполнения задачи. Использует данные VSO и предметных библиотек.

Данные хранятся распределенно, и различные функции анализа могут быть к ним применены только после того, как будут выполнены некоторые условия. Такие условия должны быть описаны в предметных библиотеках. Помимо этого, предметные библиотеки должны содержать следующее:

1. Атомарные структуры данных, хранящиеся в файлах;

2. Конструируемые структуры данных, формируемые в ходе анализа данных;

3. Унифицированные алгоритмы чтения данных, доступных в хранилище;

4. Алгоритмы обработки и анализа структур данных 1, 2.

Именно за счет этих библиотек мы задаем динамическую структуру нашего языка. Для перехода от одной предметной области к другой достаточно просто подключить другие библиотеки.

Пример использования DSL

Разработанная технология была протестирована на задаче анализа исторических данных для Балтийского моря (рис. 2). Одна из задач – анализ влияния циклонов на штормовые приливы и следующие за ними наводнения, которые возможны в Санкт-Петербурге [13]. Набор метеоданных из открытых источников, таких как HIRLAM (http://www.hirlam.org/), GFS (http://www.emc.ncep.noaa.gov/index.php?branch=GFS), BOOS (http://www.boos.org/) хранился в dStorage. Были созданы высокоуровневые структуры данных, т.е., написаны библиотеки предметной области для таких объектов, как прогноз атмосферного давления (pressure forecast) и прогноз уровня воды в Балтийском море (level forecast).

Пример скрипта для обработки гидрометеорологических данных:

```
select Pressure_forecast
where (
    threshold == 970
```

time in (01.01.2011 31.12.2011)
cyclone().direction == “North-East”
)
simulate VSO Sea from Pressure_forecast to Level_forecast
out Level_forecast.at(59.929, 30.270)

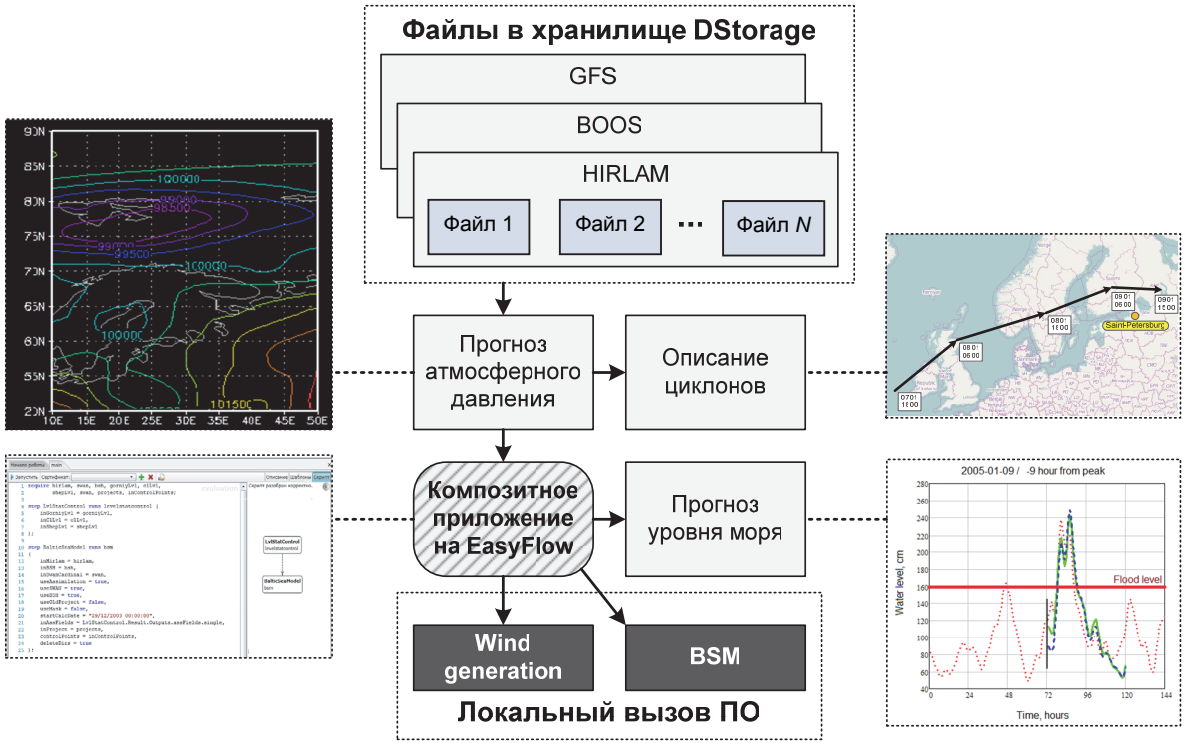


Рис. 2. Обработка гидрометеорологических данных
с целью анализа влияния циклонов

Функция select указывает, что данные должны быть взяты из хранилища и должны быть созданы объекты Pressure_forecast. Дальше в процессе интерпретации скрипта проверяются условия в блоке where, и для всех объектов, прошедших проверку, с помощью вызова сервиса VSO, доступного в облачной платформе CLAVIRE, по сценарию Sea строится абстрактный WF, результатом выполнения которого будет структура данных Level_forecast. В данном случае формируется WF, обеспечивающий вызов двух прикладных пакетов: Wind generation и BSM. Они будут исполнены локально на узлах хранилища. Последняя строка указывает, что результатом выполнения всего скрипта должен быть прогноз уровня воды в конкретной точке Балтийского моря ((59.929, 30.270) – географические координаты пункта измерений в г. Санкт-Петербург, которые будут преобразованы к координатам сетки BSM внутри библиотеки предметной области).

Вывод

С помощью динамического DSL/реализуется гибкий инструмент для высокоуровневого описания задач анализа данных. Технология, основанная на знаниях, используемых для интерпретации запроса, позволяет интерпретировать термины и структуры, определенные по семантике конкретной предметной области (или нескольких, в случае междисциплинарных задач). DSL интегрируется с облачной средой CLAVIRE и позволяет разрабатывать гибридные решения совместно с DSL для WF и PackageBase. Комплексная система для высокоуровневого описания задач, позволяющая автоматическое построение гибридных композитных приложений, в настоящее время разрабатывается на основе представленной технологии, интегрированной с существующими решениями платформы CLAVIRE.

Работа выполнена в рамках проекта "Управление большими данными в вычислительно-интенсивных приложениях" (соглашение №14613, 2014 г.).

Список литературы

1. Hey T., Tansley S., Tolle K. The fourth paradigm: data-intensive scientific discovery. 2009. – P. 252.

2. Philip Chen C.L., Zhang C.-Y. Data-Intensive Applications, Challenges, Techniques and Technologies: A Survey on Big Data // Information Sciences, 2014.

3. Dean J., Ghemawat S. MapReduce: simplified data processing on large clusters // Communications of the ACM. – 2008. – Т. 51. – № 1. – С. 107–113.

4. Big data computing and clouds: Trends and future directions / M.D. Assunção [et al.] // Journal of Parallel and Distributed Computing (2014).

5. Examining the challenges of scientific workflows / Y. Gil [et al.] // Ieee computer 40.12 2007. – P. 26–34.

6. Getting Code Near the Data: A Study of Generating Customized Data Intensive Scientific Workflows with Domain Specific Language / A. Manjunatha [et al.] // 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010.

7. Baranowski M., Belloum A., Bubak M. MapReduce Operations with WS-VLAM Workflow Management System / Procedia Computer Science, 2013. – Vol. 18. – P. 2599–2602.

8. Scientific workflow design for mere mortals / T. McPhillips [et al.] // Future Generation Computer Systems. – 2009. – Vol. 25, № 5. – P. 541–551.

9. Smirnov P.A., Kovalchuk S.V., Boukhanovsky A.V. Knowledge-Based Support for Complex Systems Exploration in Distributed Problem Solving

Environments // in Communications in Computer and Information Science. Knowledge Engineering and the Semantic Web. – 2013. – Vol. 394. – P. 147–161.

10. CLAVIRE: e-Science infrastructure for data-driven computing / K.V. Knyazkov, S.V. Kovalchuk, T.N. Tchurov, S.V. Maryin, A.V. Boukhanovsky // Journal of Computational Science. – 2012. – vol. 3, № 6. – P. 504–510.

11. Virtual Simulation Objects Concept as a Framework for System-Level Simulation / S.V. Kovalchuk, P.A. Smirnov, S.S. Kosukhin, A.V. Boukhanovsky // IEEE 8th International Conference on E-Science, 2012.

12. Ковальчук С.В., Разумовский А.В., Спивак А.И. Предметно-ориентированная технология хранения и обработки больших объемов данных в облачной платформе CLAVIRE // Динамика сложных систем – XXI век. – 2013. – № 3. – Т. 7. – С. 106–110.

13. Averkiev A.S., Klevanny K.A. Determining cyclone trajectories and velocities leading to extreme sea level rises in the Gulf of Finland // Russian Meteorology and Hydrology. – 2007. – № 32 (8). – P. 514–519.

А.Р. Кабирова, И.М. Губайдуллин

Институт нефтехимии и катализа РАН, Уфа

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ ПРИ ПРОГНОЗИРОВАНИИ ОСЛОЖНЕНИЙ В БУРЕНИИ

В работе рассматривается применение искусственных нейронных сетей (ИНС) к решению задач прогнозирования осложнений в бурении нефтегазовых скважин. Предлагается методика прогнозирования возможных осложнений при бурении новых скважин на основе информации о скважинном фонде месторождений. Нейронная сеть, построенная на основе радиально-базисных функций и обучаемая с помощью метода k-средних, показала наибольшую эффективность для решения указанной задачи классификации. Применение технологии параллельных вычислений позволит сократить временные затраты на проведение расчета и увеличить точность прогноза.

Введение

В настоящее время нефтегазовая отрасль, несмотря на сверхприбыльность, остается очень затратной. Одна из важнейших ее составляющих – бурение скважин – требует не только денежных вложений, но и временных затрат. Строительство скважины осуществляется на основе проектной документации, при этом от корректности данных, содержащихся в проекте, зависят как технико-экономические показатели бурения, так и экологическая безопасность и безаварийность. Тем не менее вплоть до на-

стоящего времени во многих нефтяных компаниях проекты на строительство скважин составляются на базе разрозненных данных или на основе первичного анализа опыта буровиков. Повысить эффективность проектирования строительства скважин возможно путем автоматизированного анализа данных по ранее пробуренным скважинам с использованием современных информационных технологий.

В данной работе рассматривается применение эффективного метода искусственного интеллекта – искусственных нейронных сетей – для прогнозирования осложнений при строительстве нефтегазовых скважин. Необходимость обработки большого объема данных из экспериментальной базы и использование нейронной сети порождают значительную вычислительную сложность задачи, что обусловливает целесообразность использования для ее решения параллельных вычислений.

Прогнозирование осложнений при бурении нефтегазовых скважин

Одним из способов эффективного предупреждения интервалов осложнений является обоснованный подбор буровых растворов на основе анализа данных по ранее пробуренным скважинам с использованием искусственных нейронных сетей – одного из наиболее широко распространенных и эффективных инструментов искусственного интеллекта. Нейронные сети обладают способностью предсказания ситуаций с неизвестным видом связей между входными и выходными параметрами, благодаря чему позволяют найти наилучшие значения свойств модели, минимизирующие погрешность вычислений внутри заданной области (в рассматриваемом случае это границы месторождения) [1]. Кроме того, нейронные сети являются оптимальным инструментом для построения прогноза на основе минимума информации по ранее пробуренным скважинам, в то время как для реализации традиционно используемых регрессионных уравнений и вероятностных моделей [2] требуется знание градиентов давлений в скважине, параметров бурового раствора и характеристик спуско-подъемных операций, а также множество других показателей, которые не всегда возможно оперативно получить.

Постановка задачи и используемые нейронные сети

В данной работе рассматривается методика прогнозирования осложнений, позволяющая строить прогноз на основе данных по ранее пробуренным скважинам месторождения (координаты скважины, наличие и/или интенсивность осложнения, глубина осложненного объекта). Таким образом, формально рассматриваемая задача для каждого объекта сводится к

задаче идентификации системы с несколькими входами и выходами на основе множества маркированных примеров (в качестве которых выступают уже пробуренные скважины).

Традиционно процесс обучения искусственной нейронной сети включает следующие стадии [3]:

1) формулировка постановки задачи и выделение набора ключевых параметров;

2) структурная идентификация нейронной сети, т.е. выбор модели (включающей в себя вид входных данных, пороговой функции, структуры сети и алгоритмов обучения), наиболее подходящей для решения поставленной задачи;

3) подготовка множества обучающих примеров, организованных в виде наборов входных данных, ассоциированных с известными выходными значениями, т.е. проектирование, разработка и реализации базы данных (БД);

4) последовательное предъявление входных данных на вход сети и сравнение полученного выходного значения с эталоном с последующей параметрической идентификацией нейронной сети, т.е. подстройкой весовых коэффициентов межнейронных соединений для минимизации ошибки между реальным и желаемым выходом сети;

5) повторение обучения до тех пор, пока суммарная ошибка во всем множестве входных значений не достигнет приемлемого уровня, либо ИНС не придет в стационарное состояние (метод обратного распространения ошибки).

Для решения поставленной задачи рассматривались несколько типов нейронных сетей, реализующих обучение как с учителем, так и без учителя. Модельный эксперимент, описанный ниже, проводился для многослойного персептрона, являющегося наиболее простым и распространенным типом ИНС, а также для сети радиально-базисных функций (РБС). К преимуществу РБС относится более высокая скорость обучения по сравнению с наиболее популярным алгоритмом обратного распространения ошибки. Но РБС могут работать медленно при большом количестве элементов (количество нейронов скрытого слоя обычно должно соответствовать числу элементов обучающей последовательности). Эту проблему можно решить путем предварительной модификации параметров РБС, например на основе алгоритма k-средних. Нейронные сети, обучающиеся без учителя (самоорганизующиеся карты), больше подходят для решения задач классификации, чем идентификации функций, поэтому было решено не рассматривать их в рамках данного проекта.

Естественным и необходимым началом использования ИНС является создание базы данных (БД).

База данных по осложнениям

Поскольку осложнения, возникающие в процессе бурения скважины, во многом определяются геологией горных пород, слагающих ее стенки, то при прогнозировании осложнений большое значение приобретают накопление и обработка промыслового материала по ранее пробуренным скважинам. Особенность объектов рассматриваемой области состоит в необходимости извлечения информации из разных информационных систем и баз данных, зачастую неполной по отдельным скважинам, и в то же время имеющей большой объем для месторождения в целом.

Для рассматриваемой задачи наиболее приемлемой является реляционная структура базы данных, при которой данные предоставляются в виде таблиц. В них собраны данные по месторождениям, необходимые для построения и анализа карт осложнений. Поскольку в данной работе рассматриваются геологические факторы возникновения осложнений, то нас в первую очередь интересует информация о координатах скважин конкретных месторождений, наличии и видах осложнений, возникших при их бурении, глубине залегания и стратиграфическом подразделении объектов осложнения. Под стратиграфическим объектом понимается совокупность горных пород, обладающих общими свойствами и залегающих в разрезе в определенной последовательности. На основании анализа промыслового материала, содержащегося в базе данных, производятся разработка и реализация математического аппарата по прогнозированию возможных осложнений при строительстве новых скважин.

База данных по осложнениям содержит как справочную информацию (таблицы «Месторождение», «Тип осложнения», «Стратиграфия»), так и промысловую информацию по месторождениям (таблицы «Объект», «Скважина», «Осложнение»). Структурная схема БД представлена на рис. 1.

Рис. 1. Структура базы данных по осложнениям

Необходимая для прогнозирования информация содержится в разных источниках, поэтому разработана программа, позволяющая собрать данные воедино и проанализировать их. В качестве исходных данных используются выгрузки из существующих баз данных в виде xls-файлов.

Одна и та же скважина разбуривает различные геологические породы, при этом на разных глубинах, относящихся к разным стратиграфическим подразделениям, могут возникать разные осложнения. Поэтому загруженная в БД информация автоматически структурируется по стратиграфическим объектам.

Структура и техническая реализация БД осложнений по пробуренным скважинам позволяют использовать её для проведения модельных экспериментов с различными типами нейронных сетей.

Модельный эксперимент

Для выбора наиболее эффективной нейронной сети был разработан модельный пример, включающий 30 точек с различными значениями функции (от 0 до 1) в соответствии с цветовой палитрой. Проведен вычислительный эксперимент по идентификации значений всех точек плоскости, реализующий использование выбранных типов нейронных сетей.

Поскольку любая нейронная сеть представляет собой совокупность узлов (нейронов), соединенных между собой, целью вычислительного эксперимента является структурная и параметрическая идентификация наиболее эффективной нейронной сети, что означает выбор ее типа и определение параметров нейронов. Наиболее эффективной (по соотношению точности и скорости вычислений) для решения поставленной задачи оказалась гибридная сеть, представляющая собой радиально-базисную сеть, которая обучается на основе алгоритма k-средних (таблица).

Выбор нейронной сети

Тип нейронной сети	Параметры сети	Время счета	Точность
Многослойный персептрон	1 скрытый слой, 50 нейронов	40 мин	90 %
Радиально-базисная сеть	30 нейронов	10 с	93 %
Радиально-базисная сеть с алгоритмом k средних	8 нейронов	< 1 с	97 %

Таким образом, алгоритм k-средних позволяет скорректировать число нейронов радиально-базисной сети и повысить ее производительность.

Реализация алгоритма прогнозирования осложнений в месторождениях

На основе спроектированной нейронной сети разработан алгоритм программного построения кластеризованной карты месторождения по наличию/интенсивности осложнений, которая позволяет отследить тенденцию их распространения для каждого стратиграфического объекта в любой точке в пределах месторождения (рис. 2).

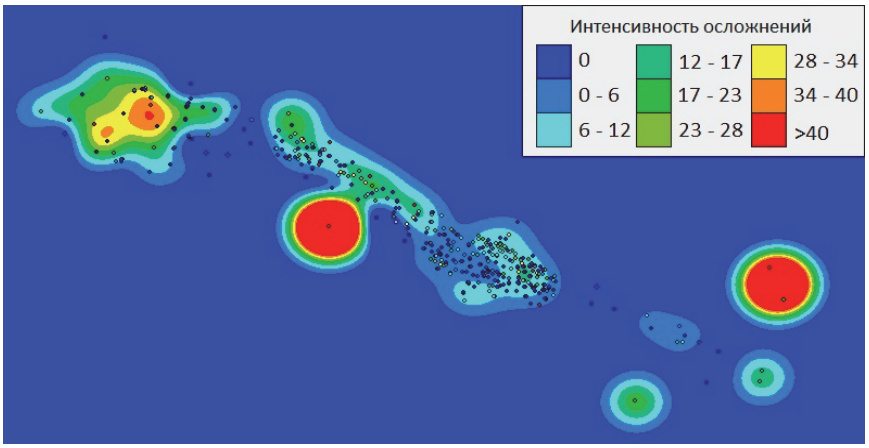


Рис. 2. Кластеризованная карта месторождения
(точки обозначают уже пробуренные скважины)

При вводе координат новой скважины с использованием кластеризованной карты месторождения программно строится прогноз по наличию/интенсивности осложнений для всех объектов, разбуриваемых данной скважиной.

На основе построенного прогноза проводится выдача рекомендаций по типу, параметрам бурового раствора и характеристикам спуско-подъемных операций при строительстве новой скважины.

Большое число записей в БД по осложнениям, решение многопараметрической задачи по подбору эффективного бурового раствора, а также необходимость сокращения времени разработки новых месторождений с учетом большой конкуренции на рынке нефтедобычи обусловливают использование технологий параллельных вычислений.

6. Распараллеливание вычислительного процесса

Для распараллеливания вычислительного процесса при решении задачи прогнозирования предлагается четырехуровневая модель распараллеливания:

1) по типам осложнений (поглощения бурового раствора, флюидопроявления, осыпи и обвалы, провалы и прихваты бурового инструмента);

2) по месторождениям;

3) по горизонтам (четвертичная, неогеновая, пермская, каменноугольная система, верхний, средний, нижний карбон и т.д.);

4) по классам интенсивности (нет осложнения, низкой, средней или высокой интенсивности).

Данная модель включает основные принципы разработанной авторами методологии распараллеливания [4]: использование внутреннего параллелизма задачи, распараллеливание по экспериментальной базе и декомпозиция метода решения задачи. Распараллеливание вычислительного процесса осуществляется с использованием интерфейса MPI.

Внутренний параллелизм задачи состоит в том, что хотя разные типы осложнений и могут быть обусловлены одними и теми же причинами, но имеют разную физическую природу и приводят к разным последствиям, в связи с чем их можно рассматривать независимо друг от друга. В настоящее время организация вычислительного процесса осуществляется комбинацией распараллеливания по экспериментальной базе и декомпозиции метода решения задачи.

Распараллеливание по экспериментальной базе состоит из двух уровней: распараллеливание по месторождениям и по горизонтам месторождения. Поскольку геологические факторы возникновения поглощений для каждого горизонта месторождения специфичны, то становится возможным при прогнозировании для одной скважины проведение расчетов для всех горизонтов независимо друг от друга.

Декомпозиция метода решения задачи прогнозирования основана на том, что при кластеризации карты интенсивности поглощений с использованием нейронной сети достройку узлов по каждому классу интенсивности можно проводить независимо. Время работы каждого процесса при этом значительно превосходит время межпроцессорного взаимодействия, что обусловливает эффективность работы параллельной программы.

Результаты и выводы

Тестирование разработанной программы проводилось для нескольких месторождений РБ [5]. С целью сравнения прогнозных значений с фактическими данными расчет был проведен для уже пробуренных скважин, результаты совпадают в 80 % случаев. Погрешность прогноза вызвана влиянием на возникновение осложнений различных технологических факторов (свойства буровых растворов, параметры технологических операций и т.п.) и компенсируется оперативностью получения прогноза на основе минимума входной информации. В целом искусственные нейронные сети позволяют проводить структурную и параметрическую идентифика-

цию объектов для эффективного решения задач нефтегазовой отрасли, использовать современные информационные технологии в виде гибких БД с элементами параллельного доступа и реализации параллельных алгоритмов прогнозирования осложнений.

Список литературы

1. Haykin S. Neural Networks: A Comprehensive Foundation. – Pearson Education, 2005. – 823 p.

2. Ясов В.Г., Мыслюк М.А. Прогнозирование поглощений при разбуривании трещиноватых коллекторов / ВНИИОЭНГ. – М., 1982. – С. 1–54.

3. Горбань А.Н. Обучение нейронных сетей. – М.: ПараГраф, 1990. – С. 1–160.

4. Линд Ю.Б., Губайдуллин И.М., Мулюков Р.А. Методология параллельных вычислений для решения задач химической кинетики и буровой технологии // Системы управления и информационные технологии. – 2009. – № 2(36). – С. 44–50.

5. Оперативное прогнозирование осложнений при бурении / Ю.Б. Линд, Р.А. Мулюков, А.Р. Кабирова, А.Р. Мурзагалин // Нефтяное хозяйство. – 2013. – № 2. – С. 55–57.

Г.Р. Карамутдинова[1], А.Ф. Шайхнурова[1], И.М. Губайдуллин[2]

[1] Башкирский государственный университет, Уфа
[2] Институт нефтехимии и катализа, Уфа

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ОПТИМИЗАЦИЯ ТРАНСПОРТНЫХ ПОТОКОВ МИЛЛИОННОГО ГОРОДА УФЫ

Разработан программный комплекс, реализующий оптимизацию транспортной сети, а именно составление наиболее точного расписания движения общественного транспорта, оценивание количества перевозимых пассажиров по типам пассажиров, по видам транспорта, маршрутам и направлениям, определение зависимости топологии сети и заторов. Для реализации модели предложен способ распараллеливания на несколько процессов, который сопоставлен с районами города.

Введение

На сегодняшний день в России состояние дорожно-транспортной системы оставляет желать лучшего. Количество транспорта в городах неизбежно увеличивается, а это, в свою очередь, приводит к возникновению

пробок, затруднительному передвижению, ухудшению условий для нормального развития города. Иными словами, возникает необходимость решения проблем, связанных с движением транспорта.

Для решения такого рода задач пользуются математическими моделями. Основными проблемами в большинстве транспортных моделей являются большой поток данных, сложность моделирования и оптимизация работы общественной сети миллионного города. Поэтому представляется возможным использование параллельного программирования в задаче моделирования транспортных потоков для повышения производительности вычислений.

Целью работы является разработка программного комплекса с использованием технологий параллельных вычислений, позволяющая моделировать и оптимизировать поведение транспортных потоков.

Для достижения поставленной цели выдвигаются следующие задачи:

1. Составление наиболее точного расписания движения общественного транспорта.
2. Определение загрузки элементов улично-дорожной сети.
3. Оценивание количества перевозимых пассажиров по типам пассажиров, по видам транспорта, маршрутам и направлениям.
4. Оценивание интенсивности пассажиропотоков между различными пунктами.
5. Определение зависимости между топологией сети и заторами, пробками.
6. Увеличение средней скорости транспортного потока через регион.

Описание модели

В настоящее время разработано много моделей, позволяющих учесть особенности формирования транспортных потоков. В данной работе рассматривается энтропийная модель [4] , позволяющая рассчитать матрицу корреспонденций [1, 5]:

$$\min_{T_{ij}} \left(\sum_{i}^{N} \sum_{j}^{M} T_{ij} c_{ij} + \beta \sum_{i}^{N} \sum_{j}^{M} \ln T_{ij} \right), \tag{1}$$

$$\sum_{j}^{N} T_{ij} = D_j , \tag{2}$$

$$\sum_{j}^{M} T_{ij} = Q_i , \tag{3}$$

$$T_{ij} \geq 0 , \tag{4}$$

где T_{ij} – корреспонденция из зоны i в зону j, β – средневзвешенная стоимость проезда, c_{ij} – стоимость проезда единицы потока из зоны i в зону j, Q_i – количество поездок из зоны i, D_j – количество поездок в зоны j.

Условие (2) означает, что суммарный поток (сумма числа поездок), который выехал из всех зон $i = 1, ..., N$ в зону j должен быть равен потоку, который прибыл в зону j. Условие (3) означает, что суммарный поток, который выехал обратно из всех зон $j = 1, ..., M$ в зону i должен совпадать с числом прибывших в зону i. Суммарное количество выехавших должно быть равно суммарному количеству прибывших, то есть должно выполняться условие $\sum_{i}^{N} Q_i = \sum_{j}^{N} D_j$. Потоки не должны быть отрицательными.

На основе данной математической модели приведен алгоритм построения матрицы корреспонденций с использованием параллельного программирования технологии MPI, которое способно в режиме реального времени решать вышеуказанные задачи.

Подготовка данных

Для расчета матрицы корреспонденций энтропийным методом необходимо определить:

– матрицу отправления;

– матрицу прибытия;

– матрицу затрат.

Для этого нужно знать условные зоны (сегменты), из которых люди поедут (на работу, учебу), и условные зоны, в которые они будут приезжать (работа, учебные заведения, организации).

Территория города делится на условные зоны при помощи прямоугольной сетки с шагом 500 м. Для этого используется электронная карта города Уфы. Получается матрица размерностью 106×76. Каждый получившийся сегмент города будет являться и «точкой» отправления и «точкой» прибытия. Следовательно, матрица отправления и матрица прибытия будут иметь одинаковую размерность 106×76. Сегменты, где никто не живет и не работает, будут равны нулю.

Для определения матрицы отправления Q необходимо при помощи карты города подсчитать «вручную» в каждой зоне число жилых домов разной этажности.

Количество жилых домов разной этажности, находящихся в сегменте *(i;j)*, умножается на соответствующее число проживающих людей в доме данной этажности и затем суммируется.

Полученное при помощи такого метода количество жителей сравнивается с известным количеством постоянного проживающего населения города для оценки погрешности расчетов.

На территории города, кроме жителей города, работают (учатся) люди с близлежащих поселков (сел, деревень). Поэтому учитывается поток из сельских населенных пунктов. Данная работа посвящена построению матрицы корреспонденций экономически активного населения, то есть необходимо умножить матрицу отправления на процент занятых.

Для нахождения матрицы прибытия D определяются места притяжения населения и их емкость: медицинские учреждения, учебные заведения, предприятия и другие.

В качестве матрицы затрат на передвижение из зоны i в зону j берутся временные затраты.

Матрица временных затрат вычисляется по следующей формуле:

$$V_{ij} = d_{ij} \cdot s_{ij},$$

где s_{ij} – скорость передвижения индивидуума в транспортной сети из условной зоны i в зону j, d_{ij} – расстояние от сегмента i до сегмента j.

Поскольку некоторые элементы получившихся матриц будут иметь нули, то данные матрицы будут разреженными, для хранения которых предлагается использовать CRS-формат. Для этого используются три массива:

– первый массив (Value) хранит значения элементов построчно (строки рассматриваются по порядку сверху вниз);

– второй (Col) – хранит номера столбцов для каждого элемента;

– третий (Row) – заменяет номера строк, используемые в координатном формате, на индекс начала каждой строки.

Количество элементов массива Row равно N+1. Элементы строки i в массиве Value находятся по индексам от Row [i] до Row [i + 1]–1 включительно. Таким образом обрабатывается случай пустых строк, а также добавляется «лишний» элемент в массив Row – устраняется особенность при доступе к элементам последней строки.

Вычислительный эксперимент

Для построения матрицы корреспонденций энтропийным методом используется алгоритм, который основан на методе балансировки [6, 7], где на каждом шагу выполняется баланс попеременно относительно строк или столбцов, а через некоторое число шагов эта процедура приводит к матрице корреспонденций.

1. В качестве начального приближения выбирается матрица временных затрат $T^0 = \exp^{-V}$ (V – время, затрачиваемое на передвижение из условной зоны i в условную зону j). Причем итерация $p = 0$.

2. Четные столбцы матрицы T^p умножаются на коэффициент калибровки γ (определяет чувствительность корреспонденций к фактору дальности – для трудовых поездок $\gamma = 0,065$) и полученные данные передаются от нулевого процесса первому при помощи функции MPI_Send. А действия для нечетных столбцов выполняет первый процесс, принимая данные от нулевого процесса функцией MPI_Recv. Таким образом, проверяется выполнение условия $\sum_j^M T_{ij}^p = Q_i$, $\forall i = 1,...,N$.

3. Полученная матрица обозначается как T^{p+1} и полагается, что $p = p + 1$.

4. Если условие $\sum_i^N T_{ij}^p = D_j$, $\forall j = 1,...,M$ выполняется, то алгоритм прекращает работу. Иначе осуществляется переход к шагу 5.

5. Четные строки матрицы T^p умножаются на коэффициент калибровки γ, и полученные данные передаются от нулевого процесса первому при помощи функции MPI_Send. А действия для нечетных строк выполняет первый процесс. Таким образом, проверяется выполнение условия $\sum_i^N T_{ij}^p = D_j$, $\forall j = 1,...,M$.

6. Полученная матрица обозначается как T^{p+1} и полагается, что $p = p + 1$. Далее осуществляется переход к шагу 2.

Полученная матрица T^p будет матрицей корреспонденций, а число p показывает, за какое число шагов было получено решение.

Как показывают расчеты, построение матрицы корреспонденций выполняется на одном процессе, но если увеличить размерности матриц, уменьшая шаг разбиения сегментов, то требуется повышение скорости вычислений на порядки.

Результаты тестирования показывают эффективность параллельной версии для моделирования и возможность управления транспортными потоками с числом транспортных средств порядка 10^6.

Заключение

Решение и оптимизация рассмотренных задач приводят к требованию максимального повышения производительности вычислений, учитывая масштабы времени в различных ситуационных моментах.

Например, если речь идет об управлении движением транспорта, то число расчетов достаточно велико и они должны происходить быстрее.

В работе учитывается тип получившихся матриц, а именно разреженных. Предлагается специальный алгоритм их хранения для упрощения расчетов и быстродействия.

В разработанном программном комплексе используется технология MPI параллельного программирования. В приложении осуществляется схема распараллеливания по регионам, где расчеты происходят независимо. Благодаря распараллеливанию время построения матрицы корреспонденций сокращается.

Список литературы

1. Моделирование транспортных потоков в крупном городе с применением к Московской агломерации / А.С. Алиев, А.И. Стрельников, В.И. Швецов, Ю.З. Шершевский // Автоматика и телемеханика. – 2005. – № 11. – С. 113–125.

2. Васильева Е.В., Игудин Р.В., Лившиц В.Н. Оптимизация планирования и управления транспортными системами. – М.: Транспорт, 1987.

3. Васильева Е.В., Левит Б.Ю., Лившиц В.Н. Нелинейные транспортные задачи на сетях. – М.: Финансы и статистика, 1981.

4. Вильсон А. Дж. Энтропийные методы моделирования сложных систем. – М.: Наука, 1978.

5. Введение в математическое моделирование транспортных потоков / А.В. Гасников, С.Л. Кленов, Е.А. Нурминский, Я.А. Холодов, Н.Б. Шамрай. – М.: Изд-во МФТИ, 2010.

Г.Г. Кашеварова, А. С. Мартиросян

Пермский национальный исследовательский политехнический университет

ВЗАИМОДЕЙСТВИЕ ВОЛНЫ ПРОРЫВА С СООРУЖЕНИЯМИ ПРИ ГИДРОДИНАМИЧЕСКОЙ АВАРИИ

В работе, в ходе решения связанной задачи в ANSYS и Flow Vision, определяются динамические нагрузки на здания, находящиеся в потенциально опасной зоне волны прорыва, в зависимости от их формы, размеров, ориентации относительно потока и проницаемости объекта, а также возможность и степень разрушения здания в зависимости от его конструкции и места положения.

Проблема возможного разрушения гидроузлов – одна из наиболее актуальных проблем современности. Последствия аварий подпорных гидротехнических сооружений можно сравнить с последствиями некоторых природных катастроф – прорывом ледниковых запруд, прохождением селей и т.д. За последние 70 лет произошло более тысячи аварий на крупных гидротехнических сооружениях.

Волна прорыва подобно цунами сметает по пути распространения любые строения на огромной территории, в том числе даже очень прочные конструкции гидросооружений, расположенных ниже по течению реки (рис. 1).

Рис. 1. Прорывы дамб в Киеве (1961 г.) и Джакарте

Разрушение (прорыв) гидротехнических сооружений происходит в результате:

– действия сил природы (землетрясений, ураганов, размывов плотин, износа и старения оборудования);

– техногенных чрезвычайных ситуаций (заклинивание затворов, разрушение шлюзовых ворот, удар судна и др);

– воздействий человека (просчеты при проектировании или ошибки при эксплуатации ГТС: непрофессиональные действия персонала, плохой контроль за состоянием гидросооружений и гидромеханического оборудования, несвоевременность и неполнота ремонтов);

– преднамеренных разрушений (терактов).

Возможно и сочетание ряда неблагоприятных факторов, которые подчас трудно предусмотреть в расчётах.

Катастрофическое затопление, являющееся следствием гидродинамической аварии, заключается в стремительном затоплении волной прорыва обширных территорий слоем воды более 0,5–10 м.

Основными поражающими факторами волны прорыва являются: высота волны, скорость движения и длительность затопления.

При разработке проектов крупных гидроузлов, создающих особую опасность затоплений в нижних бьефах, всегда предусматривается составление прогноза параметров потенциальной волны прорыва

В настоящее время при застройке перспективных территорий города, попадающих в зону возможного затопления, которое может произойти в результате гипотетической гидродинамической аварии на гидротехнических сооружениях, для принятия проектных решений требуется оценить механическую безопасность проектируемого здания и необходимость защитных мероприятий. Актуальной является задача минимизация ущерба, нанесённого стихией.

Установленные проектирующими организациями на крупномасштабных картах границы затоплений дают общее представление о возможных затопляемых территориях (что, без сомнения, представляет большую ценность), но не отражают различную степень опасности для конкретных зданий при прохождении волны прорыва.

Все известные нам методы определения гидродинамической нагрузки и последствий волны прорыва являются решением инженерных задач и предназначены для оперативного прогнозирования. Современные численные методы и программные комплексы позволяют более точно решать такие задачи, поэтому научное обоснование вопросов безопасности конкретных зданий и сооружений в потенциально опасной зоне волны прорыва с применением методов компьютерного моделирования является актуальным.

Основным поражающим фактором волны прорыва, приводящим к разрушению зданий и сооружений, является гидродинамическое давление потока воды. Величины нагрузок при воздействии потока волны прорыва на различные здания определяются параметрами потока (скоростью и глубиной потока вблизи объекта), а также параметрами самого здания.

Действие волны прорыва на здание достаточно сильно различается в зависимости от расположения здания и характера его конструктивного решения.

Нагрузки на строительные объекты от волн, действующих на обтекаемые преграды, в нормативных документах рекомендуется определять как сумму скоростного и инерционного компонентов, обусловленных соответственно локальными значениями скорости и ускорения волнового движения жидкости. При этом динамический эффект от воздействия нерегулярных волн предлагается учитывать только в тех случаях, когда период собственных колебаний сооружения соизмерим со средним периодом набегающих волн [1]. Вследствие самой природы жидкости течение ее нередко происходит весьма сложным образом, с образованием нестационарных эффектов, застойных зон и вихревых структур. Поэтому необходима дальнейшая более тщательная проработка многих положений, приемов расчетов. Эти факторы обусловливают растущий интерес к средствам математического моделирования течений жидкостей и газов с использованием современных программных комплексов и ЭВМ, позволяющих прогно-

зировать характеристики течений и параметры работы устройств на стадии их проектирования. Этим вопросам посвящены работы многих исследователей: А.Н. Кочевского, J.D. Anderson, Jr., C.T. Crowe, J.D. Swarzkopf, M. Sommerfeld, Y. Tsuji, Кондранина Т.В., Ткаченко Б.К., Березниковой М.В., Евдокимова А.В., Зуева А.П. [2–5].

Целью настоящей работы является определение динамических нагрузок на здания, находящиеся в потенциально опасной зоне волны прорыва, в зависимости от их формы, размеров, ориентации относительно направления течения потока и проницаемости объекта, а также определение возможности и степень разрушения здания в зависимости от его конструктивного решения и места положения.

Для Пермского края установлены границы затопления территории при наихудшем (наиболее тяжелом) сценарии развития гидродинамической аварии на гидротехнических сооружениях. Определены максимальные уровни воды и зоны затопления в Воткинском водохранилище при развитии прорыва в русловой грунтовой плотине Камской ГЭС, а также время добегания фронта и пика волны прорыва по акватории водохранилища до места расположения объекта строительства.

С физической точки зрения разнообразие волн на воде чрезвычайно велико. С математической точки зрения очевидно, что общего решения, охватывающего все возможные случаи, не существует. Одним из самых важных вопросов теории волн на воде является вопрос о границах пригодности различных решений, полученных при тех или иных упрощающих допущениях. В целом вопросы образования, распространения и взаимодействия волны прорыва с сооружениями остаются недостаточно изученными. Теоретически волна прорыва обычно рассматривается как потенциальное течение тяжелой идеальной несжимаемой жидкости, вызванное кратковременным импульсом или включением начального возвышения на свободной поверхности (при рассмотрении этапа распространения).

Результат действия на конструкции здания гидродинамической нагрузки зависит от взаимодействия ударяющей волны прорыва и здания.

При прохождении гидравлической волны мимо здания возникает сложная дифракционная картина взаимодействия, и объект подвергается воздействию нестационарных (изменяющихся во времени давлений). Поле давлений зависит от параметров волны, характеристик объекта и его ориентации относительно фронта волны. В настоящее время для решения таких задач чаще всего применяют численные методы и современные комплексы программ: CFX, Fluent, STAR-CD, FlowVision и др.

На рис. 2 для иллюстрации показан характер распределения давления при взаимодействии волны прорыва со зданием (изолинии) при углах атаки 0° и 45°. Расчет выполнялся в ПК FlowVision.

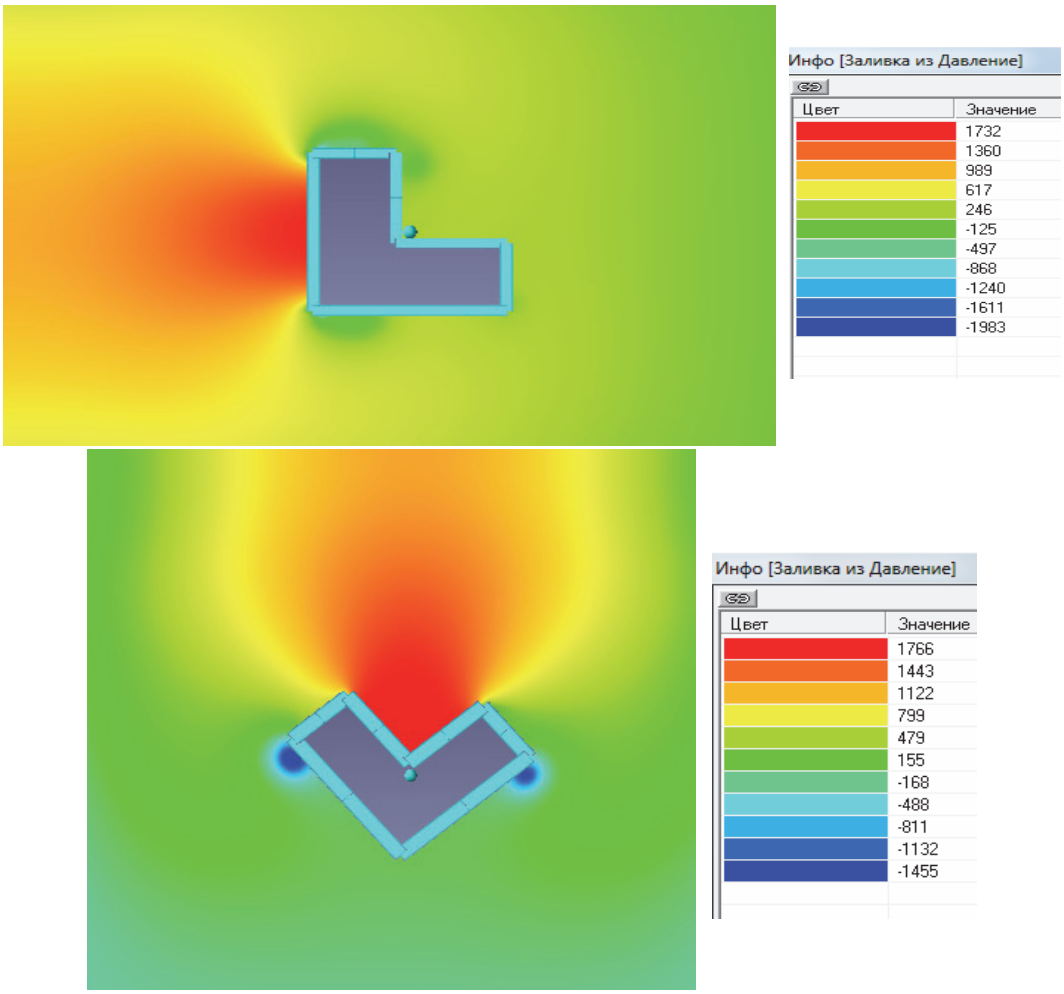


Рис. 2. Характер распределения давления при взаимодействии волны прорыва со зданием (изолинии) при угле атаки 0° и 45°

Наиболее рациональным способом обеспечения стойкости зданий является использование предохранительных конструкций (ПК), связанное с проемностью зданий, поскольку увеличение проемности стен приводит к снижению силового воздействия гидропотока. ПК в наружном ограждении помещения вскрываются при сравнительно небольшом давлении, что обеспечивает интенсивное истечение воды через проемы и снижение нагрузок на несущие колонны. Поток может проскользнуть сквозь первый этаж, вырвав перегородки, унеся оборудование, однако повреждения зданию будут минимальными

Для приближенной оценки величины гидродинамической нагрузки в СНиПе предлагаются эмпирические формулы для сооружений разного типа и графики для определения различных уточняющих коэффициентов. Следовательно, методы расчета нагрузки на сооружения от действия волны прорыва в основном являются решением инженерных задач и предназначены для оперативного прогнозирования.

Тенденцией развития ведущих программных продуктов является реализация в каждом из них набора математических моделей (ММ), позволяющих как можно более полно моделировать все встречающиеся на практике физические эффекты. Пользователь подключает нужные модели на стадии постановки задачи, задавая затем соответствующие граничные условия и прочие требуемые данные.

В данной работе нагрузки на здание определялись по разным методикам: рекомендованной СНиПом, разработанной авторами, моделированием процесса в трехмерной постановке в среде программного комплекса FlowVizion.

Для расчета нагрузок, которыми обмениваются вода и корпус здания, уравнения механики и гидродинамики решаются в связанной постановке на несвязанных сетках таким образом, что сетки лагранжевых конечных элементов, которыми моделируется здание, являются границами области течения для жидкости.

Трехмерные геометрические модели области течения, здания и расчетные сетки построены в препроцессоре ANSYS и записаны в формате пакета FlowVizion.

Механические свойства материала кирпичной кладки описывались как упруго-хрупкая среда, которая может растрескиваться и раскрашиваться [6–8].

Грунт моделировался недеформируемой плоскостью с целью воспроизводства поведения здания при его разрушении.

Задача решалась в связанной постановке, т.е. течение жидкости и деформация здания рассчитывались одновременно. Разработанная методика расчета позволяет оценить эффективность формы здания в плане, влияние размеров, ориентации относительно направления течения потока.

Список литературы

1. СНиП 2.06.04-82*. Нагрузки и воздействия на гидротехнические сооружения (волновые, ледовые и от судов) / Госстрой СССР. – М.: ЦИТП Госстроя СССР, 1986. – 40 с.

2. Кочевский А.Н. Расчет внутренних течений жидкости в каналах с помощью программного продукта FlowVision // Вестник СумГУ. – Сумы, 2004. – № 2 (61). – С. 25–36.

3. Anderson J.D., Computational Jr. Fluid Dynamics. The basics with applications. McGraw-Hill Science / Engineering / Math; 1 ed. – February 1, 1995.

4. Multiphase flows with droplets and particles / C.T. Crowe, J.D. Swarzkopf, M. Sommerfeld, Y. Tsuji // CRC Press; 1 ed. – November 13, 1997.

5. Применение пакетов прикладных программ при изучении курсов механики жидкости и газа: учеб. пособие / Т.В. Кондранин, Б.К. Ткаченко, М.В. Березникова, А.В. Евдокимов, А.П. Зуев. – М.: МФТИ, 2005. – 104 с.

6. Кашеварова Г.Г., Труфанов Н.А. Численное моделирование процессов деформирования и разрушения зданий в системе «здание – фундамент – основание» // Изв. вузов. Строительство и архитектура. – 2005. – № 10. – С. 113–116.

7. Кашеварова Г.Г., Труфанов Н.А. Численное моделирование деформирования и разрушения системы «здание – фундамент – основание» / УрО РАН. – Екатеринбург; Пермь, 2005. – 225 с.

8. Кашеварова Г.Г., Труфанов Н.А. Численный анализ эффективных упругих свойств материала кирпичной кладки // Механика композиционных материалов и конструкций. – 2005. – Т. 11, № 1. – С. 49–60.

А.Н. Коварцев, Д.А. Попова-Коварцева, Е.Е. Серповская

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (Национальный исследовательский университет), г. Самара

АЛГОРИТМ ГЛОБАЛЬНОГО ПОИСКА ТОЧЕК РАЗРЫВА ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ

Предлагается новый алгоритм глобального поиска точек разрыва функций многих переменных, ориентированного на обнаружение фатальных ошибок в математических моделях вычислительных алгоритмов программных продуктов. Эвристики, используемые в алгоритме, основанные на учете специфических особенностях решаемой задачи, обеспечили ему более высокую эффективность среди известных алгоритмов глобальной оптимизации применительно к задаче поиска точек бесконечного разрыва функции.

Введение

Современные математические модели (ММ) позволяют получить расчётным путём обширную и весьма точную информацию о процессах, происходящих в природе и в технике. Проведение вычислительных экспериментов с математической моделью, реализованной в виде компьютерной программы, обеспечивает сокращение сроков исследования и уменьшение его стоимости. Используемые в расчетах математические модели в конечном итоге воплощаются в виде программных продуктов (ПП). Ошибки (в том числе и вычислительного характера), допущенные на этапе разработки математической модели, естественным образом наследуются и в соответ-

ствующих программных продуктах [1, 2]. В связи с чем можно объединить процедуру обнаружения ошибок в математических моделях с процессом тестирования соответствующего ПП, что позволяет автоматизировать процессы поиска ошибочных ситуаций как в ММ, так и в программных продуктах.

Если рассматривать класс программ, построенных на основе численного анализа и вычислительной математики, т.е. программ, реализующих некоторые математические модели, то к наиболее сложно идентифицируемым ошибкам относятся фатальные вычислительные ошибки типа «деление на ноль», приводящие к переполнению порядка и остановке выполнения программы [3]. Тестирование подобного рода программных продуктов является сложной задачей, поскольку область комбинаций ошибочных исходных данных в этом случае невелика, а вероятность случайного попадания в эту область тестового набора данных (например, с помощью стохастических методов тестирования) ничтожно мала.

В то же время ошибки потери или переполнения порядка можно эффективно отыскивать, если их рассматривать с позиции поиска точек разрыва второго порядка и использования для этого алгоритмов глобальной оптимизации (ГО) функций многих переменных.

Постановка задачи и основные определения

Формально вычислительный модуль можно интерпретировать как векторную функцию $Y = F(X) = (f_1(X), f_2(X), ..., f_m(X))^T$, действующую из множества входных параметров $X = (x_1, x_2, ..., x_n)^T$ модуля на множество вычисляемых данных $Y = (y_1, y_2, ..., y_m)^T$. Область определения модуля опишем как декартово произведение доменов каждого из параметров $\Omega_X = \Omega_{x_1} \times \Omega_{x_2} \times ... \times \Omega_{x_n}$, где Ω_{x_k} – область определения *k*-го параметра модуля. По аналогии множество допустимых значений функции $F(X)$ представим как $\Omega_F = \Omega_{y_1} \times \Omega_{y_2} \times ... \times \Omega_{y_m}$, где Ω_{y_i} – область изменения *i*-й компоненты векторной функции.

Область ошибочных ситуаций определим как множество $\Omega_E \subset \Omega_X$, в котором при $X \in \Omega_E$ в программном модуле $F(X)$ возникает фатальная вычислительная ошибка. Если при этом $F(X) = \pm\infty$, то функция терпит разрыв второго рода.

Для точек разрыва второго рода мера множества ошибочных ситуаций равна нулю ($\mu(\Omega_E) = 0$), что создает серьезные трудности при поиске ошибок подобного типа. В работе [3] такие ошибки классифицированы как

«редкие». Исчерпывающий тест на обнаружение ошибки такого рода может потребовать миллиардов испытаний, т.е. полного перебора всех значений исходных данных модуля, что делает невозможным его практическую реализацию в течение разумного времени.

Идея метода поиска точек разрыва второго рода [3] основывается на предположении о том, что точки разрыва второго рода для математических функций можно обнаружить, решая одну из оптимизационных задач:

$$\max_X f_k(X) \ (\min_X f_k(X)), \ k = \overline{1,m}. \qquad (1)$$

Действительно, значение функции в точке разрыва второго рода условно принято обозначать $\pm\infty$, что соответствует максимально большому положительному или отрицательному числу, т.е. глобальному экстремуму в широком смысле этого слова. Применяя методы глобальной оптимизации, мы, несомненно, должны обнаружить и точки разрыва второго рода функции.

Оптимизационный подход к решению задачи поиска точек разрыва второго рода

Среди известных однопараметрических методов многоэкстремальной оптимизации наиболее эффективным является информационно-статистический метод Р.Г. Стронгина [4]. Метод основывается на использовании приближенного апостериорного распределения вероятностей расположения глобального экстремума, формируемого в процессе испытаний функции, что реализует более сбалансированную организацию поиска глобального минимума функции. Данная стратегия настолько эффективна, что её часто переносят с одномерного случая на случай оптимизации функций многих переменных.

Относительно оптимизируемой функции предполагается, что она является липшицевой функцией с константой K, т.е. для неё выполняется условие $\left| f(x') - f(x'') \right| \le K \left| x' - x'' \right|$.

В работе [40] показано, что поиск экстремума функции реализуется за счет максимизации достаточно простой характеристической функции:

$$R(i) = m(x_i - x_{i-1}) + \frac{(y_i - y_{i-1})^2}{m(x_i - x_{i-1})} - 2(y_i + y_{i-1}), \qquad (2)$$

где m – оценка константы Липшица, определяемая по результатам проведенных испытаний: $m = \begin{cases} 1 & M = 0, \\ rM & M > 0, \end{cases}$ $M = \max_i \frac{\left| y_i - y_{i-1} \right|}{(x_i - x_{i-1})}$, r – параметр.

С точки зрения решения задачи поиска точек разрыва второго рода метод Стронгина оказался не очень эффективным, поскольку этот метод эффективный для липшицевых функций, непрерывных по определению, теряет свои свойства для класса разрывных функций, существенно не липшицевых.

В связи с этим целесообразно разработать характеристический алгоритм, ориентированный на решение задачи поиска точек разрыва второго рода.

Для этого для разрывных функций попытаемся подобрать характеристическую функцию, подобную (2).

Проанализируем характеристическую функцию (2), полученную при реализации информационно-статистического подхода.

Для простоты положим $r = 1$. Формулу (2) представим в виде

$$R(i) = m\Delta x_i + \frac{\Delta y_i^{\,2}}{m\Delta x_i} - 2(y_i + y_{i-1}),\ \Delta x_i = (x_i - x_{i-1}),\ \Delta y_i = (y_i - y_{i-1}).$$

Учитывая, что $m = \max\limits_i \dfrac{|\,\Delta y_i\,|}{\Delta x_i} \approx \max\limits_i |\,f'(x_i)\,| = \max\limits_i |\,f_i'\,|$, имеем

$$R(i) = \max|f_i'|\Delta x_i + \frac{\Delta y_i^{\,2}}{\max|f_i'|\Delta x_i^2}\Delta x_i - 2(y_i + y_{i-1}) =$$

$$= \max|f_i'|\Delta x_i + \frac{f_i'^2}{\max|f_i'|}\Delta x_i - 2(y_i + y_{i-1}).$$

Воспользовавшись первыми двумя членами разложения в ряд Тейлора оптимизируемой функции в точке x_{i-1} ($y_i \approx y_{i-1} + f_i'\Delta x_i$), получим

$$R(i) = \max|f_i'|\Delta x_i + \frac{f_i'^2}{\max|f_i'|}\Delta x_i - 2f_i'\Delta x_i - 4y_{i-1} =$$

$$= \frac{(\max|f_i'|)^2 + f_i'^2 - 2f_i'\max|f_i'|)}{\max|f_i'|}\Delta x_i - 4y_{i-1} =$$

$$= \frac{(\max|f_i'| - f_i')^2}{\max|f_i'|}\Delta x_i - 4y_{i-1}.$$

Таким образом,

$$R(i) \approx \frac{(\max|f_i'| - f_i')^2}{\max|f_i'|}\Delta x_i - 4y_{i-1}. \qquad (3)$$

Анализ формулы (3) показывает, что в точках локального минимума при $f_i' = 0$ ($R(i) \approx \max\left|f_i'\right|\Delta x_i - 4y_{i-1}$) алгоритм в процессе поиска максимума характеристической функции ориентируется на глобальный минимум функции $y = f(x)$. В то же время учитываются и те участки области поиска, которые имеют сравнительно большие размеры Δx_i. Если на каком-либо участке функции первая производная близка к константе Липшица ($f_i' \approx \max\left|f_i'\right|$, $R(i) \approx -4y_{i-1}$), то характеристическая функция просто минимизирует $y = f(x)$. Таким образом, характеристическая функция «склонна» размещать точки испытаний либо в окрестностях локальных минимумов оптимизируемой функции, тяготея при этом к глобальному минимуму, либо в интервалах неопределенности, размеры которых велики по сравнению с другими участками функции.

Модифицируем характеристическую функцию (3), исходя из следующих предположений. Во первых, алгоритм поиска точек разрыва второго рода в принципе не должен ориентироваться на локальные экстремумы функции, он должен чутко реагировать на быстрое увеличение или уменьшение роста функции.

Во-вторых, сама по себе функция не несет в себе информации о своем будущем поведении. Первая производная показывает скорость роста функции, а вторая производная – ускорение роста. Очевидно, что вторая производная в этом смысле более информативна, поэтому в качестве первой составляющей характеристической функции выберем $(f_i'')^2$. Возведение в квадрат второй производной связано с тем обстоятельством, что нам не важен знак кривизны функции в точке разрыва.

В качестве меры неопределенности частичного интервала, по аналогии с методом Стронгина, выберем длину интервала, поправленного неким регулировочным параметром *r*. Учитывая изложенное, в качестве характеристической функции метода поиска точек разрыва можно взять функцию $R(i) = (f''(x_i))^2(\Delta x_i)^r$. В многомерном случае, например для функции 2 переменных, вместо второй производной можно использовать дифференциал второго порядка, а характеристическую функцию представить в виде

$$R(i) = (d^2 f(X_i))^2(\Delta x \Delta y)^r. \qquad (4)$$

Алгоритм поиска точек разрыва второго рода

Пусть областью поиска ошибок в вычислительном модуле является единичный квадрат $\Pi = [0, 1]\times[0, 1]$, который в процессе деления пропорционально разбивается на четыре квадрата меньших размеров (рис. 1).

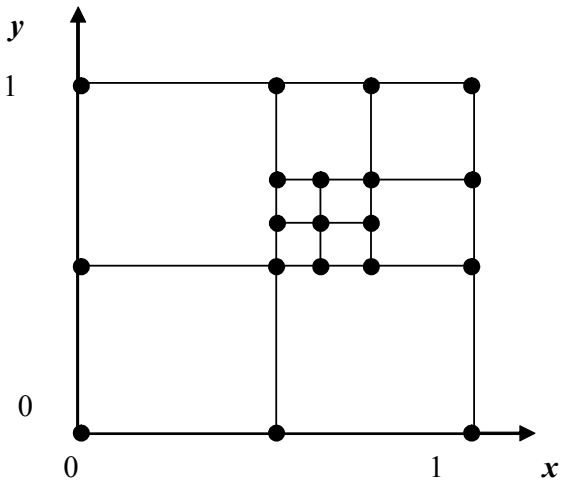


Рис. 1. Деление области поиска

Для вычисления второго дифференциала функции по её значениям, вычисленным в узловых точках сетки, несложно построить полный полином второго порядка $P_{2,2}(x,y)$, тогда $d^2 f(x,y) \approx P_{2,2}(x,y)$. Обобщая полученные результаты, несложно построить алгоритм поиска точек разрыва второго рода.

1. Произвести деление рассматриваемой *k*-й области поиска Q_k на четыре части: $Q_k = Q_{k+1} \bigcup Q_{k+2} \bigcup Q_{k+3} \bigcup Q_{k+4}$.

1.1. Для каждого Q_{k+l} $l = 1,2,3,4$ по результатам вычисления функции в узлах регулярной сетки $Z^{(k+l)} = \left\| z_{i,j}^{(k+l)} \right\|$, $i, j = 0,1,2$, строятся полиномы $P_{2,2}^{(k+l)}(x,y)$ и вычисляются характеристические функции $R(k+l) = (d^2 P_{2,2}^{(k+l)})^2 (h_k)^{2r}$, где h_k – длина стороны квадратов Q_{k+l}.

1.2. Вычисленные значения характеристической функции заносятся в характеристический список *R* в порядке убывания значений функции, т.е. $\forall i \; R(i) \geq R(i+1)$.

2. В качестве «решающего» элемента из списка *R* выбирается элемент с максимальной характеристикой, т.е. *R*(*1*). При этом первый элемент удаляется из списка *R*.

3. Если не выполняется условие остановки алгоритма $((\max_i \left| z_i \right| > M) \vee (\min_i h_i < \varepsilon))$, то перейти к п.1.

Вычислительные эксперименты

Для оценки эффективности предложенного алгоритма поиска точек разрывов второго рода обычно используются тестовые функции. В данной работе рассматривается известная модификация тестовой функции Растри-

гина [5], имеющей в единичном квадрате 625 локальных минимумов и 1 глобальный. Тест для поиска точек разрыва второго рода сформируем из исходной функции Растригина путем превращения точки глобального экстремума в точку разрыва. Модифицированная разрывная функция выглядит следующим образом:

$$F(x,y) = 1/(10\sin^2(\pi z_1) + (z_1 - 1)^2(1 + 10\sin^2(\pi z_2)) + (z_2 - 1)^2), \qquad (5)$$

где $z_1 = 1 - (11 - 20(x - a + 0.55))/4$; $z_2 = 1 - (11 - 20(y - b + 0.55))/4$;

$$x, y \in [0;1].$$

На рис. 2 показан 3D-график тестовой функции (5).

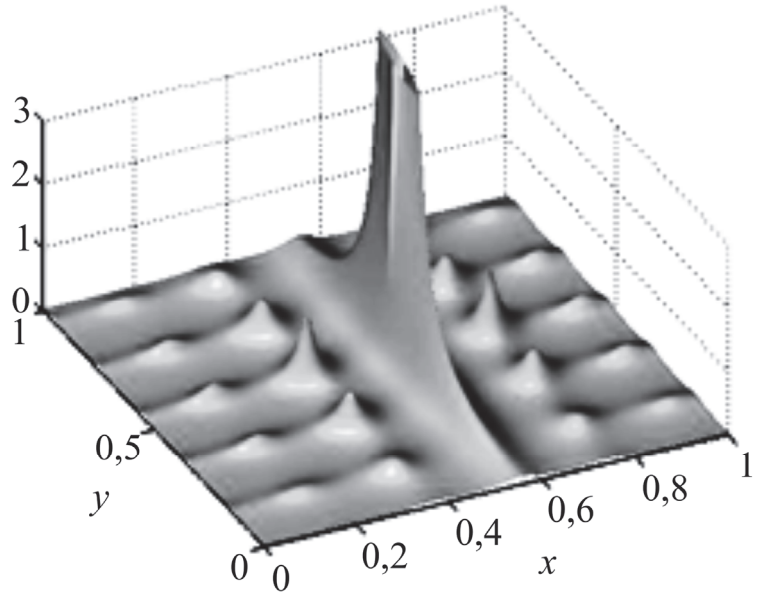


Рис. 2. Тестовая разрывная функция

Как видно из рисунка, тестирование функции (5) имеет большое количество остроконечных локальных экстремумов и «щелевидный», трубчатый разрыв второго рода, что осложняет его поиск. Об эффективности алгоритма поиска ошибочных ситуаций (АПОС) удобно судить на основании операционной характеристики эффективности поиска, введенной в работе [6] для анализа алгоритмов глобальной оптимизации. Операционная характеристика представляет собой зависимость вероятности обнаружения ошибочной ситуации $P_{\text{алг}}$ от числа обращений к тестируемой функции N_f.

На рис. 3 представлены операционные характеристики алгоритма поиска точек разрывов функции (сплошная линия) и алгоритма ГО методом половинных делений (МПД) [7] (пунктирная линия).

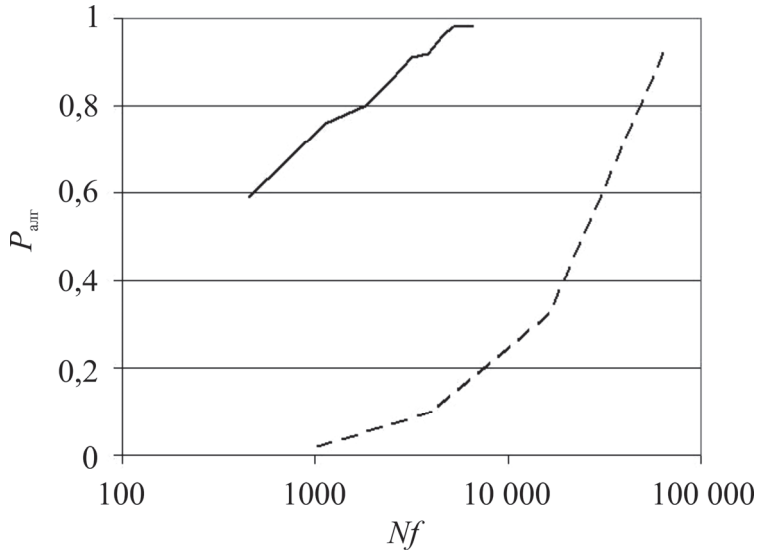


Рис. 3. Операционные характеристики
алгоритмов поиска точек разрывов

Как видно из рисунка, эффективность алгоритма АПОС существенно выше, чем для обычного алгоритма глобальной оптимизации, ориентированного на оптимизацию непрерывных функций, например МПД. Происходит это потому, что при попадании точки в окрестности разрыва функции МПД постоянно изменяет оценку константы Липшица, что приводит к необходимости более подробного исследования всех областей формирования локальных экстремумов функции. Алгоритм АПОС фактически игнорирует локальные экстремумы функции, ориентируясь на участки быстрого роста функции.

Заключение

В работе рассмотрен алгоритм глобального поиска точек разрыва второго рода, предназначенный для обнаружения фатальных ошибок в математических моделях вычислительных алгоритмов. Основная идея предложенного алгоритма состоит во введении в классический алгоритм глобальной оптимизации оригинальной эвристической характеристической функции, построенной на основе анализа характеристической функции Стронгина и учитывающей особенности решаемой задачи. Это позволило существенно повысить эффективность алгоритма поиска ошибочных ситуаций. Данный результат позволяет надеяться на организацию поиска ошибочных ситуаций для функций больших размерностей. В дальнейшем планируется развитие алгоритма АПОС с использованием современных суперкомпьютерных технологий.

Список литературы

1. Липаев В.В. Тестирование программ. – М.: Радио и связь, 1986. – 296 с.

2. Котляров В.П., Коликова Т.В. Основы тестирования программного обеспечения. – М.: БИНОМ, 206. – 285 с.

3. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 1999. – 150 с.

4. Стронгин Р.Г. Поиск глобального оптимума. – М.: Знание, 1990.

5. Абакаров А.Ш., Сушков Ю.А. Адаптация случайного поиска с использованием логистической кривой. – СПб.: Изд-во СПбГУ, 2005. – С. 67–75.

6. Гергель В.П., Стронгин Р.Г. Абсолют. Программная система для исследования и изучения методов глобальной оптимизации: учеб. пособие. – Н. Новгород: Изд-во Нижегород. ун-та, 1998. – 141 с.

7. Коварцев А.Н., Попова-Коварцева Д.А., Аболмасов П.В. Исследование эффективности глобальной параллельной оптимизации функций многих переменных // Вестник ННГУ. – 2013. – № 3 (1). – С. 252–261.

Е.А. Козинов[1], Т.В. Лаптева[1], И.Б. Мееров[1], М.В. Иванченко[1], С.В. Денисов[1,2]

[1] Нижегородский государственный университет им. Н.И. Лобачевского
[2] Университет Аугсбурга, Аугсбург, Германия

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ЧИСЛЕННЫЙ ФЛОКЕ-АНАЛИЗ НЕСТАЦИОНАРНЫХ КВАНТОВЫХ СИСТЕМ

Предложен новый численный метод интегрирования математических моделей квантовых систем, находящихся под периодическим внешним воздействием, который позволяет корректно провести флоке-анализ заданного числа состояний стационарного гамильтониана. Реализован параллельный численный алгоритм для исследования систем с большим числом состояний, продемонстрирована его хорошая масштабируемость.

Введение

Последние успехи в таких передовых областях экспериментальной физики, как слабодиссипативная оптика холодных атомов [1], динамика наноэлектромеханических систем в квантовых режимах [2] и резонаторная оптомеханика [3], выявили разрыв с накопленными теоретическими знаниями в физике нестационарных квантовых систем. В немалой степени это

связано с ограниченностью возможностей исследования достаточно детальных вычислительных моделей на современных суперкомпьютерных кластерах посредством имеющихся численных методов. Объяснение наблюдаемых явлений и количественное предсказание новых, таким образом, на настоящий момент затруднены.

Одной из наиболее интересных задач является исследование нестационарных систем, систем, подверженных внешней модуляции, поскольку получаемые здесь знания могут впоследствии быть использованы для управления состояниями квантовых систем в различных прикладных областях нанотехнологий: квантовая обработка информации, разработка оптических переключателей, компактных терагерцовых лазеров, ультрачувствительных гироскопов. Эти же задачи представляют и значительную сложность с вычислительной точки зрения, поскольку требуют не только нахождения собственных состояний стационарного гамильнониана, но и пропагации их во времени для получения Флоке-состояний.

В данной работе мы рассмотрим фундаментальную вычислительную задачу отыскания Флоке-состояний квантовой частицы в пространственно-неоднородном периодически модулированном потенциале, возникающую, например, в исследовании динамики бозе-эйнштейновских конденсатов в оптических потенциалах [4]. Ее решение состоит в разработке численного метода, включающего в себя (i) генерацию пространственно-неоднородного потенциала с «гладким», коррелированным беспорядком, (ii) нахождение достаточно большого числа N собственных состояний стационарного гамильтониана и преобразование нестационарных динамических уравнений в связанный с ними базис, (iii) численное интегрирование полного набора N линейно-независимых состояний на периоде внешней модуляции T, получение $N \times N$ матрицы Флоке, (iv) отыскание собственных векторов и собственных чисел Флоке-матрицы. В связи высокой с вычислительной трудоемкостью задачи $O(N^2 T)$ выполнена параллельная реализация алгоритма, исследована ее масштабируемость с увеличением рассматриваемого числа состояний стационарного гамильтониана.

Численный метод

В одномерном случае динамика квантовой частицы во внешнем поле описывается уравнением Шредингера:

$$i\hbar \frac{\partial}{\partial t} \psi(x,t) = \hat{H}(x,t)\,\psi(x,t), \qquad (1)$$

с гамильтонианом

$$\hat{H}(x,t) = -\frac{\hbar^2}{2}\frac{\partial^2}{\partial x^2} + [1+\varepsilon(t)]V(x) = \hat{H}_0 + \varepsilon(t)V(x), \qquad (2)$$

где $\hat{H}_0 = -\frac{\hbar^2}{2}\frac{\partial^2}{\partial x^2} + V(x)$ – стационарный гамильтониан, $V(x)$ – пространственный потенциал, находящийся под воздействием периодической во времени модуляции вида $\varepsilon(t) = \varepsilon_0 \cos\omega t$ (здесь ε_0 – некоторая постоянная, $|\varepsilon_0| < 1$).

В данной работе нас интересовал случай пространственно-неоднородного периодического потенциала $V(x+2\pi) = V(x)$ с «гладким», кореллированным беспорядком. Для генерации подобного, так называемого «спекл»-потенциала мы использовали методику, представленную в работе [4].

В стационарном случае ($\varepsilon_0 = 0$) решение уравнения (1) сводится к проблеме собственных состояний гамильтониана $\hat{H}_0$. Диагонализация $\hat{H}_0$ позволяет получить набор N собственных состояний $\{\varphi_l\}$ и соответствующих им энергий $\{E_l\}$ (где $1 \le l \le N$), формирующих ортонормированный базис.

В случае $\varepsilon_0 \neq 0$ гамильтониан задачи становится зависимым от времени, и решение (1) чрезвычайно усложняется. Однако поскольку неавтономный гамильтониан периодичен во времени $\hat{H}(x,t+T) = \hat{H}(x,t)$, решение может быть получено путем диагонализации оператора эволюции $\hat{U}(T,0)$, который пропагандирует состояния системы на период модуляции $T = 2\pi/\omega$ [5,6]:

$$|\psi(T)\rangle = \hat{U}(T,0)|\psi(0)\rangle. \qquad (3)$$

Решение проблемы собственных значений оператора $\hat{U}(T,0)$ дает набор собственных функций $\{|\psi_n(t)\rangle\}$, которые удовлетворяют теореме Флоке $|\psi_\alpha(t)\rangle = \exp[-i(\mu_\alpha/T)t/\hbar]|\varphi_\alpha(t)\rangle$ [5,6]. Здесь $|\varphi_\alpha(t+T)\rangle = |\varphi_\alpha(t)\rangle$ являются состояниями Флоке, а μ_α – так называемыми «квазиэнергиями».

Учитывая вышеизложенное, будем искать решение (1) в базисе собственных состояний стационарной задачи:

$$\psi(x,t) = \sum_{l=1}^{N} a_l(t)\varphi_l. \qquad (4)$$

Подстановка (4) в (1) и использование свойства ортогональности собственных состояний $\{\varphi_l\}$ приводит уравнение (1) к следующему, записанному в матричной форме, виду:

$$\frac{\partial}{\partial t}\vec{a} = -\frac{i}{\hbar}\left[\mathbf{diag}E_l + \varepsilon(t)\Sigma\right]\vec{a}, \tag{5}$$

где $\vec{a} = \left(a_1,...,a_N\right)^T$, $\mathbf{diag}E_l$ – диагональная матрица, содержащая E_l на главной диагонали, Σ – матрица, состоящая из элементов $\sigma_{ml} = \left\langle \varphi_m \middle| V(x) \middle| \varphi_l \right\rangle$.

Правая часть динамического уравнения (5) состоит из двух операторов $\hat{A} = -\frac{i}{\hbar}\mathrm{diag}\,E_l$ и $\hat{B} = -\frac{i}{\hbar}\varepsilon(t)\Sigma$, что позволяет построить симплектическую схему интегрирования SABA-типа [7]. В данной работе для численного интегрирования был использован симплектический интегратор первого порядка SABA_1:

$$e^{(\hat{A}+\hat{B})\Delta t} \approx e^{\hat{A}\Delta t/2} e^{\hat{B}\Delta t} e^{\hat{A}\Delta t/2}. \tag{6}$$

Пропагация $\vec{a}$ на шаг интегрирования Δt осуществляется путем последовательного применения (6), где действие каждого оператора может быть записано в явном виде:

$$\begin{aligned} e^{\hat{A}\tau}\Big|_{\tau=\Delta t/2} &:= \vec{a}(t+\tau) = \mathrm{diag}\left(\exp\left[-\frac{i}{\hbar} E_l \tau \right] \right) \cdot \vec{a}(t), \\ e^{\hat{B}\tau}\Big|_{\tau=\Delta t} &:= \vec{a}(t+\tau) = \Lambda\,\mathrm{diag}\left(\exp\left[-\frac{i}{\hbar}\lambda_j \int_t^{t+\tau} \varepsilon(t')dt' \right] \right) \Lambda^{T^*} \cdot \vec{a}(t), \end{aligned} \tag{7}$$

где Λ – унитарная матрица, содержащая по столбцам собственные вектора матрицы Σ, Λ^{T^*} – эрмитово-сопряженная матрица Λ, λ_j – диагональная матрица, содержащая собственные числа Σ.

Используя SABA_1 (6) для интегрирования $1 \le l \le N$ произвольных начальных условий $a_{l'}^{(l)}(0) = \delta_{l,l'}$ на период T, мы составляем матрицу оператора $\hat{U}(T,0)$, l-й столбец которой задан $a_{l'}^{(l)}(T)$. Как уже было отмечено, диагонализация $\hat{U}(T,0)$ дает набор состояний Флоке и соответствующих им квазиэнергий.

Программная реализация численного метода

Базовая версия

Программная реализация метода выполнена на языке C++ с использованием библиотеки Intel MKL. Основные вычислительно трудоемкие участки кода написаны на языке C. Вычисления организованы следующим образом:

1. На первом шаге решается стационарная задача ($\varepsilon_0 = 0$). В результате диагонализации гамильтониана $\hat{H}_0$ мы получаем набор собственных состояний $\{\varphi_l\}$ и соответствующих им энергий $\{E_l\}$ (где $1 \le l \le N$), формирующих ортонормированный базис.

2. Далее задаются начальные условия в виде N векторов $a_{l'}^{(l)}(0) = \delta_{l,l'}$ (где $1 \le l, l' \le N$).

3. Для каждого из начальных векторов производится численное интегрирование на периоде T с применением (6) и (7) на каждом шаге метода.

4. Из полученных векторов составляется матрица оператора $\hat{U}(T,0)$, диагонализация которой дает набор состояний Флоке и соответствующих им квазиэнергий.

Диагонализация и трудоемкие матричные операции выполняются средствами MKL. В связи с тем, что при представляющих интерес параметрах задачи основные вычисления сосредоточены в данных функциях, для работы в системах с общей памятью с приемлемой эффективностью достаточно использовать параллельную версию MKL.

Параллельная версия для систем с распределенной памятью

Для решения больших задач разработана распределенная версия алгоритма. Распараллеливание основано на том, что основные вычисления сосредоточены на этапе 3, причем пересчет каждого вектора начальных условий на период интегрирования может быть выполнен независимо. Таким образом, применимо распараллеливание по данным с использованием технологии MPI. Этапы выполняются на одном узле, при этом диагонализация распараллеливается на уровне потоков в рамках библиотеки MKL.

Вычислительные эксперименты

Тестовая инфраструктура

В качестве тестовой системы использовался суперкомпьютер ННГУ «Лобачевский»[7]. Расчеты выполнялись под управлением операционной

[7] Суперкомпьютер ННГУ «Лобачевский»: URL – http://hpc-education.unn.ru/ru/ ресурсы.

системы семейства Linux (CentOS 6.4). В расчетах использовалось до 10 узлов кластера, на каждом узле установлены два процессора Intel Sandy Bridge E5-2660 2.2 GHz (8 ядер), 64GB RAM.

Экспериментальная проверка корректности реализации

Первый тест предназначен для проверки корректности реализации численного метода. Поскольку исследуемая система (1) является периодической, то для эволюции состояний Флоке должно выполняться соотношение:

$$x_j(kT) = e^{-i\mu_j kT/\hbar} x_j(0), \tag{8}$$

где j – номер состояния Флоке, μ_j – квазиэнергия, соответствующая j-му состоянию, T – период интегрирования, k – количество периодов интегрирования состояний Флоке.

Соотношение (8) используется как основа для теста. Состояния Флоке пересчитывались на 10 периодов. После каждого периода вычислялась величина

$$Err_1 = \left(x_j(kT)e^{-i\mu_j kT/\hbar}\right)^* \left(x_j(0)\right)^T. \tag{9}$$

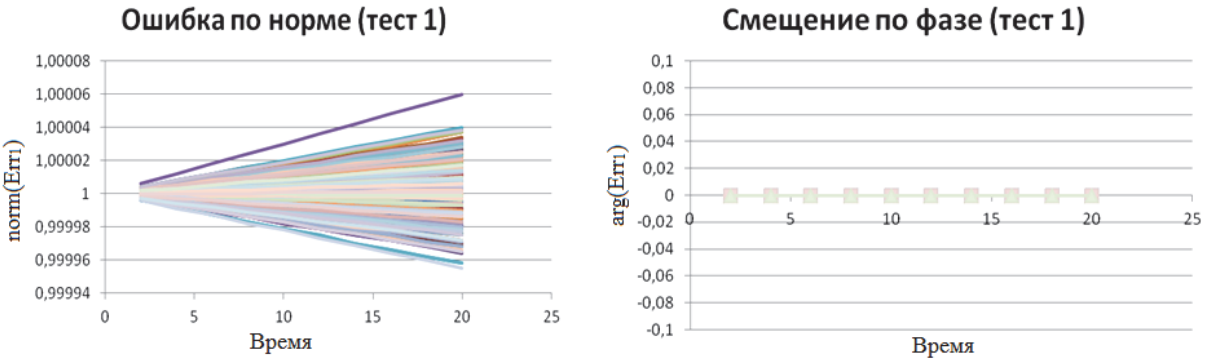


Рис. 1. Оценка точности (9) программной реализации алгоритма

При корректной работе метода норма выражения (9) должна быть близка к единице, а аргумент – к нулю. На рис. 1 представлены численные результаты первого теста для системы с параметрами $N = 201$, $T = 2$ и шагом интегрирования $\tau = T/300$. Как видим, результаты соответствуют теоретическим предсказаниям.

Экспериментальная проверка точности метода

Второй тест оценивал точность метода. Для этого рассматривалось влияние шага на получаемые результаты интегрирования. Для данной проверки в качестве начальных состояний были взяты состояния

Флоке. Интегрирование на один период (T = 2) проводилось с шагами τ и $\tau/2$. Во всех совпадающих по времени координатах считалась ошибка по норме и фазе с использованием выражения (10). Как и в первом тесте, в идеальном случае норма (10) должна быть близка к единице, а аргумент – к нулю:

$$Err_2 = \left(x_j(k\tau)\right)^* \left(x_j\left((2k)\tau/2\right)\right)^T. \qquad (10)$$

Тест был выполнен для различных шагов интегрирования $\tau = T / \mathrm{NSTEPS}$, где параметр NSTEPS варьировался от 10^2 до 10^4. Результаты теста показали, что при NSTEPS = 500 достигается norm(Err_2) = = 0,9996 и arg(Err_2) = 0,0836. При дальнейшем уменьшении шага интегрирования norm(Err_2) приближается к единице, а arg(Err_2) – к нулю, что позволяет достигнуть заданной точности метода.

Анализ масштабируемости

На рис. 2 представлены результаты вычислений для систем различной размерности N. Численное интегрирование выполнялось на один период $T = 2$ с шагом интегрирования $\tau = T / 1000$.

Эксперименты показали, что при увеличении размера задачи увеличивается и ускорение. Данный эффект связан с тем, что время, затраченное на пересчет векторов (этап 3), увеличивается.

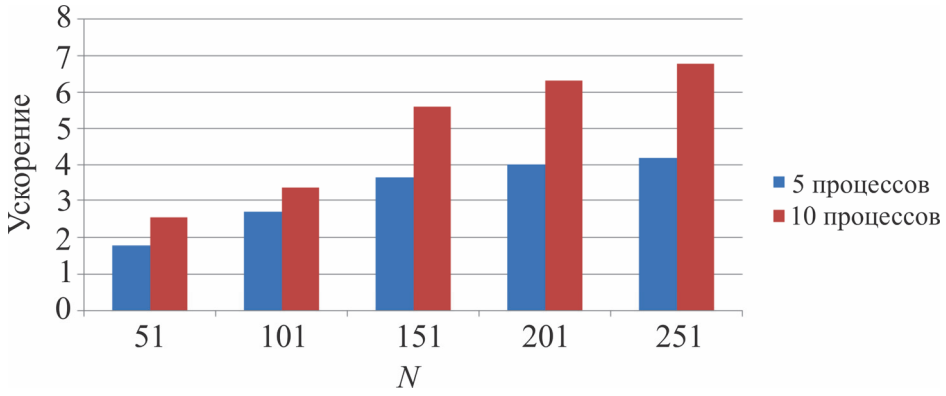


Рис. 2. Ускорение разработанной программной реализации алгоритма при разных размерах задачи

При этом затраты времени на сбор и обработку результатов становятся достаточно малыми по сравнению с общим объемом вычислений. На рис. 3 наглядно продемонстрировано изменение ускорения при разном количестве NSTEPS-шагов интегрирования на период в задаче размерности N = 201.

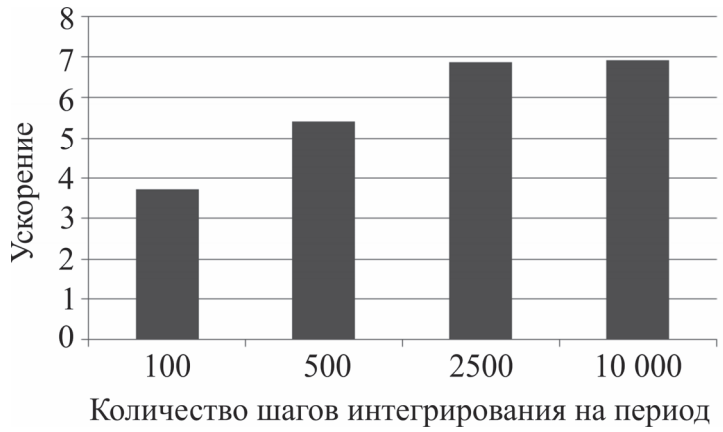


Рис. 3. Изменение ускорения при уменьшении шага интегрирования системы размерностью $N = 201$

Заключение

В работе предложен новый численный метод интегрирования математических моделей квантовых систем, находящихся под периодическим внешним воздействием, который позволяет корректно провести Флоке-анализ заданного числа низших состояний стационарного гамильтониана. Разработана параллельная программная реализация метода, проведен экспериментальный анализ корректности и точности. Эффективность параллельной версии при использовании 10 процессоров достигает 69 % в задаче размерности $N = 201$. Задача размерности $N = 2001$ решается за 406 с при 500 шагах на период.

Работа поддержана грантом Министерства образования и науки РФ, соглашение № 02.в.49.21.0003.

Список литературы

1. Lutz E., Renzoni F. Beyond Boltzmann–Gibbs statistical mechanics in optical lattices // Nature Physics. – 2013. – Vol. 9. – P. 615.

2. Poot M., van der Zant H.S.J. Mechanical systems in the quantum regimes // Phys. Rep. – 2012. – Vol. 511. – P. 273.

3. Aspelmeyer M., Kippenberg T.J., and Marquardt F. Cavity optomechanics. – URL: http://arxiv.org/abs/1303.0733v1. 2014.

4. Modugno M. Collective dynamics and expansion of a Bose-Einstein condensate in a random potential // Phys. Rev. A. – 2006. – Vol. 73. – P. 013606.

5. Denisov S., Morales-Molina L., Flach S., Hanggi P. Periodically driven quantum ratchets: Symmetries and resonances // Phys. Rev. A. – 2007. – Vol. 75. – P. 063424.

6. Grifoni M., Hanggi P. Driven Quantum Tunneling // Phys. Rep. 1998. – Vol. 304. – P. 229.

7. Laskar J., Robutel P. High order symplectic integrators for perturbed Hamiltonian systems // Celest. Mech. Dyn. Astron. – 2001. – Vol. 80. – P. 39.

А.В. Коржиманов[1,2], М. Марклунд[3]

[1] Институт прикладной физики Российской академии наук, Н. Новгород
[2] Нижегородский государственный университет им. Н.И. Лобачевского,
[3] Департамент прикладной физики Технического университета Чалмерса, Гётеберг, Швеция

МОДЕЛИРОВАНИЕ ПРОЦЕССА ЛАЗЕРНОГО УСКОРЕНИЯ ИОНОВ В МИШЕНЯХ С СУБВОЛНОВЫМИ НЕОДНОРОДНОСТЯМИ ПОВЕРХНОСТИ

Проведено численное моделирование процесса ускорения ионов сверхмощным лазерным импульсом. Показано, что использование мишеней с нанесёнными на облучаемую поверхность неоднородностями субволнового размера приводит к увеличению эффективности ускорения. Определены и теоретически обоснованны оптимальные параметры искусственных неоднородностей. Показано, что параметры, при которых достигается максимальная полная энергия ионного пучка, максимальная энергия отдельного иона и максимум поглощения значительно отличаются друг от друга.

Введение

Современные технологии получения лазерных импульсов сверхвысокой пиковой мощности и интенсивности переживают период активного роста. Освоен уровень одного петаватта, активно разрабатываются системы, позволяющие получить до 10 ПВт, обсуждаются возможности достижения сотен петаватт и даже экзаватта [1]. Взаимодействие излучения таких систем с веществом носит сложный нелинейный характер, требующий для адекватного описания самосогласованного учёта влияния как излучения на вещество, так и вещества на излучение [2]. Это делает численное моделирование главным инструментом теоретического исследования в данной области физики.

Одним из наиболее важных приложений сверхмощных лазерных систем является создание источников пучков высокоэнергичных ионов [3, 4]. Такие пучки с энергиями ионов от нескольких десятков до сотен МэВ востребованы в целом ряде отраслей: в терапии раковых опухолей, мониторинге ядерных отходов, поджиге термоядерных мишеней. На сегодняшний день экспериментально продемонстрирована генерация протонов с энер-

гиями до 100 МэВ [5]. Однако требуется как дальнейшее увеличение их максимальной энергии, так и увеличение количества ускоренных частиц. Обе эти задачи требуют для своего решения увеличения эффективности преобразования лазерной энергии в кинетическую энергию ионов.

Наиболее эффективной при относительно невысоких мощностях (интенсивностях) лазерного излучения является схема ускорения ионов, известная как TNSA [6]. Её эффективность, однако, обычно не превышает нескольких процентов. Это связано с тем, что в схеме происходит многократная конверсия энергии. Сначала лазерный импульс, облучающий тонкую металлическую фольгу, нагревает электроны, затем электроны разлетаются и, вылетая с обратной стороны, создают квазиэлектростатическое поле, которое уже, в свою очередь, ускоряет ионы, расположенные на задней стороне мишени. На каждом из этапов взаимодействия происходит потеря энергии, поэтому стоит задача оптимизации каждого из этих этапов.

Одним из методов увеличения эффективности схемы TNSA является увеличение доли поглощаемой электронами лазерной энергии за счёт модификации облучаемой поверхности [7]. В частности, в данной работе мы исследуем влияние на эффективность ускорения нанесённых на облучаемую поверхность субволновых неоднородностей.

Мишень

В качестве мишени была исследована золотая фольга толщиной 1 мкм, на заднюю поверхность которой был нанесён слой водорода толщиной 10 нм. Как было показано ранее, такая конфигурация позволяет получать квазимоноэнергетические спектры ускоренных протонов.

На облучаемую поверхность наносились неоднородности в форме периодической прямоугольной «щётки». Поскольку ранее было показано, что конкретная форма неоднородностей не оказывает существенного влияния на результат, мы не рассматривали неоднородности в виде сфер или треугольников. В то же время использованная нами форма неоднородностей позволяло независимо регулировать сразу три параметра: толщину одного элемента w, период «щётки» p и высоту h.

Моделирование

Для численного моделирования процесса взаимодействия применялся традиционный для подобных задач метод частиц в ячейках, реализованный в программном комплексе PICADOR. Данный комплекс разработан в ННГУ и предоставляет возможность производить расчёты на распределённых вычислительных системах с гибридной архитектурой, то есть задейст-

вуя как обычные CPU-процессоры, так и графические GPU и другие сопроцессоры (например, Intel Xeon Phi) [8–10].

Моделирование проводилось на двумерной сетке. Размеры сетки соответствовали физическим размерам 16×16 мкм. Шаг сетки равнялся 1/25 длины волны лазерного излучения. На одну ячейку приходилось в начале расчёта 20 частиц, моделирующих вещество в плазменном состоянии. Параметры лазерного излучения были взяты близкими к реальным системам, на которых проводятся соответствующие эксперименты: длительность 42 фс, энергия 1 Дж, длина волны 800 нм, диаметр пятна фокусировки 8 мкм.

По результатам расчётов анализировались распределения различных частиц (электронов, протонов, ионов золота) по энергиям, а также полная энергия этих частиц, а также электромагнитных полей.

Оптимальная толщина

Численное моделирование показало, что все характеристики, по которым имеет смысл проводить оптимизацию, достигают оптимальных значений для одной и той же конкретной толщины w. Эта толщина зависит от плотности плазмы и по порядку величины совпадает с толщиной скин-слоя. В частности, для использованной в большинстве расчётов плотности плазмы, равной 30 критических плотностей, оптимальная толщина равнялась 150 нм.

Наличие единой оптимальной толщины и её связь со скин-слоем объясняются тем фактом, что с точки зрения поглощения лазерной энергии при меньшей толщине уменьшается количество электронов, взаимодействующих с излучением, а при большей – увеличивается количество ионов золота, являющихся дополнительным каналом утечки полезной энергии.

Оптимизация по периоду и высоте

В дальнейших расчётах толщина w была зафиксирована на оптимальном значении и происходила вариация двух других параметров неоднородностей – периода p и высоты h.

На рис. 1 приведена зависимость доли полной поглощённой энергии по отношения к полной энергии падающего импульса. Видно, что эта доля непрерывно растёт с ростом высоты неоднородностей и имеет некий оптимум в районе 1–1,5 длин волн по периоду. Это объясняется тем, что для данных периодов излучение свободно проходит в поверхностную структуру, а за счёт увеличения её высоты можно неограниченно увеличивать время взаимодействия излучения с веществом, в результате чего всё большая и большая часть лазерной энергии конвертируется в энергию плазмы.

Однако если сравнить диаграмму для поглощённой энергии с диаграммой для доли энергии, переданной в протоны (рис. 2), то можно увидеть, что они кардинально различаются. Максимум энергии, переданной в протоны, также достигается для периода структуры, равной 0,9 длине волны, однако оптимальная высота составляет только 0,4 длины волны, а для неограниченно высоких структур доля энергии, переданной в протоны, стремится к нулю.

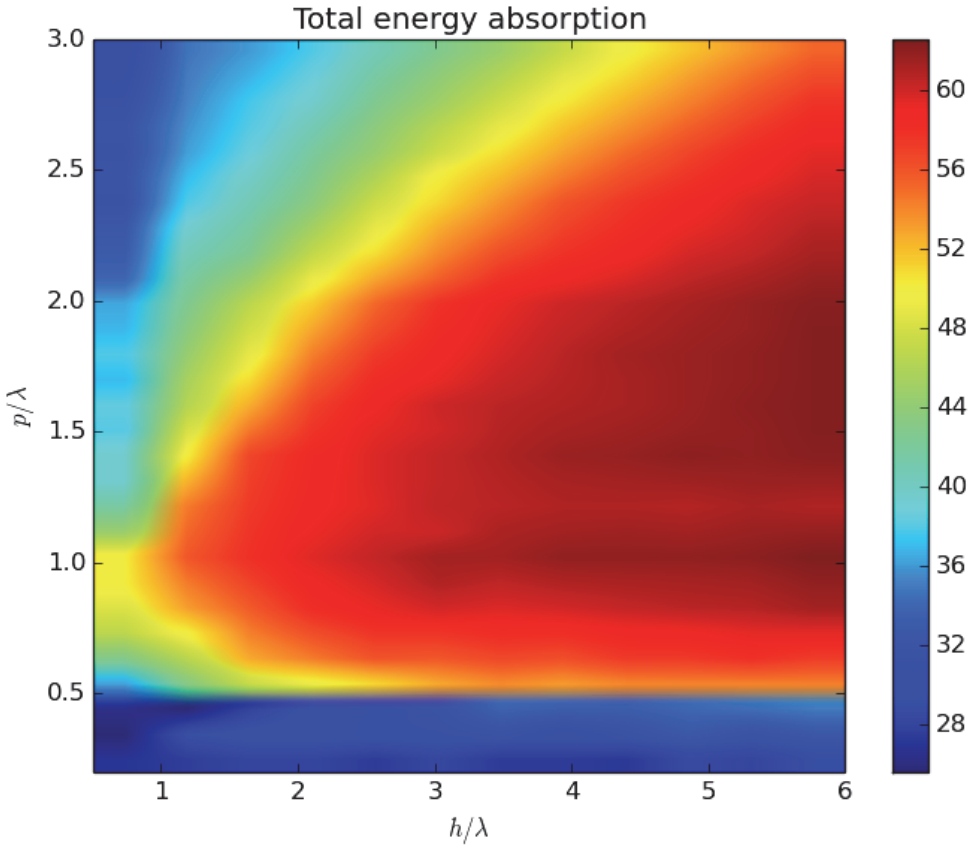


Рис. 1. Зависимость доли полной поглощённой энергии от высоты h и периода p искусственных неоднородностей

Это объясняется тем очевидным обстоятельством, что при больших значениях h энергия поглощается в области, далёкой от области локализации протонов. В результате происходит её конвертация в энергию ионов золота, разлетающихся преимущественно перпендикулярно направлению распространения лазерного импульса.

Выводы

В данной работе нами независимо подтверждено, что эффективность схемы лазерного ускорения ионов TNSA может быть увеличена в 4–5 раз за счёт нанесения на облучаемую поверхность мишени регулярных неоднородностей субволнового размера.

Путём численного моделирования были определены параметры неоднородностей, при которых достигаются максимальное поглощение лазерного излучения и максимальная конвертация энергии в энергию ускоряемых ионов. Продемонстрировано, что оптимальные параметры для этих двух характеристик отличаются друг от друга кардинально, в то время как поглощение максимально для максимально высоких неоднородностей, максимум полной энергии ионов достигается для высоты порядка полуволны.

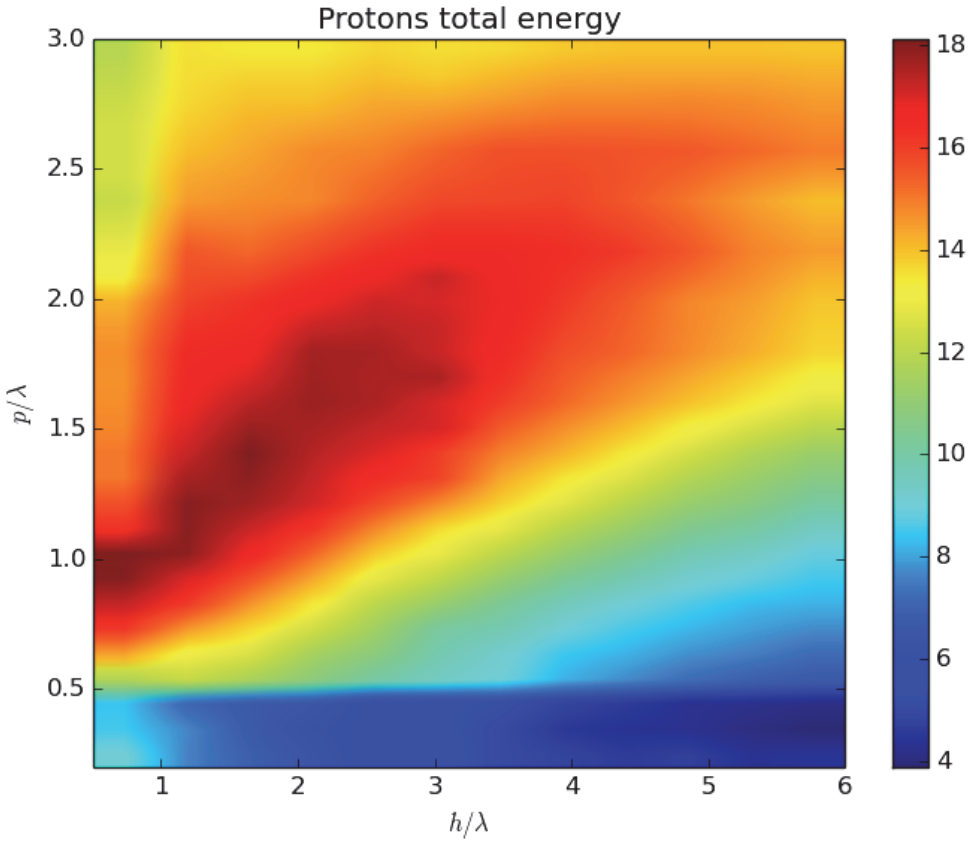


Рис. 2. Зависимость доли полной энергии, переданной протонам, от высоты *h* и периода *p* искусственных неоднородностей

Авторы работы благодарны А.А. Гоноскову и Г. Вальстрёму за продуктивные обсуждения изложенных идей и результатов, а также И.Б. Мееров у и его группе в ННГУ за разработку использованных при моделировании численных программ.

Численное моделирование проводилось на суперкомпьютерах Межведомственного суперкомпьютерного центра РАН и Шведской национальной вычислительной инфраструктуры SNIC.

Работа частично поддержана грантом (соглашение от 27 августа 2013 г. № 02.B.49.21.0003 между Министерством образования и науки РФ и Нижегородским государственным университетом).

Список литературы

1. Горизонта петаваттных лазерных комплексов / А.В. Коржиманов, А.А. Гоносков, Е.А. Хазанов, А.М. Сергеев // Успехи физических наук. – 2011. – Т. 181, № 1. – С. 9.

2. Mourou G.A., Tajima T., Bulanov S.V. Optics in the relativistic regime // Reviews of modern physics. – 2009. – Vol. 78, № 2. – P. 309.

3. Daido H., Nishiuchi M., Pirozhkov A. S. Review of laser-driven ion sources and their applications // Reports on Progress in Physics. – 2012. – Vol. 75. – P. 056401.

4. Macchi A., Borghesi M., Passoni M. Ion acceleration by superintense laser-plasma interaction // Reviews of Modern Physics. – 2013. – Vol. 85. – P. 751.

5. 160 MeV laser-accelerated protons from CH2 nano-targets for proton cancer therapy / B.M. Hegelich [et al.]. – arXiv:1310.8650.

6. Energetic proton generation in ultra-intense laser-solid interactions / S.C. Wilks [et al.] // Physics of Plasmas. – 2001. – Vol. 8. – P. 542.

7. Improvement of energy-conversion eficciency from laser to proton beam in a laser-foil interaction / Y. Nodera [et al.] // Physical Review E. – 2008. – Vol. 78. – P. 046401.

8. Particle-in-cell plasma simulation on heterogeneous cluster systems / S. Bastrakov [et al.] // Journal of Computaional Science. – 2012. – Vol. 3. – P. 474.

9. Моделирование плазмы методом частиц в ячейках с использование сопроцессоров Intel Xeon Phi / И.А. Сурмин, С.И. Бастраков, А.А. Гоносков, Е.С. Ефименко, И.Б. Мееров // Вычислительные методы и программирование. – 2014. – Т. 15. – С. 530.

10. Particle-in-Cell Simulation on CPUs, GPU sans Xeon Phi Coprocessors / S. Bastrakov, I. Meyerov, I. Surmin, E. Efimenko, A. Gonskov, A. Malyshev, M. Shiryaev // Lecure Notes in Computer Science. – Springer, Heidelberg, 2014. – Vol. 8488. – P. 513.

Е.В. Коромыслов[1,2,3], М.В. Усанин[1], А.А. Синер[1,3], Л.Ю. Гомзиков[1]

[1] ОАО «Авиадвигатель», Пермь
[2] Институт механики сплошных сред УрО РАН, Пермь
[3] Пермский государственный национальный исследовательский университет

РАСЧЕТ ЗАДАЧ АВИАЦИОННОГО ДВИГАТЕЛЕСТРОЕНИЯ МЕТОДАМИ ВЫСОКОГО ПОРЯДКА НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Описывается параллельный газодинамический решатель для графических процессоров (ГПУ), основанный на схемах высокого порядка точности. Также приводятся различные примеры задач из области авиадвигателестроения, расчет которых можно производить в данном решателе.

Введение

В настоящее время инженерные газодинамические расчёты проводятся, в подавляющем большинстве, схемами низкого (до 2-го) порядка точности. Также для них в основном используются осреднённые по Рейнольдсу уравнения Навье–Стокса, а сами расчёты проводятся в стационарных постановках с использованием секторов периодичности, допускаемых конструкцией. Такая постановка задач в контексте турбомашин и авиадвигателестроения позволяет определить многие важные характеристики двигателя (потери давления, КПД, тягу и т.д). Тем не менее при проектировании авиационных двигателей (особенно это касается современных двигателей нового поколения) стоят и другие важные задачи, например, определение акустических характеристик, а также динамической прочности конструкций. Данные задачи в общем случае могут быть решены только с применением нестационарных расчётов конструкции в целом, без использования периодичности. При этом необходимые вычислительные ресурсы значительно увеличиваются, а время расчёта начинает составлять недели и месяцы (например, примерно месяц для расчёта шума вентиляторной ступени двигателя). Более того, для расчётов генерации и распространения звука требуются модели с использованием метода крупных вихрей (LES).

Для проведения таких нестационарных расчётов методами высокого порядка точности за приемлемое время на ОАО «Авиадвигатель» создаётся решатель GHOST CFD (GPU-oriented High Order STructured). GHOST CFD позволяет решать нестационарные газодинамические задачи в областях со сложной геометрией, а также при наличии относительного движения объектов в расчётной области (такого, как вращение лопаток ротора турбомашин относительно лопаток статора) с помощью перекрывающихся

(«CHIMERA») сеток. Кроме того, поддерживаются расчёты многокомпонентных и трансзвуковых течений газа. В решателе используются конечно-разностные схемы высокого порядка точности и высокого разрешения (схемы типа DRP [1,2]). Данные схемы за счёт применяемой в них селективной фильтрации также реализуют метод крупных вихрей (так называемый LES-RF [3,4]).

Все основные расчётные процедуры GHOST CFD реализованы в виде ядер на CUDA C. Также имеется возможность расчёта на нескольких ГПУ, расположенных на различных узлах кластера с одновременным выполнением расчётных процедур и передачей данных между ГПУ с помощью GPU Direct, что позволяет увеличить эффективность распараллеливания.

Необходимо отметить, что количество ячеек расчётной сетки на 1 Гб оперативной памяти ГПУ составляет порядка 1–2 млн. Это достигается минимизаций хранимых данных: хранятся только значения неизвестных полей давления, плотности и скорости, а также метрические коэффициенты расчётной сетки и параметры точек. Все остальные необходимые данные, такие как производные от полей, находятся непосредственно в течение расчёта шага по времени. Таким образом, даже на одном ГПУ можно решать задачи с достаточно подробной расчётной сеткой. При этом абсолютная скорость расчёта составляет порядка 1 шага по времени в секунду (для задач без движения сетки) с расчётной сеткой ~10 млн ячеек. В качестве особенности реализации решателя стоит также отметить принцип расчёта производных от полей неизвестных. Данная процедура производится в 2 этапа. На первом расчёт производится по логическим индексам i и j блока структурированной сетки, на втором – по индексу k. При данном подходе обеспечивается объединённый (coalesced) доступ к памяти ГПУ, что является одним из требований эффективной реализации расчётов на CUDA. Как отмечалось ранее, расчёт полей на новом шаге по времени происходит одновременно с расчётом производных.

Ускорение расчёта в решателе составляет порядка 12–20 раз на 1 ГПУ Nvidia Tesla M2090 по сравнению с 8 ядрами Intel Xeon E5-2680. График ускорения и масштабируемости расчёта для тестовой задачи (4 блока расчётной сетки, 4 млн ячеек) представлен на рис. 1.

В качестве тестовых задач использовались расчёт генерация шума модельного дозвукового сопла, расчёты до- и трансзвукового обтекания стандартных и спроектированных на ОАО «Авиадвигатель» лопаточных профилей. На данных тестах результаты, полученные в GHOST CFD, достаточно хорошо согласовались с экспериментом и расчётами в других пакетах (ANSYS CFX\Fluent).

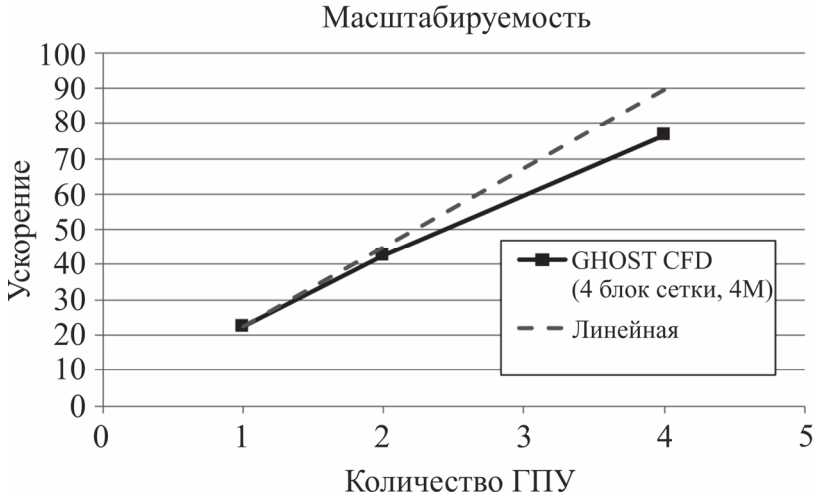


Рис. 1. Масштабируемость решателя GHOST CFD на несколько ГПУ (ускорение представлено для Nvidia Tesla M2090 по сравнению с 8 ядрами Intel Xeon E5-2680)

Результаты расчёта звукового давления в различных точках вдали от сопла представлены на рис. 2. Данные результаты были получены методом акустической аналогии [5].

Размер расчётной сетки для GHOST CFD составлял порядка 12 млн ячеек, время расчёта – порядка недели на трех ГПУ. Для ANSYS Fluent сетка составляла 25 млн ячеек, время расчёта – более 2 месяцев на 28 ядрах AMD Opteron.

На рис. 3 представлены результаты расчёта обтекания лопаточного профиля стандартной конфигурации STCF4. На рисунке видно, что в данном расчёте есть области, в которых число Маха больше единицы, т.е. течение трансзвуковое. Для STCF4 размер сетки составлял порядка 11 млн ячеек.

На данный момент ведутся работы по применению решателя в реальной инженерной практике на ОАО «Авиадвигатель». Первой задачей в этом направлении стал расчёт шума, генерируемого вентилятором авиационного двигателя.

В упрощённой постановке данный расчёт требует сетки порядка 30–40 млн ячеек. При этом расчёт одного оборота вентилятора на четырех ГПУ занимает около 1,5–2 суток. Аналогичное время для постановки с ~65 млн ячеек в ANSYS CFX на десяти узлах кластера (160 ядер Intel Xeon E5-2680) составляет порядка 8 суток. Для дальнейшего определения шума вентилятора нужно провести расчёт трех оборотов, что составляет около 1 недели для GHOST CFD и 3 недели с помощью ANSYS CFX в описанных постановках.

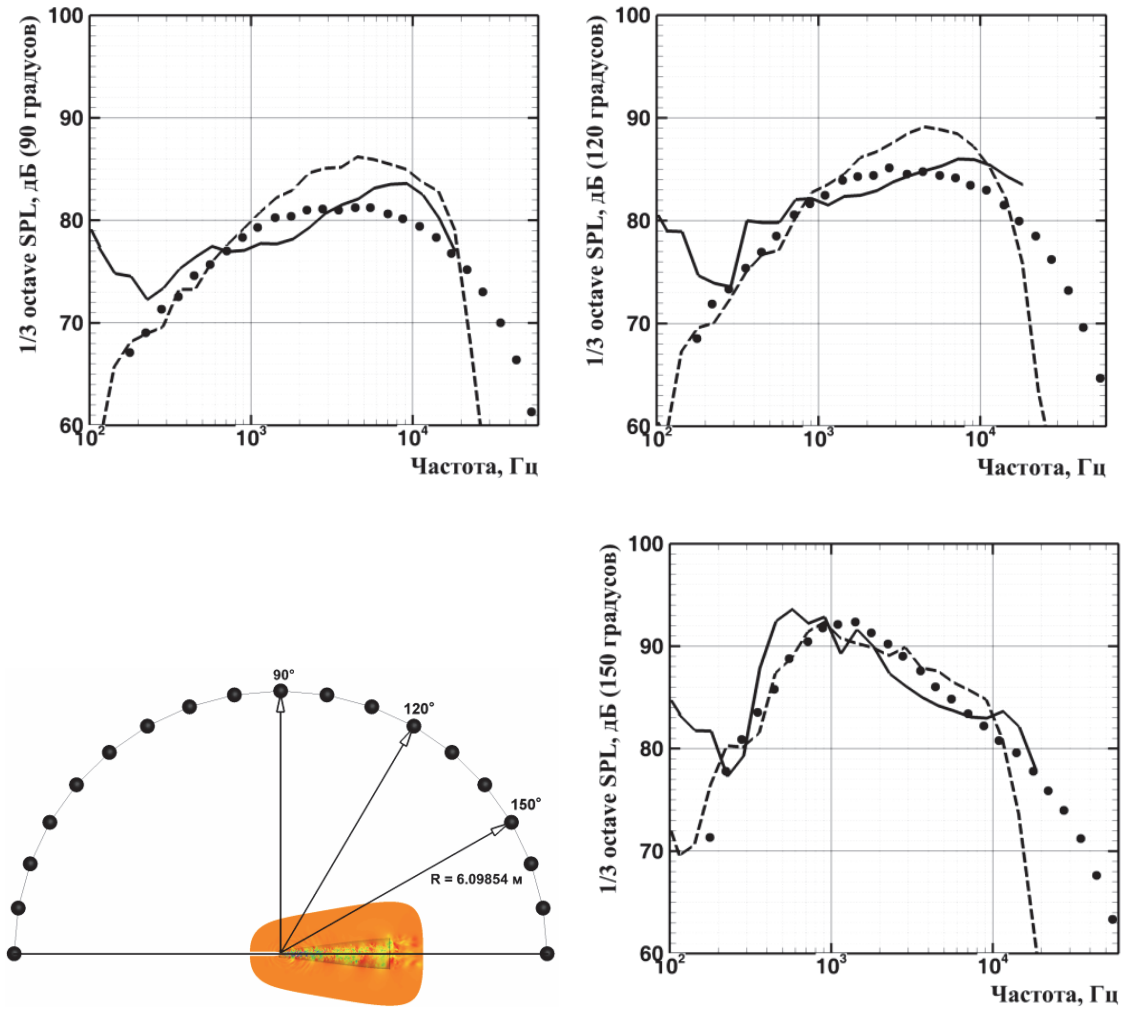


Рис. 2. Точки наблюдения и соответствующие им звуковые давления для конического сопла: • – эксперимент, — – GHOST CFD, --- – ANSYS Fluent

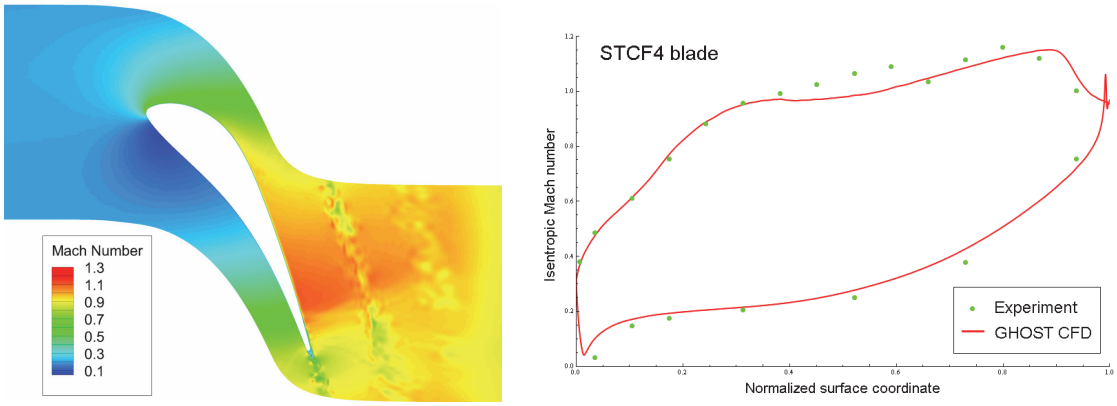
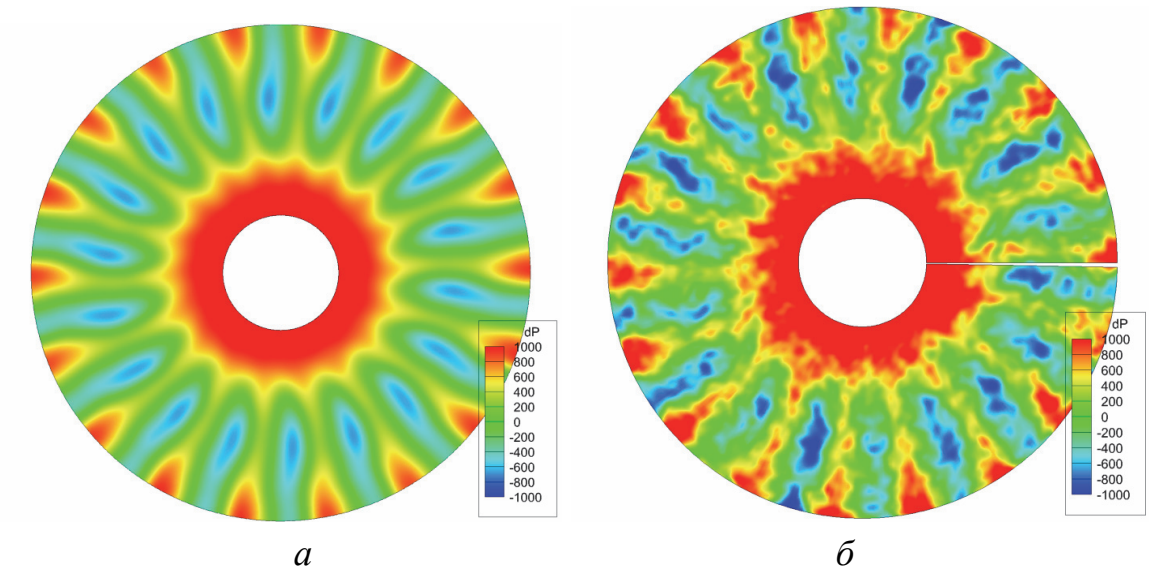


Рис. 3. Мгновенное распределения числа Маха в расчётной области и изэнтропического числа Маха на поверхности лопатки для STCF4

На рис. 4 представлены типичные мгновенные распределения давления для GHOST CFD и ANSYS CFX. Из рисунка видно различие метода крупных вихрей (LES), используемого в GHOST CFD, и нестационарных

уравнений Рейнольдса (uRANS), используемых в ANSYS CFX. В силу аппроксимации высокого порядка и метода крупных вихрей, GHOST CFD показывает значительно более неоднородное течение.



а *б*

Рис. 4. Мгновенное распределение давления вблизи вентилятора для а) ANSYS CFX; б) GHOST CFD

В дальнейшем планируется реализация метода отсоединённых вихрей (DES), позволящего более эффективно проводить моделирование турбулентности, а также расширение круга решаемых задач и продолжение верификация решателя.

Список литературы

1. Berland J., Bogey C., Bailly C., Optimized explicit schemes: matching and boundary schemes and 4th-order Runge-Kutta algorithm // AIAA Paper. – 2004. – P. 2004–2814.

2. Bogey C., Bailly C., A family of low dispersive and low dissipative explicit schemes for computing the aerodynamic noise // AIAA Paper. – 2002. – P. 2002–2509.

3. Bogey C., Bailly C., Computation of a high Reynolds number jet and its radiated noise using large eddy simulation based on explicit filtering // Computers & Fluids. – 2006. – Vol. 35. – P. 1344–1358.

4. Bogey C., Bailly C. Large eddy simulations of round free jets using explicit filtering with/without dynamic Smagorinsky model, International // Journal of Heat and Fluid Flow. – 2006. – Vol. 27. – P. 603–610.

5. Prieur J., Rahier G. Aeroacoustic Integral Methods, Formulation and Efficient Numerical Implementation // Aerospace Science and Technology. – 2001. – № 5. – P. 457–468.

А.Г. Коротченко, В.М. Сморякова

Нижегородский государственный университет им. Н.И. Лобачевского

О ПОСТРОЕНИИ ФОРМУЛ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ

В случае решения систем дифференциальных уравнений больших порядков узким местом в реализации алгоритма становится необходимость многократного решения системы линейных алгебраических уравнений. Поэтому целесообразным является применение параллельных методов решения систем линейных уравнений. В работе рассматривается использование различных схем параллельных вычислений систем линейных алгебраических уравнений, приводятся результаты вычислительного эксперимента.

При построении адекватных математических моделей физических объектов часто возникает явление жёсткости. В системах обыкновенных дифференциальных уравнений, описывающих данные модели, решение разбивается на «медленные» и «быстрые» составляющие. Поэтому построение формул численного интегрирования таких систем приводит к задаче выбора их параметров таким образом, чтобы минимизировать число вычислений правых частей системы при ограничениях, определяемых точностью вычислений. Данная проблема не теряет актуальности и при использовании высокопроизводительных вычислительных систем.

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений:

$$Y' = B(x)Y + b(x), Y(x_0) = Y_0, \qquad (1)$$

$$Y = Y(x) = (y^1(x),...,y^n(x)),\ x_0 \le x \le t,$$

где $B(x)$ – матрица размера $n \times n$, а $b(x)$ – вектор размерности n.

Будем предполагать, что решение задачи (1) $Y = Y(x)$ единственно и трижды непрерывно дифференцируемо на отрезке $[x_0, t]$. Здесь значение t неизвестно априори и определяется в процессе интегрирования системы. Для определения численных значений $Y_i = Y(x_i)$, $i = 1,2...$ используем следующую конечно разностную формулу:

$$Y_i = a_{i-2}Y_{i-2} + a_{i-1}Y_{i-1} + h_i a_i q_i, \qquad (2)$$

где

$$q_i = B(x_i)Y_i + b(x_i)\,, Y_i = (y_i^1, y_i^2, ..., y_i^n)\,,$$

$$a_{i-2} = -h_{i-1}^{-1} d_i^{-1} h_i^2\,,\ a_{i-1} = h_{i-1}^{-1} d_i^{-1} (h_{i-1} + h_i)^2\,, \tag{3}$$

$$a_i = d_i^{-1}(h_{i-1} + h_i)\,,\ d_i = h_{i-1} + 2h_i\,.$$

Будем считать, что система (1) удовлетворяет условиям, при выполнении которых вектор $Y = (y_1^i, ..., y_n^i)$ может быть определён из решения системы линейных уравнений

$$C_i Y_i = c_i\,, \tag{4}$$

где $C_i = E - h_i a_i B(x_i)\,,\ c_i = a_{i-2}Y_{i-2} + a_{i-1}Y_{i-1} + a_{i-1}Y_{i-1} + h_i a_i b(x_i)\,,\ E$ – единичная матрица.

В качестве основной характеристики формулы (1) будем рассматривать локальную ошибку, получаемую на *i*-м шаге для каждой *j*-й компоненты решения $Y_i = (y_i^1, ..., y_i^n)$ для формулы (2):

$$R_i^j = -\frac{h_i^4 + 2h_{i-1}h_i^3 + h_{i-1}^2 h_i^2}{2(6h_i + 3h_{i-1})} (y'''(\Theta_i))^j\,, \tag{5}$$

где $x_{i-1} - h_{i-1} \le \Theta_i \le x_{i-1} + h_i$ и $(y'''(\Theta_i))^j$ – значение третьей производной от *j*-й компоненты вектора решения $Y(x)$ в точке Θ_i. Потребуем, чтобы в процессе интегрирования системы выполнялись условия:

$$\max_{1 \le j \le n} | R_i^j | \le \varepsilon_i\,,$$

где ε_i – заданная точность на i-м шаге интегрирования.

Пусть третья производная вектора $Y(x)$ для формулы (2) удовлетворяет на отрезке $[x_0, x_0 + z]$ условию:

$$\max_{1 \le j \le n} | y'''(x)^j | \le K\,,$$

где $z > 0$ – некоторое число, K – вещественная константа.

Тогда получаем, что ограничение на формулу (2), обусловленное точностью вычислений на i-м шаге интегрирования, сводится к выполнению неравенства

$$\varphi_i(h_{i-1}, h_i) = h_i^4 + 2h_{i-1}h_i^3 + h_{i-1}^2 h_i^2 - 6\Delta_i h_i - 3\Delta_i h_{i-1} \le 0. \tag{6}$$

Таким образом, если $t \le x_0 + z$, t – момент окончания процесса интегрирования, то конкретный алгоритм численного решения задачи (1) со-

стоит в использовании разностной формулы (2), где шаги интегрирования должны удовлетворять соотношениям (6).

Теперь можно поставить задачу о выборе оптимального шага численного интегрирования для формулы (2) как задачу математического программирования:

$$\begin{aligned} &Q_m(H^m) = \sum_{i=1}^{m} h_i \to \max, \\ &\varphi_i(h_{i-1}, h_i) \le 0,\ h_i > 0,\ i = 1,...,m, \\ &H_m = (h_1,...,h_m), m = 1,2,.... \end{aligned} \tag{7}$$

Здесь h_0 равно некоторому неизвестному значению $\overline{h}_0$ ($\overline{h}_0 > 0$).

Наряду с данной задачей рассмотрим частные задачи:

$$\begin{aligned} &h_1 \to \max, \\ &\varphi_1(h_0, h_1) \le 0,\ h_1 > 0, \end{aligned} \tag{8}$$

где h_0 равно некоторому неизвестному $\overline{h}_0$ ($\overline{h}_0 > 0$),

$$\begin{aligned} &h_i \to \max, \\ &\varphi_i(h_{i-1}, h_i) \le 0,\ h_i > 0, \end{aligned} \tag{9}$$

$$h_{i-1} \in D_{i-1} = \{h \mid \varphi_{i-1}(h_{i-2}, h) \le 0\},\ i = 2,...,m.$$

Пусть h_1' – решение задачи (8) для фиксированного h_0 (при условии его существования). Пусть также h_i' – решение задачи (9) при $h_{i-1} = h_{i-1}'$, $j = 2,...,m$ (при условии его существования). Показано, что решением задачи (7) является кортеж $< h_1', h_2', ..., h_m' >$ для любого $m = 1,2,....$ Таким образом, решение задачи (7) можно получить последовательным решением частных задач (8), (9). Указанный факт позволяет предложить алгоритм численного интегрирования, минимизирующий число вычислений правых частей рассматриваемой системы дифференциальных уравнений при использовании конечно разностной формулы (2).

Список литературы

1. Korotchenko A.G., Smoryakova V.M. Multistage mathematical programming problems // Numerical Computations: Theory and Algorithms. – 2013. – 88 p.

2. Коротченко А.Г., Сморякова В.М. Об одном классе задач, имеющих многоэтапный характер // Проблемы теоретической кибернетики: материалы XVII междунар. конф.; Казань, 16–20 июня 2014 г. – Казань, 2014. – С. 146.

М.А. Кривов[1,2], М.Н. Притула[1], П.С. Иванов[2]

[1]ООО «ТТГ "Лабс"», г. Москва
[2]Московский государственный университет им. М.В. Ломоносова

ОПЫТ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ АВТОАДАПТАЦИИ ПРОГРАММ ДЛЯ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ ЭЙЛЕРА НА GPU-КЛАСТЕРЕ

Рассматривается возможность применения парадигмы автоадаптации программ для проведения расчётов на GPU-кластерах. С использованием разработанной авторами библиотеки TTG Apptimizer произведена оптимизация тестовой MPI-программы, осуществляющей моделирование двумерного потока несжимаемого газа. Выполнено тестирование подготовленной программной реализации на GPU-кластере из 20 узлов, в соответствии с которым ускорение от автоадаптации MPI-программы оказалось равным порядка 1,7–2,0 раз.

Введение

При разработке программ, которые планируется использовать на разнообразных GPU-кластерах, требуется достаточно основательно подходить к вопросам оптимизации, так как зачастую выполнение достаточно простых и очевидных модификаций с целью учёта особенностей используемого кластера может заметно повысить итоговую производительность. Обусловлено это тем, что у кластерных систем на базе GPU можно выделить достаточно большое количество характеристик, в зависимости от значений которых узким местом в программе становятся совершенно разные и порой неожиданные части.

В качестве простого примера можно привести две практически идентичные с точки зрения программы системы — облачный кластер Amazon EC2 из узлов типа G2.2XLarge и суперкомпьютер «Ломоносов». В каждой из систем используются узлы с очень близкими моделями графических и центральных процессоров (и там, и там по два CPU и GPU на узел), однако из-за иного межсетевого соединения и дополнительного слоя виртуализации при выполнении вычислений на кластере Amazon EC2 узким местом вместо самих вычислений с большой вероятностью станут операции барьерных синхронизаций. Если же при переходе от одной кластерной системы к другой отличий окажется ещё больше (например, разные версии шины PCI-E, иное количество ускорителей, вычислители с разной архитектурой, разная пропускная способность памяти), то скорость работы программы вполне может оказаться заметно ниже ожидаемой.

Для решения подобных проблем применяются технологии автоадаптации программ, получивших известность под термином autotuning. Суть

их заключается в вынесении некоторых параметров программ, влияющих на их производительность, во внешнюю сущность, которая перед запуском или в процессе вычислений может подстраивать уже скомпилированную программу под «доставшуюся» систему. В качестве наиболее наглядных примеров реализации данной идеи можно привести пакет для выполнения матричных операций ViennaCL [1], в котором имеется специальный модуль для адаптации вычислительных ядер под графический ускоритель, а также работы авторов данной статьи по разработке библиотеки TTG Apptimizer [2], в рамках которых, в частности, был оценен эффект от динамической оптимизации программ при решении модельных CFD-задач.

Имеется также множество и других примеров явной и неявной реализации данной идеи, например [3, 4], однако общим их недостатком является ориентированность на одиночные системы, а не на кластеры, в результате чего достигаемый эффект зачастую ограничивается ускорением в рамках одного GPU. Поэтому в данной работе предпринимается попытка расширить сферу применения технологий автоадаптации программ на кластерные системы, для чего оценивается возможный эффект от их использования при оптимизации MPI-программы по обсчёту обтекания двумерных объектов потоком невязкого газа.

Постановка задачи

В рамках задачи, на которой в дальнейшем будет проводиться оценка качества оптимизации программной реализации, требуется численно решить двумерную систему уравнений Эйлера в смешанной постановке. Сама система может быть записана следующим образом:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho V_x)}{\partial x} + \frac{\partial (\rho V_y)}{\partial y} = 0,0,$$

$$\frac{\partial \rho V_x}{\partial t} + \frac{\partial (\rho V_x^2 + p)}{\partial x} + \frac{\partial (\rho V_x V_y)}{\partial y} = 0,0,$$

$$\frac{\partial \rho V_y}{\partial t} + \frac{\partial (\rho V_x V_y)}{\partial x} + \frac{\partial (\rho V_y^2 + p)}{\partial y} = 0,0,$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial (\rho V_x H)}{\partial x} + \frac{\partial (\rho V_y H)}{\partial y} = 0,0,$$

$$E = \frac{1,0}{\gamma - 1,0} \frac{p}{\rho} + \frac{V_x^2 + V_y^2}{2},$$

$$H = \frac{\gamma}{\gamma - 1,0} \frac{p}{\rho} + \frac{V_x^2 + V_y^2}{2},$$

где ρ, p, V_x, V_y – искомые скалярные функции от двух переменных (x, y), отвечающие плотности, давлению и двум компонентам вектора скорости потока газа соответственно, а γ – показатель адиабаты. Решение уравнений производится на прямоугольной области размером $[0, X]\times[0, Y]$ и для временного промежутка $[0, T]$, на которые также накладываются три типа граничных условий:

– для левой стенки ($x = 0, \forall y \in [0, Y]$) ставится задача Дирихле, в соответствии с которой параметры входного потока фиксированы и не изменяются;

– для верхней, нижней и правой стенок ($\forall x \in [0, X], y = \{0, Y\}$ и $x = 1, \forall y \in [0, Y]$) ставится естественное граничное условие, сводящееся к требованию равенства нулю производных по нормали для всех искомых величин;

– для границ обтекаемых объектов ($(x, y) \in G_1 \cup ... \cup G_N$) ставится условие непротекания, в соответствии с которым требуется равенство нулю сразу проекции вектора скорости на нормаль и производных по нормали для давления и плотности.

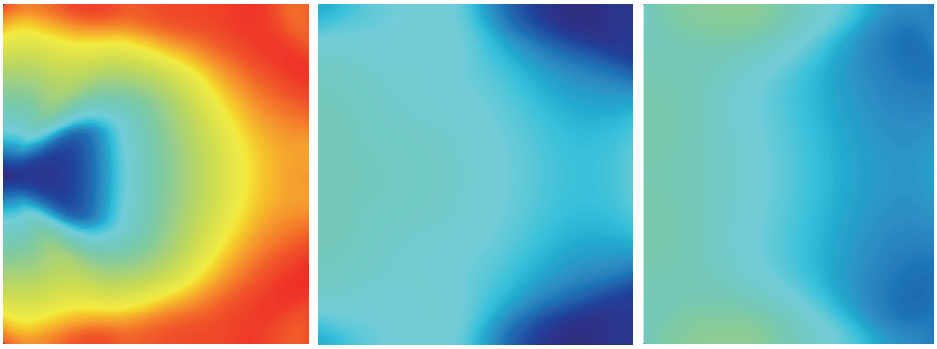


Рис. 1. Модуль скорости, плотность и давление
распространяющегося потока газа

Программная реализация

Решение задачи производилось на структурированной сетке методом HLL, получившим название в честь его авторов Хартена, Лакса и Ван Лира и, в частности, подробно описанным в [5]. Данный метод был реализован в виде двух вычислительных ядер, одно из которых было написано на языке C++ с использованием технологии OpenMP, а другое – с помощью технологии NVidia CUDA, благодаря чему появилась возможность задействовать для расчётов сразу все вычислители кластера.

В целях осуществления подстройки программы под целевую систему в ней были выделены следующие виды модифицируемых параметров:

- ширина теневых граней для MPI. Задавалась как целое число, кратное 32, определяющее, как сильно расчётные области соседних узлов будут перекрываться друг с другом, что позволяет уменьшить количество операций пересылок по сети, но повышает объём вычислений;
- ширина теневых граней для одного узла. Параметр, идейно аналогичный предыдущему, но позволяющий уменьшить число пересылок граничных элементов по шине PCI-E между вычислителями в рамках одного узла. Также определялся как целое число, но кратное уже 8;
- распределение нагрузки по устройствам. Определяет, какие вычислительные устройства стоит использовать и в каком процентном соотношении следует распределять по ним вычислительную нагрузку. Необходимо отметить, что в данном случае рассматриваются и такие ситуации, когда на узле с двумя идентичными графическими ускорителями один из них не используется или же получает меньший объём работы, что в ряде случаев оказывается оправданным;
- размер CUDA-блока. Задаётся двумя целыми числами, произведение которых не превышает 512, определяющими высоту и ширину блока нитей, которые будут обрабатываться одним мультипроцессором GPU.

Легко заметить, что изменение рассматриваемых параметров в уже готовой программе является легко осуществимым действием, однако в зависимости от выбранной их комбинации удаётся сделать программу более требовательной либо к вычислительной мощности, либо к задержкам при пересылке данных на разных уровнях. Таким образом, появляется возможность адаптировать программу под целевой кластер, учитывая его «слабые» стороны и тем самым повышая итоговую скорость расчётов.

К сожалению, подбор оптимальной комбинации параметров необходимо осуществлять, рассматривая не только конфигурацию кластера, но и обрабатываемые данные (в рассматриваемой задаче это размеры области и форма объектов), в результате чего определить искомую комбинацию на момент компиляции программы оказывается крайне затруднительно. В данной работе, в частности, эта проблема решалась с помощью разработанной авторами и уже упомянутой ранее библиотеки TTG Apptimizer. С её помощью при запуске программы на первых ста итерациях проводилось обучение, после чего использовалась лучшая комбинация параметров, которую библиотеке удавалось найти за это время.

Результаты тестирования

Тестирование рассмотренной выше программной реализации осуществлялось на 20 узлах кластера, каждый из которых оснащён двумя шестиядерными процессорами Intel Xeon и тремя ускорителями NVidia Tesla по-

коления Fermi. Непосредственно сравнивались два режима работы программы – с отключённой (далее обозначенный на графиках как static) и включённой (обозначенный как TTG) динамической адаптацией.

Перед тем как переходить к рассмотрению самих результатов, стоит сказать пару слов о комбинации параметров, которая были использована в режиме с отключенной оптимизацией. Она подбиралась «вручную» путём осуществления порядка десяти запусков с целью нахождения некоего локального экстремума производительности при фиксированном размере данных (сетка 4096×4096) и фиксированном количестве узлов (5 штук). В качестве начального варианта была взята комбинация из наиболее очевидных значений, кратных степеням 2. Таким образом подобранная комбинация хоть и не являлась самой удачной, но вполне соответствовала «популярным» значениям, статически закладываемым в большинство MPI-программ.

На рис. 2 представлена зависимость скорости расчётов от количества используемых узлов (на сетке размером 4096×4096):

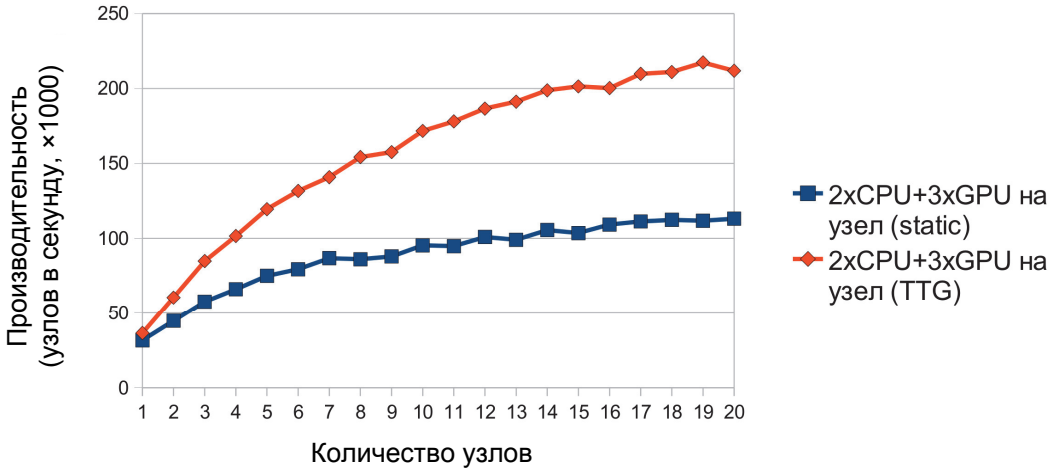


Рис. 2. Ускорение в зависимости от количества узлов

Таким образом, выигрыш от динамической подстройки программы под кластер оказывается весьма существенным и с ростом числа узлов от 1 до 20 он также растёт с 15 до 87 %, что заметно больше ускорений, обычно достигаемых при применении технологий автотюнинга в рамках только одного узла. Также достаточно интересным результатом является оценка достигаемого ускорения в зависимости от размера данных, график которого представлен ниже (при использовании 5 узлов) (рис. 3).

Как ни странно, максимальное ускорение было получено на сетках небольшого размера, а именно 2,64 раза на сетке 512×512 против 1,28 раза на сетке 8192×8192. Объясняется это тем, что на маленьких сетках доступный параллелизм алгоритма уже не позволяет эффективно загрузить все ядра вычислителей, поэтому более эффективной стратегией является отказ

от использования части вычислительной мощности, что и позволяет получить столь заметное ускорение. В частности, на сетке 1024×1024 в результате динамической адаптации на каждом узле использовались только по два GPU, в то время как третий ускоритель и все 12 ядер центрального процессора простаивали.

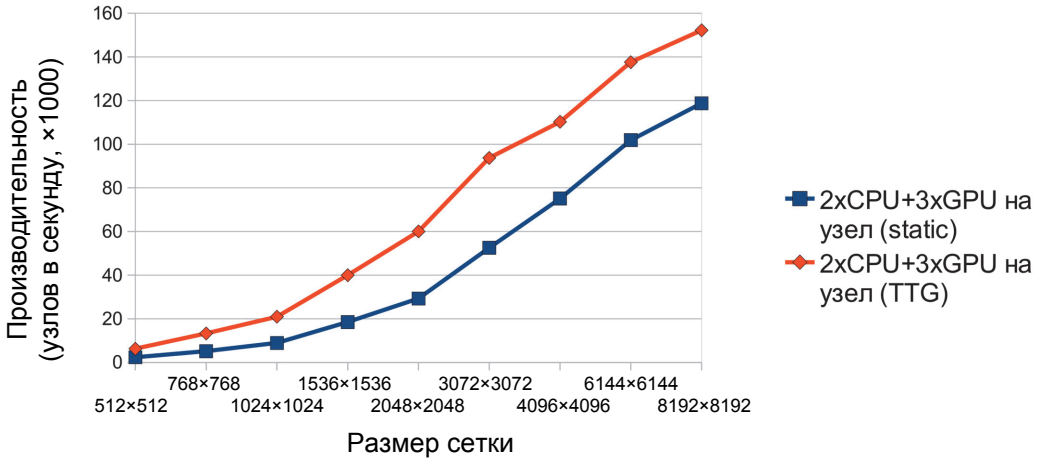


Рис. 3. Ускорение в зависимости от размера данных

Наконец, на последнем графике (рис. 4) проиллюстрирован эффект от применения рассматриваемых видов оптимизации при изменении пропорций расчётной области.

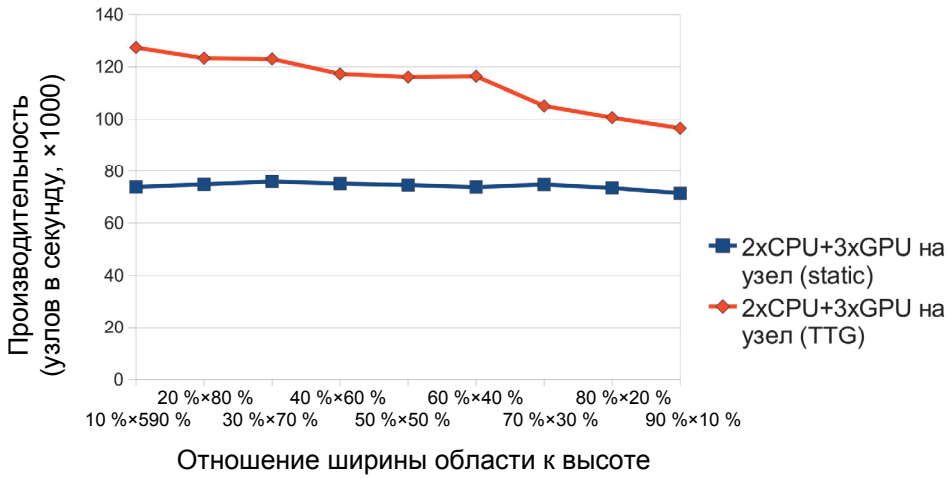


Рис. 4. Ускорение в зависимости от размера данных

В случае статической версии программы изменение соотношения сторон области практически никак не сказались на скорости расчётов. Однако с точки зрения возможностей для оптимизации длинная узкая вертикальная полоса оказывается предпочтительнее горизонтальной, что и было учтено в автоматически оптимизируемой версии программы (достигнутое ускорение – 72 против 35 %).

Заключение

В работе был проиллюстрирован достижимый эффект от использования технологий автоадаптации программ при расширении сферы их применимости с одиночных узлов на GPU-кластерах. На тестовой задаче, заключающейся в численном решении двумерной системы уравнений Эйлера на структурированной сетке, было показано, что подбор оптимальных значений ряда параметров с учётом специфики кластерной системы и размера данных позволяет получить ускорение вплоть до 2,5 раз.

Список литературы

1. Rupp K., Weinbub J., Rudolf F. Automatic Performance Optimization in ViennaCL for GPUs // Proceedings of the 9th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing. – 2010.

2. Кривов М., Притула М., Иванов П.. Апробация парадигмы автоадаптации GPGPU-программ в задачах аэрогидродинамики // CAD/CAM/CAE Observer #3 (87). – 2014.

3. Auto-tuning a High-Level Language Targeted to GPU Codes / S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, J. Cavazos // Innovative Parallel Computing (InPar). – 2012.

4. Application-independent Autotuning for GPU / M. Tillmann, T. Karcher, C. Dachsbacher, W. Tichy // ParCo-Symposium: Application Autotuning for HPC (Architectures). – 2013.

5. Toro E. Riemann Solvers and Numerical Methods for Fluid Dynamics. – Springer, 1999.

К.С. Кузьмина, И.К. Марчевский

Московский государственный технический университет им. Н.Э. Баумана

ОБ ИСПОЛЬЗОВАНИИ ВИХРЕВЫХ МЕТОДОВ ДЛЯ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ В СОПРЯЖЕННЫХ ЗАДАЧАХ АЭРОГИДРОУПРУГОСТИ

Описано применение вихревых методов к решению задач о численном моделировании динамики деформируемого профиля в потоке несжимаемой среды; при этом профиль может двигаться по заданному закону либо перемещаться и деформироваться под действием гидродинамических нагрузок. Задача сводится к численному решению интегрального уравнения относительно интенсивности вихревого слоя, генерируемого на профиле, и системы обыкновенных дифференциальных уравнений, описывающих движение вихревых элементов в области течения. Разработан

программный комплекс POLARA-TVEM, позволяющий проводить расчеты на многопроцессорных вычислительных комплексах с использованием технологии MPI. Приведены оценки эффективности распараллеливания.

Введение

Задачи вычислительной аэрогидродинамики, в особенности сопряженные задачи аэрогидроупругости, традиционно относятся к числу наиболее сложных; необходимость их решения служит мощным стимулом развития как численных методов, так и (в некоторой мере) вычислительной техники. Среди множества разработанных к настоящему времени методов и подходов к численному моделированию аэрогидроупругих явлений следует выделить вихревые методы [1, 2, 3], которые относятся к классу лагранжевых методов и не требуют построения сетки в области течения.

Вихревые методы имеют достаточно узкую область применимости, ограниченную малыми дозвуковыми скоростями течений, поскольку они основаны на модели несжимаемой среды, однако в указанных задачах они могут применяться весьма успешно и заметно превосходить по эффективности другие (сеточные) методы. Наибольшее развитие и обоснование к настоящему времени получили вихревые методы решения двумерных задач, однако даже они на сегодняшний день не реализованы в коммерческих программных комплексах.

В связи с этим разработка эффективных алгоритмов решения сопряженных задач аэрогидроупругости, учитывающих современные достижения в области вихревых методов, а также их программная реализация, предполагающая возможность распаралеливания вычислений и проведения расчетов на многопроцессорных вычислительных комплексах, является актуальным направлением исследований.

Краткое описание метода вихревых элементов

Метод вихревых элементов (МВЭ) – это бессеточный лагранжев метод численного моделирования нестационарного обтекания профилей, основанный на моделировании эволюции завихренности. Распределение завихренности моделируется набором вихревых элементов (ВЭ), для каждого из которых задаются положение в пространстве и интенсивность. Скорость среды в любой точке течения представляет собой суперпозицию влияний отдельных ВЭ, вычисляемых по закону Био – Савара, а давление определяется при помощи аналога интеграла Коши – Лагранжа или иными способами.

Эффективность МВЭ связана с тем, что при решении многих задач область с отличной от нуля завихренностью сравнительно невелика; она расположена вблизи профиля и позади него. Использование завихренности

в качестве первичной расчетной величины позволяет сосредоточить вычислительные ресурсы именно в этой области, а не «распылять» их на всю расчетную область. Особенно эффективным МВЭ является при решении внешних задач обтекания, поскольку на каждом шаге расчета по времени выполнение уравнения неразрывности и граничного условия на бесконечности происходит автоматически.

Граничное условие непротекания или прилипания обеспечивается генерацией ВЭ на поверхности обтекаемых профилей, а течение идеальной либо вязкой среды, описываемое уравнениями Эйлера либо Навье–Стокса, моделируется движением имеющихся ВЭ. Интенсивности генерируемых ВЭ находятся из решения соответствующей системы линейных алгебраических уравнений, аппроксимирующей граничное условие на профиле, а движение ВЭ описывается системой обыкновенных дифференциальных уравнений, правые части которой представляют собой скорости движения ВЭ. Движение и/или деформация обтекаемого профиля, которые могут либо происходить по заданному закону, либо определяться нестационарными гидродинамическими нагрузками в случае решения сопряженной задачи гидроупругости, моделируются присоединенным вихревым слоем и присоединенным слоем источников, интенсивности которых определяются текущими величинами скоростей точек профиля.

Таким образом, алгоритм решения задачи по расчету обтекания профиля содержит несколько основных этапов, представленных на рис. 1, выполняемых на каждом шаге расчета по времени.

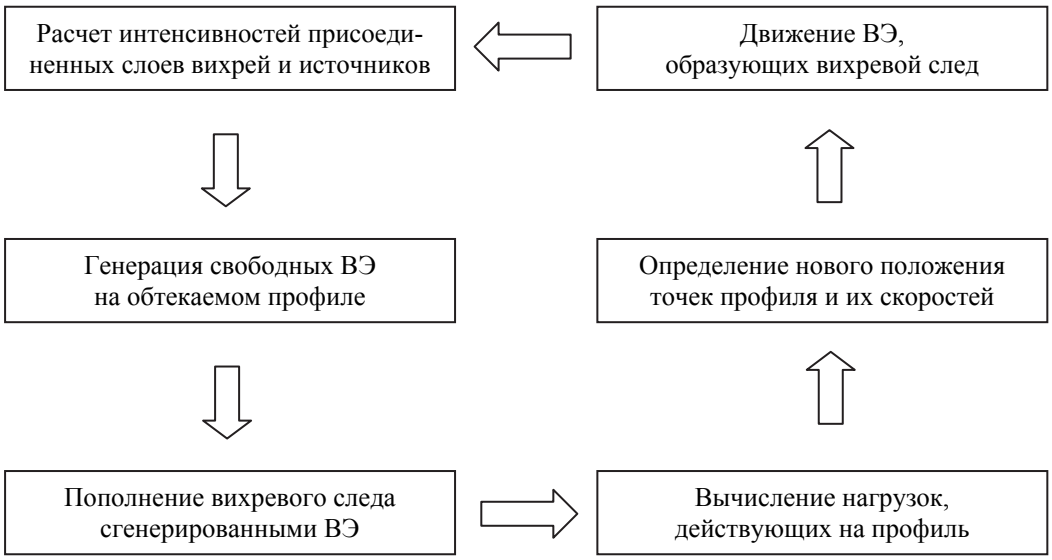


Рис. 1. Основные этапы расчета обтекания профиля методом вихревых элементов

Наиболее затратным с вычислительной точки зрения является этап расчета движения ВЭ. При этом происходит операция определения скоростей всех вихревых элементов, требующая учета взаимного влияния всех пар ВЭ друг на друга, а также несколько менее трудоемкая, но также достаточно затратная операция реструктуризации вихревой пелены, выполняемая на этапе моделирования движения вихревого следа. Основным направлением ускорения расчетов представляется применение параллельных алгоритмов, а также приближенных быстрых методов типа быстрого метода в «задаче N тел». Аналог задачи N тел в методе вихревых элементов возникает при вычислении скоростей движения ВЭ, индуцируемых другими вихревыми элементами.

Математическая модель

Движение вязкой несжимаемой среды описывается уравнениями Навье – Стокса:

$$\nabla \cdot \vec{V} = 0, \qquad \frac{\partial \vec{V}}{\partial \vec{t}} + \left(\vec{V} \cdot \nabla\right)\vec{V} = \nu \Delta \vec{V} - \frac{\nabla p}{\rho},$$

где $\vec{V} = \vec{V}(\vec{r},t)$ – скорость среды; $p = p(\vec{r},t)$ – давление; $\rho = \text{const}$ – плотность, принимаемая постоянной; ν – коэффициент кинематической вязкости.

Граничные условия на бесконечности (условия затухания возмущений) имеют вид:

$$\vec{V}\left(\vec{r},t\right) \to \vec{V}_\infty, \quad p\left(\vec{r},t\right) \to p_\infty, \quad \text{при } \left|\vec{r}\right| \to \infty,$$

где $\vec{V}_\infty$ и p_∞ – постоянные скорость и давление в набегающем потоке; на профиле K выполняется условие прилипания:

$$\vec{V}\left(\vec{r},t\right) = \vec{V}_K\left(\vec{r},t\right), \qquad \vec{r} \in K.$$

Используя вектор завихренности $\vec{\Omega}\left(\vec{r},t\right) = \nabla \times \vec{V}\left(\vec{r},t\right)$, уравнение Навье – Стокса можно записать в форме Гельмгольца [2]:

$$\frac{\partial \vec{\Omega}}{\partial t} + \nabla \times \left(\vec{\Omega} \times \vec{U}\right) = 0, \tag{1}$$

где $\vec{U}\left(\vec{r},t\right) = \vec{V}\left(\vec{r},t\right) + \vec{W}\left(\vec{r},t\right)$, $\vec{W}\left(\vec{r},t\right)$ – диффузионная скорость:

$$\vec{W}\left(\vec{r},t\right) = \nu \frac{\left(\nabla \times \vec{\Omega}\right) \times \vec{\Omega}}{\left|\vec{\Omega}\right|^2}.$$

Уравнение (1) означает, что генерации новой завихренности в области течения S не происходит, а осуществляется только движение существующей вдоль векторных линий поля $\vec{U}$; новая завихренность генерируется только на профиле K.

Влияние профиля заменяется суперпозицией влияний присоединенных (связанных с профилем) вихревого слоя $\gamma_{att}\left(\vec{r},t\right)$ и слоя источников $q_{att}\left(\vec{r},t\right)$ и свободного вихревого слоя $\gamma\left(\vec{r},t\right)$. Все эти слои расположены на профиле, при этом

$$\gamma_{att}\left(\vec{r},t\right)=\vec{V}_K\left(\vec{r},t\right)\cdot\vec{\tau}\left(\vec{r},t\right),\qquad q_{att}\left(\vec{r},t\right)=\vec{V}_K\left(\vec{r},t\right)\cdot\vec{n}\left(\vec{r},t\right),\qquad \vec{r}\in K,$$

где $\vec{\tau}\left(\vec{r},t\right)$ и $\vec{n}\left(\vec{r},t\right)$ – орты касательной и внешней нормали к профилю в соответствующей точке, причем $\vec{n}\left(\vec{r},t\right)\times\vec{\tau}\left(\vec{r},t\right)=\vec{k},$ где $\vec{k}$ – орт нормали к плоскости течения.

Скорость течения может быть вычислена по обобщенному закону Био – Савара:

$$\vec{V}\left(\vec{r},t\right)=\vec{V}_\infty+\frac{1}{2\pi}\int\limits_{S(t)}\frac{\vec{\Omega}\left(\vec{\xi},t\right)\times(\vec{r}-\vec{\xi})}{\left|\vec{r}-\vec{\xi}\right|^2}dS+\frac{1}{2\pi}\oint\limits_{K(t)}\frac{\vec{\gamma}\left(\vec{\xi},t\right)\times(\vec{r}-\vec{\xi})}{\left|\vec{r}-\vec{\xi}\right|^2}dl_\xi+$$
$$+\frac{1}{2\pi}\oint\limits_{K(t)}\frac{\vec{\gamma}_{att}\left(\vec{\xi},t\right)\times(\vec{r}-\vec{\xi})}{\left|\vec{r}-\vec{\xi}\right|^2}dl_\xi+\frac{1}{2\pi}\oint\limits_{K(t)}\frac{q_{att}\left(\vec{\xi},t\right)(\vec{r}-\vec{\xi})}{\left|\vec{r}-\vec{\xi}\right|^2}dl_\xi. \qquad (2)$$

Здесь $S(t)$ и $K(t)$ – область течения и контур профиля в момент времени t; $\vec{\gamma}_{att}=\gamma_{att}\vec{k}$ и $\vec{\gamma}=\gamma\vec{k}$ – векторы интенсивностей присоединенного и свободного вихревых слоев на профиле соответственно.

Если рассматривать поле скоростей $\vec{V}(\vec{r},t)$ на всей плоскости $\mathbb{R}^2$, то на обтекаемом контуре оно будет иметь разрыв. Можно показать [3], что предельное значение вектора скорости среды со стороны профиля на его границе определяется как

$$\vec{V}_-\left(\vec{r},t\right)=\vec{V}\left(\vec{r},t\right)-\frac{\vec{\gamma}\left(\vec{r},t\right)-\vec{\gamma}_{att}\left(\vec{r},t\right)}{2}\times\vec{n}\left(\vec{r},t\right)+\frac{q_{att}\left(\vec{r},t\right)}{2}\vec{n}\left(\vec{r},t\right). \qquad (3)$$

Интенсивность свободного вихревого слоя $\gamma\left(\vec{\xi},t\right)$ может быть найдена из условия прилипания на поверхности тела, которое с учетом представлений (2)–(3) имеет вид

$$\vec{V}_{-}\left(\vec{r},t\right)=\vec{V}_{K}\left(\vec{r},t\right),\qquad \vec{r}\in K.$$

Для решения задачи могут быть использованы различные расчетные схемы: классическая *N*-схема [1, 2] и модифицированная *T*-схема [3, 4]. Их отличие состоит в подходе к удовлетворению граничного условия прилипания на профиле: за счет обеспечения равенства нормальных и касательных компонент скорости на обтекаемом контуре соответственно. Показано [4], что *T*-схема позволяет получать решение для более широкого класса задач по сравнению с *N*-схемой, она также обладает более высокой точностью. К важным особенностям разработанной *T*-схемы следует отнести ее «симметрию» – в случае поступательно движущегося в потоке жесткого профиля она позволяет получать одинаковое решение в прямом и обращенном движении.

Программный комплекс POLARA-TVEM

Для численного решения описанного класса задач разработан параллельный программный комплекс POLARA-TVEM, позволяющий проводить расчеты на многопроцессорных вычислительных комплексах с использованием технологии MPI. Проведенные оценки показали, что наиболее затратными с вычислительной точки зрения являются следующие операции алгоритма МВЭ.

Операция 1. Вычисление скоростей ВЭ как результата их взаимного влияния.

Операция 2. Реструктуризация вихревой пелены: объединение близкорасположенных ВЭ и исключение ВЭ, удалившихся от профиля на значительное расстояние.

Операция 3. Вычисление нагрузок, которые действуют на профиль со стороны потока, по рассчитанному полю скоростей и известному распределению завихренности.

На рис. 2 представлены оценки вычислительной трудоемкости различных этапов последовательного алгоритма МВЭ.

Наиболее трудоемким является вычисление скоростей ВЭ. Другие две операции менее трудоемки, в то время как затратами машинного времени на

Рис. 2. Трудоемкость операций в методе вихревых элементов

прочие операции (например, решение системы линейных уравнений) можно пренебречь. Согласно закону Амдаля для вычислительной системы с S процессорами максимально возможное ускорение программы с долей P параллельного кода определяется как

$$\alpha = \frac{1}{P / S + (1 - P)}.$$

Результаты расчета максимального ускорения, которое теоретически достижимо при распараллеливании только операций 1, операций 1, 2 или операций 1, 2, 3 алгоритма, например, для вычислительных систем с 4, 16 и 64 ядрами, представлены в таблице.

Максимальное ускорение параллельного кода

Распараллеленные операции	Доля параллельного кода	Максимальное ускорение		
		4 ядра	16 ядер	64 ядра
1	95,3 %	3,5	9,4	16,2
1, 2	98,2 %	3,8	12,6	30,0
1, 2, 3	99,5 %	3,9	14,9	48,7

Для вычислений с использованием небольшого числа ядер максимальные значения ускорений близки, но при использовании системы с десятками вычислительных ядер распараллеливание операций 2 и 3 метода вихревых элементов дает значительный вклад в ускорение, несмотря на то, что трудоемкость этих операций составляют вместе лишь малую долю от трудоемкости операции 1 (приблизительно 4,4 %).

Реальное ускорение, наблюдаемое в расчетах, несколько меньше, поскольку оценки по закону Амдаля получены для «идеального» распараллеливания, т. е. ситуации, когда параллельный код выполняется в S раз быстрее на S-ядерной вычислительной системе.

На рис. 3 для программного комплекса POLARA-TVEM представлена зависимость ускорения от числа задействованных вычислительных ядер при распараллеливании операций 1, 2 и 3 алгоритма, полученная при решении одной, весьма сложной с вычислительной точки зрения задачи расчета обтекания профиля. Для сравнения показаны максимальные значения ускорения из оценок по закону Амдаля.

Реальное ускорение при распараллеливании операций 1, 2, 3 близко к оценке по закону Амдаля для распараллеливания только операций 1, 2 (сказывается наличие пересылок данных, затрат на синхронизацию и прочих «накладных расходов»), поэтому можно сделать вывод о том, что распараллеливание операции 3 важно для достижения высокой эффективности, хотя ее трудоемкость крайне мала – всего около 1,3 %.

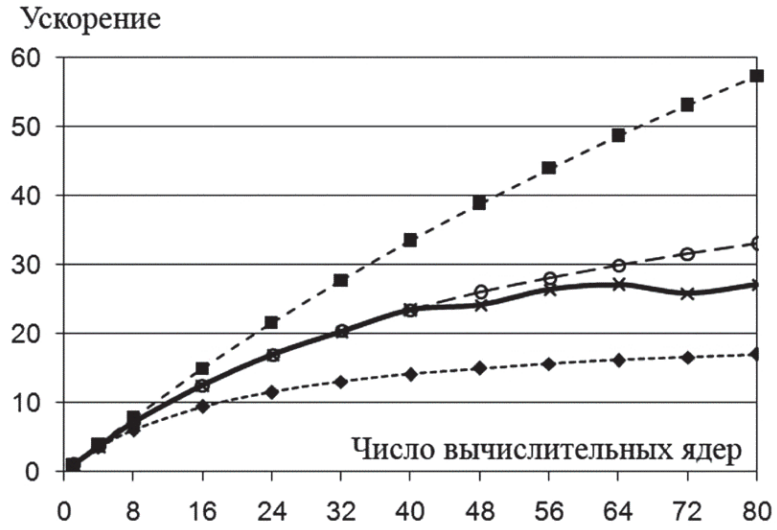


Рис. 3. Реальное ускорение и оценки по закону Амдаля:
– распараллеливание операции 1; – распараллеливание операции 1, 2;
– распараллеливание операции 1, 2, 3; – реальное ускорение

Работа выполнена в рамках государственного задания № 2014/104 в сфере научной деятельности в части государственного задания Минобрнауки России (проект 1712) при финансовой поддержке гранта Президента РФ (проект МК-3705.2014.8).

Список литературы

1. Cottet G.-H., Koumoutsakos P.D. Vortex Methods: Theory and Practice. – Cambridge University Press, 2008. – 328 p.

2. Андронов П.Р., Гувернюк С.В., Дынникова Г.Я. Вихревые методы расчета нестационарных гидродинамических нагрузок. – М.: Изд-во МГУ, 2006. – 184 с.

3. Accuracy considerations for implementing velocity boundary conditions in vorticity formulations / S.N. Kempka, M.W. Glass, J.S. Peery, J.H. Strickland // SANDIA REPORT. SAND96-0583, UC-700, 1996. – 52 p.

4. Kuzmina K.S., Marchevsky I.K. On Numerical Schemes in 2D Vortex Element Method for Flow Simulation Around Moving and Deformable Airfoils // Advanced Problems in Mechanics: Proc. of the XLII School-Conference. – St.Petersburg, 2014. – P. 335–344.

В.Д. Кустикова, П.Н. Дружков

Нижегородский государственный университет им. Н.И. Лобачевского

ПРИМЕНЕНИЕ МЕТОДОВ ГЛУБИННОГО ОБУЧЕНИЯ ДЛЯ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Рассматривается задача классификации изображений. Для решения задачи применяются методы глубинного обучения, основанные на построении многослойных нейронных сетей. Описывается реализация на базе библиотеки машинного обучения Pylearn2, использующей ресурсы графического процессора при выполнении ряда операций. Анализируются результаты вычислительных экспериментов.

Введение

Искусственные нейронные сети – широко известные модели, которые в течение длительного времени используются при решении задач распознавания образов. Повышенный интерес к многослойным нейросетям в последние несколько лет обусловлен созданием и разработкой методов *глубинного обучения* (deep learning) [1 – 3]. В настоящее время активно развивается направление, связанное с разработкой методов предварительной настройки параметров сетей (pre-training), а также построением эффективных топологий нейросетей для решения определенного круга задач. Задача классификации изображений не является исключением.

Задача классификации изображений

Задача классификации предполагает, что для изображения необходимо построить описание и определить категорию (класс), которой оно принадлежит. Задача существенно усложняется с ростом количества категорий, при наличии на изображении нескольких объектов разных классов и введении семантической иерархии категорий, так как изображение может одновременно принадлежать нескольким классам. Наличие нечетких границ между категориями ставит проблему определения вероятности принадлежности изображения каждому из классов.

Методы решения

Методы глубинного обучения в настоящее время находятся среди наиболее перспективных при решении задач классификации. Это подтверждается значительным прорывом в результатах известных конкурсов ImageNet [4] и PASCAL Visual Object Challenge [5]. В основе этих методов лежит идея построения многоуровневых иерархий представления данных, например, с помощью нейронных сетей, содержащих более двух слоев. Ие-

рархии позволяют автоматически синтезировать описания данных – *признаки* – более высокого уровня на основании признаков более низкого уровня. Так, применительно к изображениям сеть может содержать несколько слоев, в простейшем случае это «пикселы→ребра→комбинации ребер».

В данной работе демонстрируется применение *разреженного кодирования* (sparse coding) [6] – одного из передовых подходов группы методов визуальных слов (bag-of-words) [7] – и *сверточных нейронных сетей* (convolutional neural networks) [8].

Библиотека глубинного обучения Pylearn2

Pylearn2 [9] – библиотека машинного обучения, содержащая реализацию широкого спектра методов глубинного обучения. Обеспечивает возможность формирования нейросетей посредством их описания в формате yaml-файлов, предоставляет инструменты обучения многослойных моделей и последующего их тестирования. Среди моделей присутствуют многослойные нейросети и сверточные нейронные сети, также имеется реализация ряда методов предварительной настройки параметров сети. Сверточные сети считаются одними из наиболее перспективных и ресурсоемких. В связи с этим Pylearn2 для ускорения вычислений интегрирует библиотеку Cuda-convnet [10] на базе технологии CUDA. Реализация большого количества моделей, относительная простота и удобство использования по сравнению с аналогами, а также приемлемая скорость выполнения вычисления стали решающими при выборе библиотеки.

Заключение

В процессе исследования проведены эксперименты с различными видами многослойных моделей. Выполнено сравнение с результатами, полученными при решении задачи классификации с помощью ранее популярных методов, основанных на извлечении признаков.

Актуальное направление текущих исследований – поиск и разработка методов с целью повышения качества классификации, изучение возможности применения методов для решения задачи детектирования объектов определенных классов.

Работа выполнена при поддержке гранта РФФИ по теме «Исследование применимости методов глубокого обучения к решению задач компьютерного зрения на примере классификации изображений с большим числом категорий, детектирования пешеходов и автомобилей» (договор № НК 14-07-31269\14) в лаборатории «Информационные технологии» факультета ВМК ННГУ им. Н.И. Лобачевского.

Список литературы

1. Hinton G.E. Learning Multiple Layers of Representation // Trends in Cognitive Sciences. – 2007. – Vol. 11. – P. 428–434.

2. Schmidhuber J. Deep Learning in Neural Networks: An Overview. – URL: http://arxiv.org/abs/1404.7828.

3. Актуальная информация по методам глубинного обучения. – URL: http://deeplearning.net.

4. ImageNet. – URL: http://www.image-net.org.

5. PASCAL Visual Object Challenge. – URL: http://pascallin.ecs.soton.ac.uk/challenges/VOC.

6. Lee H., Battle A., Raina R., Ng A.Y. Efficient sparse coding algorithms // Advances in Neural Information Processing Systems 19 (NIPS 2006). – 2006. – P. 801–808.

7. Visual categorization with bags of keypoints / C. Dance, J. Willamowski, L. Fan, C. Bray, G. Csurka // Proc. ECCV International Workshop on Statistical Learning in Computer Vision. – 2004.

8. LeCun Y., Kavukcuoglu K., Farabet C. Convolutional networks and applications in vision // Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS). – 2010. – P. 253–256.

9. Pylearn2 – a machine learning library. – URL: http://deeplearning.net/software/pylearn2.

10. Cuda-convnet – high-performance C++/CUDA implementation of convolutional neural networks. – URL: http://code.google.com/p/cuda-convnet.

И.Г. Лебедев, К.А. Баркалов

Нижегородский государственный университет им. Н.И. Лобачевского

GPU-РЕАЛИЗАЦИЯ АЛГОРИТМА ГЛОБАЛЬНОГО ПОИСКА

Рассмотрен параллельный алгоритм решения многомерных многоэкстремальных задач оптимизации. Описыается реализация алгоритма на современных вычислительных системах с использованием графических процессоров. Экспериментально подтверждено ускорение алгоритма с использованием GPU по сравнению с тем же алгоритмом, реализованным только на CPU. Вычислительные эксперименты проведены на серии из нескольких сотен многомерных многоэкстремальных задач.

Введение

Рассмотрим параллельные алгоритмы решения задач многоэкстремальной оптимизации. Целевая функция φ удовлетворяет условию Лип-

шица с неизвестной априори константой L. Подобные задачи широко распространены в приложениях (см. обзор [1]).

Одним из перспективных направлений в области параллельной глобальной оптимизации является использование графических ускорителей (GPU), востребованных в решении целого ряда вычислительно трудоемких задач [2]. Однако в области глобальной оптимизации потенциал графических ускорителей до конца еще не раскрыт. С использованием GPU в основном распараллеливают алгоритмы, которые так или иначе основаны на идее случайного поиска (см. работы [3–6]); обзор данного направления GPU-алгоритмов оптимизации представлен в [7]. В силу своей вероятностной природы алгоритмы подобного типа, вообще говоря, не гарантируют сходимости к глобально-оптимальному решению, что невыгодно отличает их от детерминированных методов.

Для многих детерминированных алгоритмов липшицевой глобальной оптимизации с гарантированной сходимостью предложены их параллельные варианты [8–11]. Однако указанные версии алгоритмов распараллелены на CPU с использованием технологий MPI и/или OpenMP, реализации на GPU отсутствуют. В настоящей статье приведены результаты исследования GPU-реализации параллельного алгоритма глобального поиска, разработанного в рамках информационно-статистического подхода, представленного в монографии [12].

Многомерный алгоритм глобального поиска

Рассмотрим задачу поиска глобального минимума N-мерной функции $\varphi(y)$ в гиперинтервале $D = \{y \in R^N : a_i \le y_i \le b_i,\ 1 \le i \le N\}$. При этом будем предполагать, что функция удовлетворяет условию Липшица с априори неизвестной константой L.

$$\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \tag{1}$$

$$\left|\varphi(y_1) - \varphi(y_2)\right| \le L\left\|y_1 - y_2\right\|,\ \ y_1, y_2 \in D,\ \ 0 < L < \infty, \tag{2}$$

В данной работе мы будем использовать подход, основанный на идее редукции размерности с помощью кривой Пеано $y(x)$, непрерывно и однозначно отображающей отрезок вещественной оси [0,1] на n-мерный куб:

$$\left\{y \in R^N : -2^{-1} \le y_i \le 2^{-1},\ 1 \le i \le N\right\} = \left\{y(x) :\ 0 \le x \le 1\right\}. \tag{3}$$

Вопросы численного построения отображений типа кривой Пеано и соответствующая теория подробно рассмотрены в [12, 15].

Использование подобного рода отображений позволяет свести многомерную задачу (1) к одномерной задаче:

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

Важным свойством является сохранение ограниченности относительных разностей функции: если функция $\varphi(y)$ в области D удовлетворяла условию Липшица, то функция $\varphi(y(x))$ на интервале $[0,1]$ будет удовлетворять равномерному условию Гельдера $\left|\varphi(y(x_1)) - \varphi(y(x_2))\right| \le H\left|x_1 - x_2\right|^{1/N}$, $x_1, x_2 \in [0,1]$, где константа Гельдера H связана с константой Липшица L соотношением $H = 4Ld\sqrt{N}$, $d = \max\{b_i - a_i : 1 \le i \le N\}$.

Поэтому, не ограничивая общности, можно рассматривать минимизацию одномерной функции $f(x) = \varphi(y(x))$, $x \in [0,1]$, удовлетворяющей условию Гельдера.

Рассматриваемый алгоритм решения данной задачи предполагает построение последовательности точек x^k, в которых вычисляются значения минимизируемой функции $z^k = f(x^k)$. Процесс вычисления значения функции (включающий в себя построение образа $y^k = y(x^k)$) будем называть испытанием, а пару (x^k, z^k) – результатом испытания. Множество пар $\{(x^k, z^k)\}$, $1 \le k \le n$ составляют поисковую информацию, накопленную методом после проведения n шагов. В нашем распоряжении имеется $p \ge 1$ вычислительных элементов и в рамках одной итерации метода будем проводить p испытаний одновременно (синхронно). Обозначим $k(n)$ общее число испытаний, выполненных после n параллельных итераций.

На первой итерации метода испытание проводится в произвольной внутренней точке x^1 интервала $[0,1]$. Пусть выполнено $n \ge 1$ итераций метода, в процессе которых были проведены испытания в $k = k(n)$ точках x^i, $1 \le i \le k$. Тогда точки $x^{k+1}, \ldots, x^{k+p}$ поисковых испытаний следующей $(n+1)$-й итерации определяются в соответствии с правилами.

Шаг 1. Перенумеровать точки множества $X_k = \{x^1, \ldots, x^k\} \cup \{0\} \cup \{1\}$, которое включает в себя граничные точки интервала $[0,1]$, а также точки предшествующих испытаний, нижними индексами в порядке увеличения значений координаты, т.е.

$$0 = x_0 < x_1 < \ldots < x_{k+1} = 1.$$

Шаг 2. Полагая $z_i = f(x_i), 1 \le i \le k$, вычислить величины

$$\mu = \max_{1 \le i \le k} \frac{|z_i - z_{i-1}|}{\Delta_i}, \quad M = \begin{cases} r\mu, \mu > 0, \\ 1, \mu = 0, \end{cases}$$

где $r > 1$ является заданным параметром метода, а $\Delta_i = (x_i - x_{i-1})^{1/N}$.

Шаг 3. Для каждого интервала $(x_{i-1}, x_i), 1 \le i \le k+1$ вычислить характеристику в соответствии с формулами:

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M},\ R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M}.,$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2\Delta_i} - 2\frac{z_i + z_{i-1}}{M},\ 1 < i < k+1.$$

Шаг 4. Характеристики $R(i), 1 \le i \le k+1$ упорядочить в порядке убывания:

$$R(t_1) \ge R(t_2) \ge \ldots \ge R(t_{k-1}) \ge R(t_{k+1})$$

и выбрать p наибольших характеристик с номерами интервалов $t_j, 1 \le j \le p$.

Шаг 5. Провести новые испытания в точках $x^{k+j}, 1 \le j \le p$, вычисленных по формулам:

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2},\ t_j = 1, t_j = k+1,$$

$$x^{k+1} = \frac{x_{t_j} + x_{t_j-1}}{2} - \mathrm{sign}(z_{t_j} - z_{t_j-1})\frac{1}{2r}\left[\frac{|z_{t_j} - z_{t_j-1}|}{\mu}\right]^N,\ 1 < t_j < k+1.$$

Алгоритм прекращает работу, если выполняется условие $\Delta_{t_j} \le \varepsilon$ хотя бы для одного номера $t_j, 1 \le j \le p$; здесь $\varepsilon > 0$ есть заданная точность. В качестве оценки глобально-оптимального решения задачи (1) выбираются значения:

$$f_k^* = \min_{1 \le i \le k} f(x^i),\ x_k^* = \arg \min_{1 \le i \le k} f(x^i).$$

Обоснование данного способа организации параллельных вычислений см. в [12, 18].

Реализация на GPU

Общая схема организации вычислений с использованием GPU приведена на рисунке. В соответствии с данной схемой на CPU выполняются шаг 1–4 параллельного алгоритма глобального поиска. Координаты *p* точек испытаний, вычисленные на шаге 4 алгоритма, накапливаются в промежуточном буфере, а затем передаются на графический процессор. На GPU происходит вычисление значений функции в этих точках, после чего результаты испытаний (снова через промежуточный буфер) передаются на CPU.

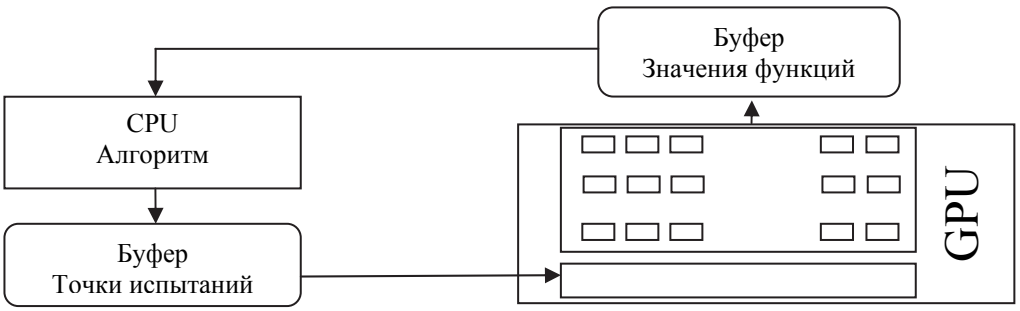


Рис. Схема информационных обменов в GPU-алгоритме

Результаты численных экспериментов

Вычислительные эксперименты проводились на одном из узлов высокопроизводительного кластера ННГУ им. Н.И. Лобачевского. Узел кластера располагает двумя процессорами Intel Sandy Bridge E5-2660 2.2 GHz, 64 Gb RAM и графическим процессором NVIDIA Tesla M2090. Центральный процессор является 8-ядерным (т.е. всего на узле доступно 16 физических ядер и 16 виртуальных в режиме HyperThreading), а на графическом процессоре доступны 16 мультипроцессоров (512 ядер). Для реализации GPU-алгоритма использовалась технология CUDA.

С целью имитации вычислительной трудоемкости, присущей прикладным задачам оптимизации, расчет целевой функции во всех проводимых экспериментах был усложнен дополнительными вычислениями, не меняющими вид функции и расположение ее минимумов.

В работе [13] описан GKLS-генератор, позволяющий порождать задачи многоэкстремальной оптимизации с заранее известными свойствами: количеством локальных минимумов, размерами их областей притяжения, точкой глобального минимума, значением функции в ней и т.п.

Ниже приведены результаты численного сравнения трех последовательных алгоритмов – DIRECT [13], DIRECT*l* [16] и алгоритма глобального поиска (АГП) (результаты работы первых двух алгоритмов приводятся по работе [13]). Численное сравнение проводилось на классах функций Simple и Hard размерности 4 и 5 из [13]. Глобальный минимум y^* считался найденным, если алгоритм генерировал точку испытания y^k в δ-окрестности глобального минимума, $\delta = \|b-a\|\sqrt[N]{\Delta}$, здесь N – размерность решаемой задачи, a и b – границы области поиска D, параметр $\Delta = 10^{-6}$ при $N = 4$ и $\Delta = 10^{-7}$ при $N = 5$. При использовании метода АГП для класса Simple выбирался параметр $r = 4{,}5$, для класса Hard – $r = 5{,}6$; параметр построения кривой Пеано был фиксированным, $m = 10$. Максимально допустимое число итераций составляло $K_{\max} = 1\,000\,000$.

В табл. 1 отражено среднее число итераций k_{av}, которые выполнил метод при решении серии задач из данных классов. Символ «>» отражает ситуацию, когда не все задачи класса были решены каким-либо методом. Количество нерешенных задач указано в скобках.

Таблица 1

Среднее число итераций

N	Problem class	DIRECT	DIRECT*l*	АГП
4	Simple	>47282 (4)	18983	11953
	Hard	>95708 (7)	68754	25263
5	Simple	>16057 (1)	16758	15920
	Hard	>217215 (16)	>269064 (4)	>148342 (4)

Как видно из табл. 1, АГП превосходит методы DIRECT и DIRECT*l* на всех классах задач по среднему числу итераций.

Оценим теперь ускорение, которое достигается при использовании параллельного алгоритма глобального поиска, в зависимости от числа p используемых ядер. В табл. 2 приведено среднее число итераций, которые выполнил метод при решении серии задач на CPU, а в табл. 3 – среднее время решения одной задачи, в табл. 4 и 5 приведены аналогичные данные для GPU-реализации. Ускорение и избыточность GPU-алгоритма будем измерять относительно CPU-алгоритма, запущенного на 32 ядра (см. столбец $p = 32$ в табл. 2 и 3)

Таблица 2

Среднее число итераций на CPU

N	Problem class	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
4	Simple	11953	6014	2547	1413	756	323
	Hard	25263	16509	8970	4666	1910	1252
5	Simple	15920	17797	8246	3773	1713	781
	Hard	>148342(4)	98667	45410	26732	11322	11745

Таблица 3

Среднее время решения задачи на CPU

N	Problem class	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
4	Simple	3,02	1,80	1,00	0,71	0,57	0,38
	Hard	6,05	4,48	2,94	1,88	1,36	1,25
5	Simple	4,84	5,84	3,18	1,82	1,44	0,95
	Hard	>42,5	33,77	15,62	14,16	8,37	11,67

Таблица 4

Среднее число итераций на GPU

N	Problem class	$p = 100$	$p = 200$	$p = 300$	$p = 400$	$p = 500$	$p = 800$	$p = 1000$
4	Simple	105	56	53	28	24	18	12
	Hard	352	181	107	87	84	37	33
5	Simple	193	105	62	44	41	55	36
	Hard	2452	809	508	376	353	246	172

Таблица 5

Среднее время решения задачи на GPU

N	Problem class	$p = 100$	$p = 200$	$p = 300$	$p = 400$	$p = 500$	$p = 800$	$p = 1000$
4	Simple	0,16	0,16	0,22	0,16	0,16	0,19	0,15
	Hard	0,50	0,47	0,43	0,44	0,52	0,37	0,41
5	Simple	0,33	0,32	0,29	0,27	0,31	0,60	0,48
	Hard	3,46	2,11	1,98	1,97	2,25	2,54	2,15

Таблица 6

Ускорение по времени $S(p)$ на GPU

N	Problem class	$p = 100$	$p = 200$	$p = 300$	$p = 400$	$p = 500$	$p = 800$	$p = 1000$
4	Simple	2,30	2,35	1,75	2,37	2,38	1,94	2,45
	Hard	2,49	2,68	2,93	2,85	2,41	3,38	3,07
5	Simple	2,90	2,99	3,32	3,58	3,05	1,59	1,98
	Hard	4,09	6,72	7,17	7,19	6,30	5,56	6,59

Таблица 7

Ускорение по итерациям $s(p)$ на GPU

N	Problem class	$p = 100$	$p = 200$	$p = 300$	$p = 400$	$p = 500$	$p = 800$	$p = 1000$
4	Simple	3,07	5,78	6,11	11,43	13,54	17,96	27,42
	Hard	3,55	6,90	11,70	14,38	14,98	33,46	37,40
5	Simple	4,05	7,45	12,57	17,80	18,83	14,19	21,91
	Hard	4,79	14,51	23,12	31,21	33,26	47,68	68,21

Результаты экспериментов показывают преимущество GPU-алгоритма по сравнению с CPU-алгоритмом. Так, алгоритм с использованием максимального количества ядер CPU работает медленнее, чем любая из представленных GPU-реализаций. Также результаты показывают значительное ускорение по числу итераций.

Список литературы

1. Pinter J.D. (Ed.) Global Optimization: Scientific and Engineering Case Studies. – Springer Verlag, 2006. – 546 p.

2. Wen-mei Hwu. GPU Computing Gems Emerald Edition (Applications of GPU Computing Series). – Morgan Kaufmann, 2011. – 886 p.

3. An efficient implementation of parallel simulated annealing algorithm in GPUs / A.M. Ferreiro, J. A. García, J. G. López-Salas, C. Vázquez // Journal of Global Optimization. – 2013. – Vol. 57, Is. 3. – P. 863–890.

4. Zhu W. Massively parallel differential evolution – pattern search optimization with graphics hardware acceleration: an investigation on bound constrained optimization problems // Journal of Global Optimization. – 2011. – Vol. 50, Is. 3. – P. 417–437.

5. García-Martínez J.M., Garzón E.M.,·Ortigosa P.M. A GPU implementation of a hybrid evolutionary algorithm: GPuEGO // The Journal of Supercomputing. – 2014.

6. GPU implementation of a road sign detector based on particle swarm optimization / L. Mussi [et al.] // Evolutionary Intelligence. – 2010. – Vol. 3, Is. 3-4. – P. 155–169.

7. Langdon W.B. Graphics processing units and genetic programming: an overview // Soft Computing. – 2011. – Vol. 15, Is. 8. – P. 1657–1669.

8. Gergel V.P., Strongin R.G. Parallel computing for globally optimal decision making on cluster systems // Future Generation Computer Systems. – 2005. – Vol. 21, Is. 5. – P. 673–678.

9. Evtushenko Yu.G., Malkova V.U., Stanevichyus A.A., Parallel global optimization of functions of several variables // Computational Mathematics and Mathematical Physics. – 2009. – Vol. 49, Is. 2. – P. 246–260.

10. Design and implementation of a massively parallel version of DIRECT / J. He, A. Verstak, L.T. Watson, M. Sosonkina. // Computational Optimization and Applications. – 2008. – Vol. 40, Is. 2. – P. 217–245.

11. Paulavicius R., Zilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds // Optimization Methods & Software. – 2011. – Vol. 26, Is. 3. – P 487–498.

12. Strongin R.G., Sergeyev Ya.D., Global optimization with non-convex constraints. Sequential and parallel algorithms. – Kluwer Academic Publishers, Dordrecht. – 2000. – 704 p.

13. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008. – 352 с.

14. Параллельные вычисления в задачах глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов. – М.: Изд-во МГУ, 2012. – 280 с.

15. Sergeyev Ya.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space-filling curves. – Springer, 2013. – 125 p.

16. Gablonsky J.M., Kelley C.T. A Locally-Biased Form of the DIRECT Algorithm // Journal of Global Optimization. – 2001. – Vol. 21, Is. 1. – P. 27–37.

С.А. Лебедев, Е.А. Козинов

Нижегородский государственный университет им. Н.И. Лобачевского

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ВЫПОЛНЕНИЯ ЧИСЛЕННОЙ ФАЗЫ РАЗЛОЖЕНИЯ ХОЛЕЦКОГО

Рассматривается проблема распараллеливания вычислений при решении разреженных СЛАУ с симметричной положительно определенной матрицей методом Холецкого. Предлагается метод распараллеливания численной фазы разложения, основанный на динамической схеме балансировки нагрузки. Приводятся результаты вычислительных экспериментов, показывающие сопоставимость выполненной реализации с рядом распространенных библиотек.

Введение

Системы линейных алгебраических уравнений с разреженной симметричной положительно определенной матрицей возникают при моделировании процессов в разных предметных областях. В задачах, представляющих практический интерес, размер таких систем может составлять 10^7 и более, что обусловливает необходимость разработки высокопроизводительного программного обеспечения. Прямые методы, основанные на факторизации матрицы системы с последующим решением более простых систем, широко применяются на практике. Разработано немало программных библиотек, ориентированных на использование прямых методов (Intel MKL PARDISO, MUMPS, SuperLU, CHOLMOD и другие [8]). С 2011 г. на факультете ВМК ННГУ разрабатывается прямой решатель разреженных СЛАУ с симметричной положительно определенной матрицей, основанный на методе Холецкого [11]. В рамках данного решателя изучаются вопросы оптимизации соответствующих алгоритмов под современные многоядерные архитектуры. В данной работе рассматривается задача распараллеливания численной фазы разложения Холецкого, наиболее затратной по времени в большинстве случаев. Реализация численной фазы основана на мультифронтальном методе и описана в [12]. Предлагается метод распараллеливания, основанный на динамической схеме балансировки нагрузки.

Постановка задачи и метод решения

Дана система линейных уравнений $Ax = b$, где A – разреженная симметричная положительно определенная матрица, b – плотный вектор, x – вектор неизвестных. Необходимо найти решение системы x.

Принцип работы прямых методов основан на факторизации матрицы системы с последующим решением треугольных систем. Для симметричных положительно определенных матриц факторизация выполняется методом Холецкого. Для этого в большинстве случаев используется двухфазный подход: вначале находится портрет фактора, т.е. расположение ненулевых элементов (символьное разложение), затем полученный портрет заполняется значениями (численное разложение). Необходимо отметить, что символьная фаза выполняется гораздо быстрее численной, поэтому множество усилий исследователей направлено на оптимизацию и распараллеливание численной фазы.

Мультифронтальный метод [7] является одним из самых эффективных и масштабируемых методов решения задач численного разложения. Мультифронтальный метод может эффективно использовать кэш-память всех уровней, при использовании подходящих структур данных основан на операциях BLAS 3. Данный метод используется в одном из широко распространенных решателей с открытым исходным кодом MUMPS [10]. Подробное описание мультифронтального метода можно найти в работе [9].

Параллельные методы разложения Холецкого

Распараллеливание мультифронтального метода может быть выполнено на основе дерева исключения [9], содержащего информацию о зависимостях по данным, возникающим в ходе вычислений. Ключевым здесь является тот факт, что структура строки матрицы L представляет собой подмножество поддерева дерева исключения с корнем в вершине, соответствующее этой строке. Поэтому любой узел из множества ненулевых элементов *j*-го столбца является потомком *j*-го узла в дереве исключения. Таким образом, *j*-й столбец фактора не может быть вычислен до тех пор, пока не вычислены столбцы с номерами из указанного множества. Продолжая эти рассуждения, можно показать, что для вычисления очередного столбца фактора необходимо вычислить все столбцы с номерами из поддерева соответствующего узла. Тем не менее остальные узлы не используются в вычислениях и поэтому могут рассматриваться независимо. Следовательно, два столбца *i* и *j* могут быть вычислены параллельно тогда и только тогда, когда поддеревья *T[i]* и *T[j]* не пересекаются, то есть не имеют общих узлов. Для распараллеливания по дереву исключения могут использоваться как статические, так и динамические схемы.

Статические схемы распараллеливания

Существует множество методов, использующих статическую схему распараллеливания, большинство из них основано на алгоритме Гейста-Нг [4]. Идея алгоритма заключается в нахождении некоторого слоя в дереве

исключения, то есть множества узлов, которые не обязательно находятся на одном уровне, но при этом не имеют общих потомков. Найденный слой должен обладать свойством сбалансированности, так, чтобы количество операций, необходимых для обработки узлов поддеревьев с корнями в узлах найденного слоя, удовлетворяло заданному порогу и было примерно одинаковым. Сравнение различных статических методов распараллеливания можно найти в статьях [1, 3]. Одним из основных недостатков статических схем является невозможность достаточно точно оценить объем работы, необходимый для обработки каждого узла дерева исключения. Поэтому в данной работе предлагается другой способ балансировки нагрузки, основанный на динамической схеме.

Динамическая схема распараллеливания

В рамках динамической схемы строится пул задач, который организуется в виде приоритетной очереди. На каждом шаге алгоритма поток достает задачу из очереди и приступает к ее выполнению (рис. 1).

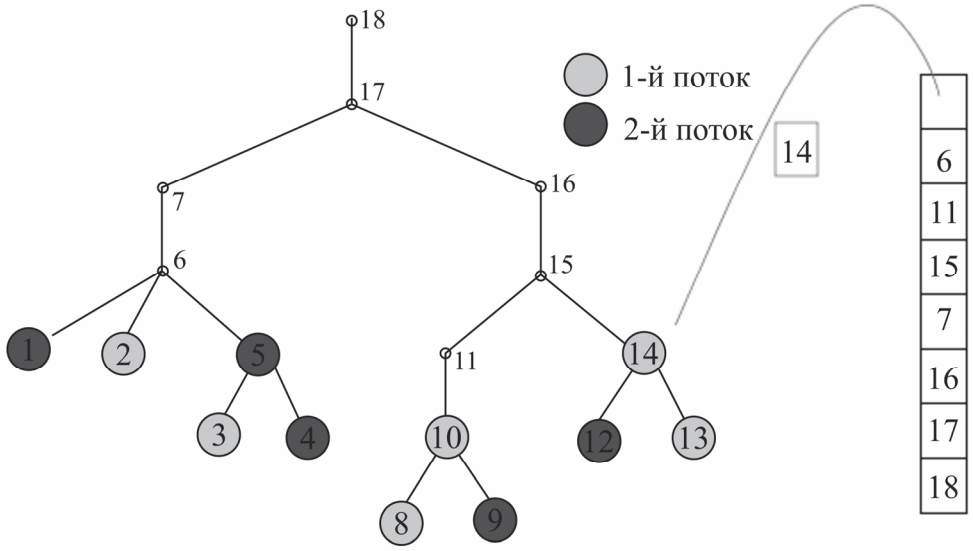


Рис. 1. Динамическая схема распараллеливания

Каждая задача соответствует вычислению соответствующего столбца фактора и состоит из четырех подзадач: вычисление матрицы узла, вычисление фронтальной матрицы, формирование из фронтальной матрицы столбца фактора, вычисление матрицы обновления.

Для формирования очереди задач используется алгоритм, который учитывает различные характеристики узлов дерева для достижения лучшей балансировки. Алгоритм обходит все узлы дерева в соответствии с топологической перестановкой, сформированной ранее, и на каждой итерации цикла добавляет рассматриваемый узел в очередь. Приоритет узла в

очереди складывается из основного и второстепенного, где первый отвечает за правильный обход столбцов в параллельном мультифронтальном методе, а второй – за улучшение балансировки. Основной приоритет равен количеству детей узла в дереве исключения, а второстепенный вычисляется как оценка трудоемкости решения соответствующих подзадач. Трудоемкость решения задачи в данной работе оценивается как количество операций с плавающей запятой.

Результаты вычислительных экспериментов

Результаты экспериментов получены с использованием узла кластера[1], содержащего два восьмиядерных процессора Intel Sandy Bridge E5-2660 2.2 GHz, 64GB RAM, работающего под управлением ОС Linux CentOS 6.4. Использовались компилятор Intel C++ Compiler и библиотека Intel MKL BLAS из пакета Intel® Parallel Studio XE 2013 SP1. Для проведения экспериментов были выбраны матрицы из коллекции университета Флориды [2]. Характеристики тестовых матриц представлены ниже (таблица). Все они являются симметричными, положительно определенными.

Характеристики тестовых матриц

Название матрицы	Порядок	Число ненулевых элементов	Число ненулевых элементов в факторе
Pwtk	217 918	5 926 171	49 025 872
Msdoor	415 863	10 328 399	51 882 257
parabolic_fem	525 825	2 100 225	25 571 376
tmt_sym	726 713	2 903 835	28 657 615
boneS10	914 898	28 191 660	266 173 272
Emilia_923	923 136	20 964 171	1 633 654 176
audkiw_1	943 695	39 297 171	1 225 571 121
bone010_M	986 703	12 437 739	363 650 592
bone010	986 703	36 326 514	1 076 191 560
ecology2	999 999	2 997 995	35 606 934
thermal2	1 228 045	4 904 179	50 293 930
StocF-1465	1 465 137	11 235 263	1 039 392 123
Hook_1498	1 498 023	31 207 734	1 507 528 290
Flan_1565	1 564 794	59 485 419	1 451 334 747
G3_circuit	1 585 478	4 623 152	90 397 858

Был проведен ряд экспериментов на тестовых матрицах с использованием решателей MKL PARDISO [5] из Intel® Parallel Studio XE 2013 SP1 и MUMPS ver. 4.10.0 [10]. Во всех экспериментах использовались одина-

[1] Суперкомпьютер ННГУ «Лобачевский»: http://hpc-education.unn.ru/ru/ресурсы

ковые перестановки, полученные с помощью METIS [6]. Полученные результаты приведены на диаграммах ниже (рис. 2).

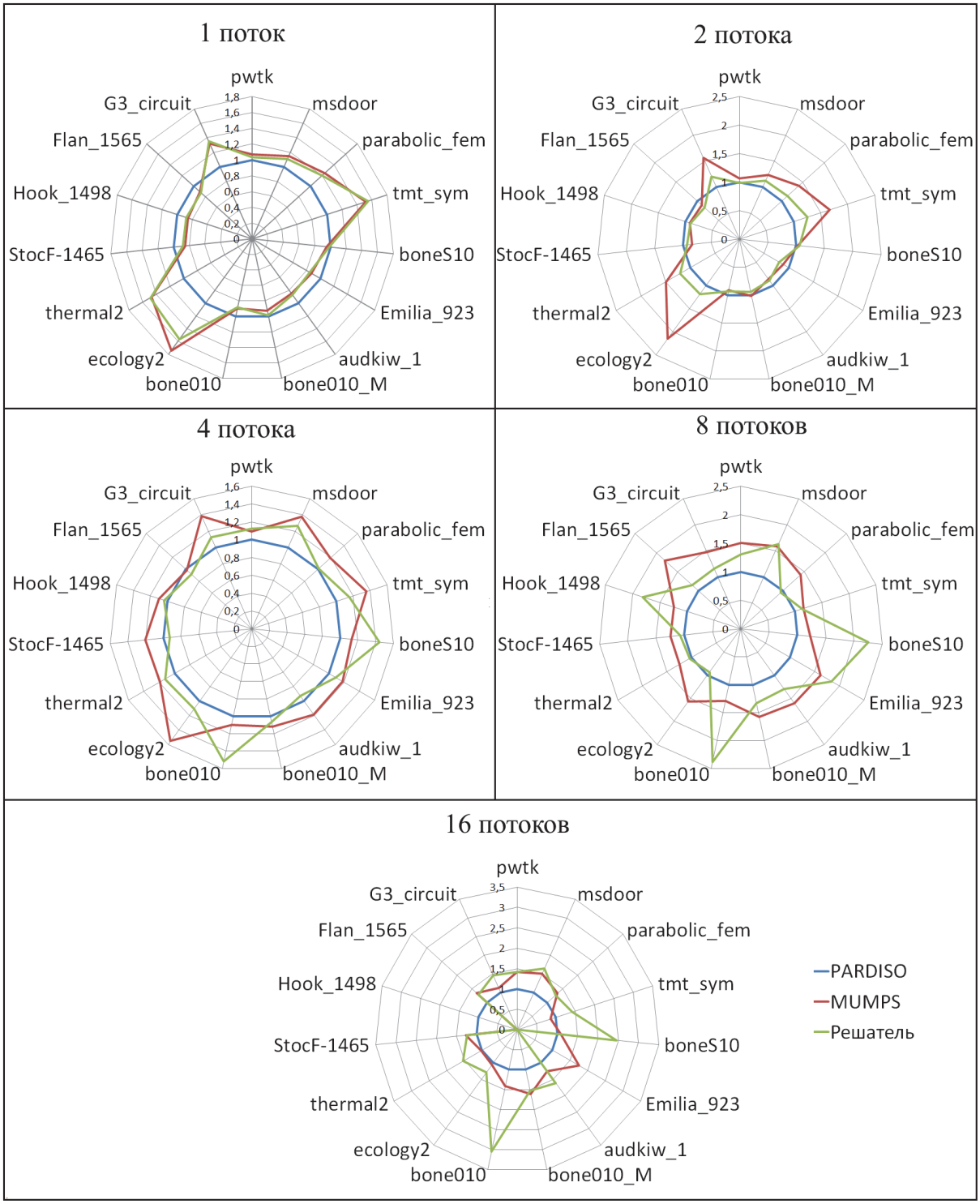


Рис. 2. Сравнение численных фаз решателей (по осям отложено время работы; за единицу взято время работы MKL PARDISO)

Из результатов можно сделать следующие выводы. Для запусков в 1 поток разработанный параллельный алгоритм показал схожие результаты по сравнению с MUMPS. На некоторых матрицах получен выигрыш

во времени работы до 16 % относительно MKL PARDISO. При запусках в 2 потока решатель авторов работает за сравнимое время относительно MKL PARDISO, при этом MUMPS значительно отстает на матрице ecology2. Для 4 потоков ситуация схожая, но времена работы решателей в большей степени сопоставимы. Для 8 и 16 потоков заметно значительное превосходство MKL PARDISO по сравнению с другими решателями.

Времена работы разработанного метода и MUMPS похожи, но стоит отметить что в большинстве случаев MUMPS выигрывает, в среднем на 34 %. Также заметим, что при запуске в 16 потоков MUMPS и решатель авторов не смогли корректно завершить работу на матрице Hook_1498, а также решатель авторов не отработал на матрице Emilia_923 в связи с нехваткой памяти.

Заключение

Основным результатом работы является программная реализация параллельного мутльтифронтального метода. В ходе работы выполнен анализ современных подходов к реализации и оптимизации параллельной численной фазы разложения Холецкого для разреженных матриц. Разработана новая динамическая схема распараллеливания разложения Холецкого, основанная на дереве исключения. Выполнено сравнение параллельной версии с широко распространенными аналогами MKL PARDISO и MUMPS.

Работа выполнена в лаборатории «Информационные технологии» ВМК ННГУ при поддержке гранта РФФИ №14-01-3145514.

Авторы благодарят И.Б. Меерова, А.Ю. Пирову, А.В. Сысоева и А.В. Линёва за полезные обсуждения и внимание к работе.

Список литературы

1. Comparison of Three Column-Based Distributed Sparse Factorization Schemes / C. Ashcraft, C. Eisenstat, J. Liu, A.A. Sherman // Yale univ new haven ct dept of computer science, 1990. – №. yaleu/dcs/rr-810.

2. Davis T., Hu Y. The University of Florida sparse matrix collection // ACM Trans. Math. Software. – 2011. – Vol. 38, No. 1 (Article 1).

3. Excellent J, Mohamed S. Introduction of shared-memory parallelism in a distributed-memory multifrontal solver// Research Report 82227. – Grenoble-Rhone-Alpes Research Center, 2013.

4. Geist G.A., Ng E. Task scheduling for parallel sparse Cholesky factorization //International Journal of Parallel Programming. – 1989. – Т. 18. – №. 4. – С. 291–314.

5. Intel Math Kernel Library Reference Manual. – URL: http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf.

6. 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 Minnecota, Department of Computer Science and Engeneering. – 2011. – URL: http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf.

7. L'Excellent J. Y. Multifrontal Methods: Parallelism, Memory Usage and Numerical Aspects : дис. – Ecole normale supérieure de Lyon. – ENS LYON, 2012.

8. Li X. Direct Solvers for Sparse Matrices. – URL:http://crd-legacy.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf.

9. Liu J. W. H. The multifrontal method for sparse matrix solution: Theory and practice //SIAM review. – 1992. – Т. 34. – №. 1. – С. 82–109.

10. 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]

11. 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.

12. Лебедев С. А., Козинов Е. А. Разработка нового решателя разреженных систем линейных уравнений // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XIII Всерос. конф. – Н. Новгород: Изд-во ННГУ, 2013. – С. 301–306.

А.В. Лобанов

ОАО «НИИ "Субмикрон"», Зеленоград

ИНТЕРФЕЙС ОТКАЗОУСТОЙЧИВОСТИ В ЗАМКНУТЫХ МНОГОКЛАСТЕРНЫХ СИСТЕМАХ

Рассматривается проблема организации интерфейса отказоустойчивости в замкнутой кластерной системе, параллельно решающей заданное множество взаимодействующих задач, для каждой из которых задан требуемый уровень отказоустойчивости. Также необходимый уровень отказоустойчивости задается для каждой среды межзадачного взаимодействия.

Введение

Рассматривается многокластерная система, в которой параллельно и отказоустойчиво решается множество взаимодействующих задач. Отказоустойчивость задачи обеспечивается ее репликацией путем параллельного решения данной задачи на нескольких ЦВМ, составляющих кластер (комплекс [1]) данной задачи, с обменом результатами и выбором из них правильного. Заданный уровень отказоустойчивости данной задачи определяет допустимое количество неисправных ЦВМ, при котором еще сохраняется возможность определения правильного результата ее решения в кластере. Задачи в системе взаимодействуют между собой путем посылки сообщений между соответствующими кластерами по определенным путям, составляющим среды данных посылок. Для каждой такой среды также задан требуемый уровень сбое- и отказоустойчивости посылки по ней сообщения.

В качестве модели системы применяется ориентированный граф (орграф), пронумерованные от 1 до n вершины которого отображают элементы системы (ЦВМ, коммутаторы, маршрутизаторы, концентраторы, устройства хранения информации и т.п.) с соответствующими номерами, а дуги – существующие каналы межэлементной связи. Задача определения структуры многокластерной системы в сети, в которой для каждой задачи выделяется кластер и для каждой взаимодействующей пары задач определяются соответствующие среды посылок, рассматривается в [1, 2].

Интерфейс отказоустойчивости замкнутой (необслуживаемой) системы составляют все аппаратурно-программные механизмы, вводимые в систему с целью обеспечения требуемых уровней сбое- и отказоустойчивости.

Интерфейс сбое- и отказоустойчивости многокластерных систем

Сложность механизмов интерфейса сбое- и отказоустойчивости многокластерных систем зависит от принимаемой модели неисправности элементов системы. Наиболее общей моделью неисправности является модель враждебной неисправности [3], при которой поведение неисправного элемента системы может быть произвольным, в том числе и подобным злонамеренному. Парирование враждебных неисправностей гарантирует парирование неисправностей всех других моделей.

Повышение сбое- и отказоустойчивости рассматриваемой системы может достигаться за счет дорогостоящего обеспечения отказоустойчивости входящих в нее ЦВМ путем применения в них n-модульной избыточности (резервирования) и мажорирования выходных значений всех избыточных модулей. Другой подход, более учитывающий сетевую особенность рассматриваемых систем (замкнутость системы, наличие большого

количества взаимосвязанных распределенных ЦВМ и возможность оперативного формирования из них требуемых вычислительных структур), состоит в репликации задач и введении в систему динамической избыточности, обеспечивающих: 1) парирование проявлений допустимых враждебных неисправностей за счет параллельного выполнения одной и той же задачи на нескольких ЦВМ с обменом полученными результатами и выбором из них правильного, 2) обнаружение и идентификацию по месту возникновения и типу (сбой, программный сбой, отказ) возникающих неисправностей, 3) восстановление целевой работы и исправления ошибочной информации после сбоев и программных сбоев, 4) реконфигурацию системы (с использованием запасных элементов) и восстановление целевой работы после отказов, 5) управляемую деградацию системы с возможным допустимым снижением характеристик вплоть до предельно заданной возможной конфигурации, 6) безопасный останов системы при невозможности построения такой конфигурации, 7) возможность перераспределения ресурсов системы для изменения соотношения производительность-достоверность между различными решаемыми задачами. Именно этот подход рассматривается в данной работе

Интерфейс сбое- и отказоустойчивости многокластерных систем содержит механизмы двух уровней: базового и основного. Базовый уровень включает механизмы синхронизации системы и ее взаимного информационного согласования. Синхронность обеспечивается путем организации в системе и непрерывной работы подсистемы единого системного времени, включающей средства как начальной, так и промежуточной синхронизации. Начальная синхронизация [4] осуществляется при начальном несинхронном включении различных ЦВМ системы и формирует путем взаимообмена сообщениями между включенными ЦВМ начальную конфигурацию системы в момент, когда эта конфигурация будет содержать достаточное количество исправных ЦВМ при условии, что среди ЦВМ конфигурации может иметься допустимое количество враждебно неисправных ЦВМ. Промежуточная синхронизация [5] обеспечивает на основе межмашинного взаимообмена сообщениями требуемую синхронность внутренних часов автономных ЦВМ, расходящихся из-за индивидуальных значений дрейфов этих часов и возникновения допустимых враждебных неисправностей.

Согласованность действий и принимаемых решений в различных ЦВМ и подсистемах гарантируется применением алгоритмов взаимного информационного согласования (ВИС). Достижимость ВИС составляет концептуальную основу создания отказоустойчивых алгоритмов для решения основных задач организации распределенных вычислений. В настоящее время разработано значительное число алгоритмов, различающихся по

постановкам задачи и критериям эффективности [3]. Целью всех этих методов являлось только достижение ВИС, и специальная задача обнаружения и идентификации проявлений неисправностей в процессе ВИС не ставилась. Однако исследуемая в настоящей работе задача организации сбое- и отказоустойчивых вычислений в многокластерных как полносвязных, так и неполносвязных системах на основе динамической избыточности требует разработки алгоритмов ВИС, которые вместе с достижением ВИС обеспечивали бы также обнаружение и идентификацию проявившихся в процессе ВИС враждебных неисправностей, что предотвращает накопление латентных неисправностей, одновременное проявление которых может привести к отказу всей многокластерной системы. Такие методы ВИС для однокластерных полносвязных систем предложены в работах [6–8]. В [9,10] представлены обоснованные методы ВИС для неполносвязных систем, а в [11] – метод ВИС для неполносвязных систем с обнаружением и идентификацией случившихся проявлений неисправностей. Задачи, связанные с обеспечением системного ВИС в многокластерных системах, рассматриваются в [1, 2, 12–14].

Основные механизмы обеспечения сбое- и отказоустойчивости на основе динамической избыточности для рассматриваемых однокластерных систем при возникновении допустимых враждебных неисправностей включают механизмы парирования допустимых враждебных неисправностей (гарантирования правильности выходной информации системы при возникновении допустимых неисправностей) [15–19], функционального диагностирования системы с обнаружением и идентификацией возникающих допустимых враждебных неисправностей в процессе целевой работы и тестового диагностирования однокластерных систем [15–24] и двухкластерных систем [25], тестового диагностирования подсистем и системы в целом [20–24, 26], восстановления целевой работы подсистем и системы в целом при возникновении программных сбоев и отказов, самоуправляемой реконфигурации и деградации кластеров и системы в целом, выполняемых также в условиях возникновения допустимых враждебных неисправностей [19, 26, 27].

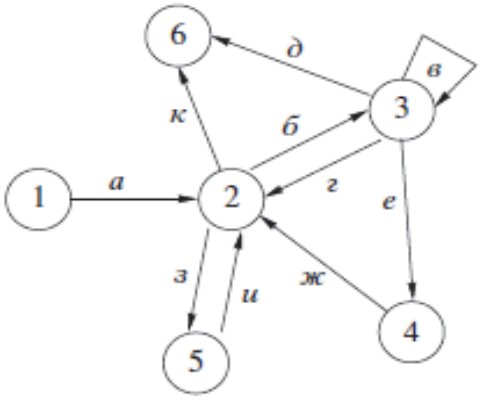


Рис. Граф диаграммы обобщенного алгоритма работы многозадачной многокластерной системы со сбое- и отказоустойчивостью

Граф диаграммы обобщенного алгоритма работы многозадачной многокластерной системы со сбое- и отказоустойчивостью на основе использования динамической избыточности (репликации задач, обнаружении и идентификации враждебных неисправностей и управляемой деградации) представлен на рисунке. Здесь вершины графа на диаграмме отображают состояния – выполняемые системой действия: 1) начальная проверка системы с целью выявления исправных элементов; 2) выбор в соответствии с заданием на конфигурацию системы начальной текущей рабочей конфигурации системы (ТРКС) – часть этой системы, которая включает все ЦВМ, каналы связи, непосредственно участвующие в выполняемых системой действиях, а также выбор ТРКС по результатам функционального и тестового самодиагностирования; 3) целевая работа по выполнению задания, основанная на репликации задач в создаваемых комплексах, межзадачных взаимообменах между комплексами, начальном обнаружении и идентификации случившихся проявлений неисправностей; 4) реконфигурация системы, восстановление необходимого вычислительного процесса с заданной достоверностью результатов вычислений в комплексах и в системе в целом за счет восстановления сбившихся ЦВМ и ЦВМ из горячего запаса; 5) включение необходимых ЦВМ из холодного запаса с целью их вовлечения в ТРКС и создания необходимого горячего запаса; 6) выполнение безопасного останова системы. Дуги обозначают события, по которым должны происходить переходы между состояниями: а) начальная проверка выполнена успешно; б) приемлемая структура ТРКС имеется и определена; в) обнаружена и идентифицирована по месту с возможной точностью допустимая совокупность сбоев; г) идентифицирована по месту и типу допустимая совокупность программных сбоев или отказов НО; д) идентифицированы недопустимые совокупности неисправностей или недопустимые последовательности неисправностей, или случилось непредусмотренное событие; е) по условиям применения системы есть возможность восстановления сбившихся и запасных ЦВМ и удовлетворен критерий необходимости восстановления; ж) процесс восстановления закончен, восстановленные ЦВМ переведены в ТРКС; з) требуется включение одной или нескольких ЦВМ из холодного запаса; и) включение ЦВМ из холодного запаса выполнено, и эти ЦВМ переведены в текущий горячий запас; к) выбор приемлемой ТРКС невозможен.

Заключение

Приведенный список литературы показывает, что для ряда задач по построению интерфейса сбое- и отказоустойчивости замкнутых многокластерных систем предлагаются оригинальные, теоретически обоснованные

решения. Однако значительно больше проблем и задач остаются открытыми. К ним относятся задачи снижения оценок сложности предлагаемых методов по объемам требуемых аппаратурной, временной и информационной избыточности, разработки приемлемых методов самоорганизации сбое- и отказоустойчивых параллельных взаимосвязанных вычислений на основе использования динамической избыточности, разработки и взаимной увязки всех необходимых архитектурных, аппаратурных и программных механизмов ее реализации, разработки методов моделирования и оценки эффективности таких систем, методов отладки и испытаний (включая инжекцию допустимых неисправностей и создание возможных нештатных ситуаций) как отдельных элементов и подсистем, так и системы в целом.

Список литературы

1. Ашарина И.В., Лобанов А.В. Выделение комплексов, обеспечивающих достаточные структурные условия системного взаимного информационного согласования в многокомплексных системах //Автоматика и телемеханика. – 2014. – № 6. – С. 115–131.
2. Ашарина И.В., Лобанов А.В. Выделение структурной среды системного взаимного информационного согласования в многокомплексных системах // Автоматика и телемеханика. – 2014. – № 8. – С. 146–156.
3. Генинсон Б.А., Панкова Л.А., Трантенгерц Э.А. Отказоустойчивые методы обеспечения взаимной информационной согласованности в распределенных вычислительных системах // Автоматика и телемеханика. – 1989. – № 5. – С. 3–18.
4. Лобанов А.В. Синхронизация и взаимное информационное согласование // Программирование. – 1997. – № 2. – С. 76–80.
5. Anceaume E., Puaut I. A taxonomy of clock synchronization algorithms: Research Report 1103 / Institut National de Recherche en Informatique et Syst`emes Al.eatoires (IRISA). – Rennes, France, July 1997. – URL: fttp://ftp.irisa.fr/techreports/1997/PI-1103.ps.gz.
6. Лобанов А.В. Взаимное информационное согласование с идентификацией неисправностей в распределенных вычислительных системах // Автоматика и телемеханика. – 1992. – № 4. – С. 137–146.
7. Лобанов А.В. Взаимное информационное согласование с идентификацией неисправностей на основе глобального синдрома // Автоматика и телемеханика. – 1996. – № 5. – С. 150–159.
8. Лобанов А.В., Сиренко В.Г., Гришин В.Ю. Взаимное информационное согласование в многомашинных вычислительных системах с обнаружением и идентификацией кратных враждебных неисправностей // Автоматика и телемеханика. – 2003. – № 4. – С. 123–133.

9. Лобанов А.В., Ашарина И.В., Мищенко И.Г. Взаимное информационное согласование в неполносвязных многомашинных вычислительных системах // Автоматика и телемеханика. – 2003. – № 5. – С. 190–198.

10. Ашарина И.В., Лобанов А.В. Взаимное информационное согласование в неполносвязных гетерогенных многомашинных вычислительных системах // Автоматика и телемеханика. – 2010. – № 5. – С. 133–146.

11. Лобанов А.В. Взаимное информационное согласование с обнаружением и идентификацией враждебных неисправностей в неполносвязных многомашинных вычислительных системах // Автоматика и телемеханика. – 2003. – № 6. – С. 175–186.

12. Лобанов А.В. Алгебраический подход к задаче выделения комплексов при организации сбое- и отказоустойчивых параллельных вычислений в сетях ЦВМ // Открытое образование. – 2011. – № 2 (86), Ч. 2. – С. 36–39.

13. Ашарина И.В. Алгебраический метод определения достаточной среды межкомплексной посылки при организации сбое- и отказоустойчивых параллельных вычислений в сетях ЦВМ // Открытое образование. – 2011. – № 2. – С. 29–32.

14. Лобанов А.В., Ашарина И.В. Временная оптимизация алгоритма системного взаимного информационного согласования в многокомплексных вычислительных системах// Открытое образование. – 2012. – № 2. – С. 32–35.

15. Лобанов А.В. Протокол отказоустойчивого обмена // Приборы и системы управления. – 1993. – № 7. – С. 8–11.

16. Лобанов А.В., Нахаев С.А. Обеспечение сбое- и отказоустойчивости в протоколе отказоустойчивого обмена // Приборы и системы управления. – 1993. – № 7. – С. 12–13.

17. Лобанов А.В. Распределенное мажорирование информации с обнаружением и идентификацией неисправностей // Автоматика и телемеханика. – 1997. – № 1. – С. 145–149.

18. Лобанов А.В. Организация сбое- и отказоустойчивых вычислений в полносвязных многомашинных вычислительных системах // Автоматика и телемеханика. – 2000. – № 12. – С. 138–146.

19. Лобанов А.В. Обнаружение и идентификация неисправностей в распределенных управляющих вычислительных системах с программно-управляемой сбое- и отказоустойчивостью // Автоматика и телемеханика. – 1998. – № 1. – С. 155–164.

20. Лобанов А.В. Обнаружение и идентификация «враждебных» неисправностей путем одновременного сочетания функционального и тестового диагностирования в многомашинных вычислительных системах // Автоматика и телемеханика. – 1999. – № 1. – С. 159–165.

21. Лобанов А.В., Сиренко В.Г., Гришин В.Ю. Функциональное диагностирование в распределенном системном диагностировании многомашинных вычислительных систем // Автоматика и телемеханика. – 2002. – № 1. – С. 152–158

22. Сиренко В.Г. Функциональное диагностирование процессов посылки информации в вычислительных системах при неизвестном исходном значении передаваемой информации // Автоматика и телемеханика. 2005. – № 11. – С. 135–154.

23. Сиренко В.Г. Метод локализации «враждебных» неисправностей в многомашинных вычислительных системах // Известия вузов. Электроника. – 2006. – № 3. – С. 38–43.

24. Лобанов В.А., Гришин В.Ю., Сиренко В.Г. Распределенное системное диагностирование враждебных неисправностей в неполносвязных многомашинных вычислительных системах // Автоматика и телемеханика. – 2005. – № 2. – С. 148–157.

25. Лобанов А.В. Организация сбое- и отказоустойчивой работы двухкомплексной многомашинной вычислительной системы // Автоматика и телемеханика. – 1998. – № 2. – С. 143–152.

26. Лобанов А.В. Процессы восстановления и управляемая деградация в многомашинных отказоустойчивых вычислительных системах // Электронная техника. Сер. 3. Микроэлектроника. – М.: Изд-во ЦНИИ «Электроника», 1997. – Вып. 1 (151). – С. 35–41.

27. Лобанов А.В. Динамическое распределение ЦВМ в многокомплексных вычислительных системах с программно-управляемой сбое- и отказоустойчивостью // Электронная техника. Сер.3. Микроэлектроника. – М.: Изд-во ЦНИИ «Электроника», 2000–2001. – Вып. 1 (154)–1(155). – С. 66–70.

А.Г. Масич[1], Г.Ф. Масич[1], В.П. Матвеенко[1], Г.Г. Тирон[2], И.А. Хохлов[3]

[1] Институт механики сплошных сред УрО РАН, Пермь
[2] ООО «Научно-производственное объединение «Импульс», Пермь
[3] Институт математики и механики УрО РАН, Екатеринбург

СКОРОСТНЫЕ СЕТИ И РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ В УРО РАН

Приведены реализованные сервисы научно-образовательной скоростной оптической магистрали на участке Пермь–Екатеринбург и их влияние на организацию распределенных вычислений в Уральском отделении РАН.

Введение

Текущая революция в науке – интенсивные потоки в эру больших данных и высокопроизводительных вычислений. Источниками больших данных являются установки, обсерватории и вычислительные модели. Движущей силой науки является end-to-end перемещение больших данных между оконечными системами. Способ перемещения больших данных – доведение оптики до оконечных систем. λ-каналы связи являются фундаментом распределенных данных и консолидированных вычислений. Наиболее яркие примеры: National LambdaRail (NLR) [1] объединяет региональные оптические сети США; GEANT2 (Gigabit European Academic Network Technology, http://www.geant2.net) – седьмое поколение Европейской научно-образовательной сети, объединяющей 28 национальных сетей.

В 2009 г. в Уральском отделении РАН инициированы работы по построению собственной научно-образовательной магистрали посредством «темного» оптического волокна и DWDM-технологии по трассе Архангельск – Сыктывкар – Ижевск – Пермь – Екатеринбург, названной «Инициатива GIGA UrB RAS» [2]. Основной мотив этих работ – решение проблем скоростей передачи данных Уральского отделения и создание условий построения распределенной вычислительной инфраструктуры для исследователей, которые работают по различным аспектам распределенных, параллельных, сетевых и облачных вычислений.

DWDM-тракт Пермь – Екатеринбург

Оптическая длина волокна Пермь – Екатеринбург равна 456 км. Используется стандартное одномодовое волокно с дисперсией, соответствующей Рек. ITU-T G.652. Вышеперечисленные классы услуг сети реализованы на оборудовании компании ECI-Telecom.

В мировой практике системы связи с длинными пролетами обычно используются при создании соединений между островами/городами и OADM-ответвлений к транснациональным сетям. В нашем случае применение технически самых совершенных, но дорогих решений неоправданно. Три регенерационных пункта, равноудаленно охватывающих точками присутствия участки трассы Пермь–Екатеринбург, обеспечивают длину между пунктами регенерации не более 130 км.

При расчете регенерационного участка DWDM-системы связи учитывалась величина требуемого OSNR, которая является ключевым параметром транспондера и от которого напрямую зависит максимальная длина однопролетной линии связи (и стоимость). Optical Signal-to-Noise Ratio (OSNR) – отношение сигнал/шум для оптического сигнала.

Оконечные узлы DWDM-тракта Пермь–Екатеринбург на платформах XDM-2000 установлены на площадках Президиум ПНЦ УрО РАН (Пермь, Ленина, 13а) и в суперкомпьютерном центре ИММ УрО РАН (Екатеринбург, Софьи Ковалевской, 16). Оптические мультиплексоры (MUX) на 16 λ-каналов обеспечивают возможность использования транспондеров со скоростью передачи 10–40 Гбит/с в каждой λ.

Регенерационное оборудование XDM-40 промежуточных узлов размещено в необслуживаемых регенерационных пунктах (НРП). Спецификация промежуточных узлов обеспечивает только оптическое усиление сигнала, однако закладываемое DWDM-оборудование предполагает возможность установки ROADM-мультиплексоров для выделения и маршрутизации λ-каналов не только на оконечных, но и на промежуточных узлах. Система мониторинга и управления магистральной ВОЛС (Sun Fire V245 и программное обеспечение LightSoft) располагается в центре управления магистралью в ИМСС УрО РАН (Пермь).

Каналы связи

Установленные в XDM-2000 транспондеры используют три λ-канала (каналы C21, C22 и C23 ITU-T частотного плана) со скоростью потоков по 10,7 Гбит/с (ITU-T G.709) в каждой λ, поддерживают механизм упреждающего исправления ошибок (FEC – Forward Error Correction) и имеют следующие характеристики портов ввода/вывода для подключения оконечного оборудования (рисунок): два Ethernet-порта по 10 Гбит/с (2x10GE), выделенных из λ-каналов C21 и C23, и 8 Ethernet-портов по 1 Гбит/с (8x1GE), сформированному из λ-канала C22. Такая совокупность портов образует слой гарантированных каналов связи по схеме точка-точка, позволяет использовать распространенные в оконечных системах 1GE порты и осуществить постепенный переход на скорость 10 Гбит/с. Оставшиеся 13 λ-каналов связи – потенциал масштабирования по 10–40 Гбит/с или когерентной DWDM 100 Гбит/с в каждой λ.

Для большей гибкости проводимых исследований на площадках в Перми и Екатеринбурге установлены L2 коммутаторы Cesar, имеющие по 24 порта 1GE и 4 порта 10GE, один из которых подключен к 10GE порту XDM2000. Эта совокупность портов образует слой не гарантированных каналов связи при нагрузке больше 10 Гбит/с.

Аналогичные по производительности и сетевым сервисам региональные научно-образовательные сети в Российской Федерации отсутствуют.

Реализован следующий класс услуг передачи данных, охватывающий три уровня эталонной модели взаимодействия открытых систем (L1-L3 OSI RM):

а) **λ-услуга** – предоставление фиксированного диапазона длин волн («лямбда»).

б) **FrameNet-услуга** – представление двухточечного или многоточечного Ethernet-канала связи между Ethernet-портами DWDM-магистрали;

в) **PacketNet услуга** – присоединение к Public Интернет.

λ-услуга обеспечивает точка-точка соединение на физическом уровне эталонной модели взаимодействия открытых систем (L1 OSI RM).

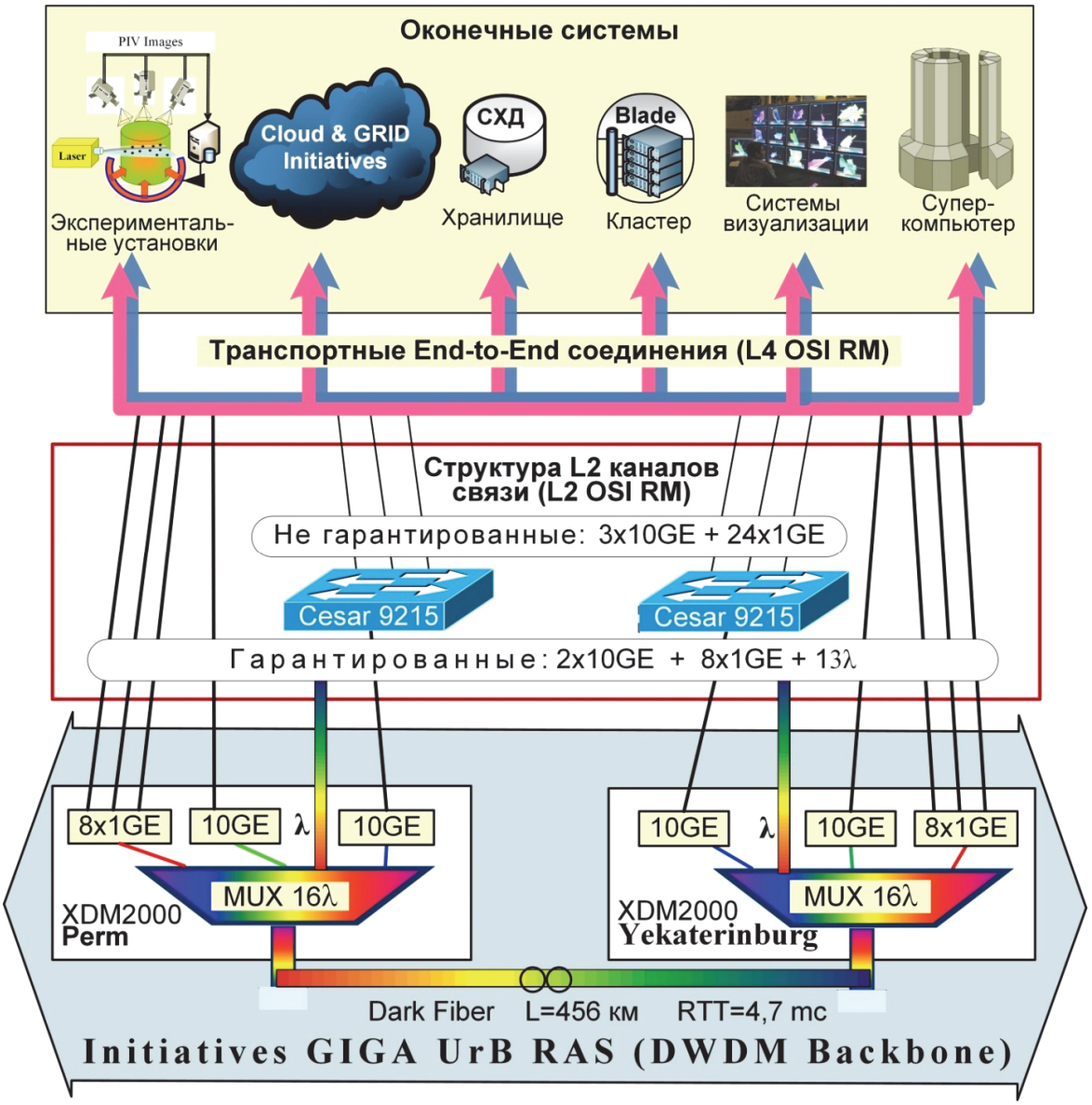


Рис. Оптическая инфраструктура Пермь – Екатеринбург

FrameNet-услуга предполагает использование доступных Ethernet GE (LX или на ZX)-интерфейсов, таких как 1-10 GE портов оконечного оборудования магистрали. Обеспечивается точка-точка и многоточечная доставка данных на Ethernet (канальном)-уровне (L2 OSI RM). Доступны два базовых типа услуг FrameNet:

– специализированная FrameNet-Ethernet-услуга, гарантирующая заказанную пропускную способность;

– неспециализированная FrameNet-Ethernet-услуга, не гарантирующая постоянную пропускную способность.

PacketNet-услуга предоставляет базовую услугу IP-маршрутизации для доступа к public internet, обеспечивая соединения на сетевом уровне (L3 OSI RM). Прием анонсов IP-сетей, как и анонсирование IP-сетей, выполняется согласно политикам маршрутизации, отраженным в RIPE NCC.

Приложения

Возможность использования скоростных каналов связи, соединяющих ресурсы ИМСС УрО РАН (Пермь) с ресурсами суперкомпьютерного центра ИММ УрО РАН (Екатеринбург), стимулировала ряд проектов, кратко описанных в порядке их появления.

– Выполнено сопряжение IP-сетей передачи данных Пермского научного центра (ПНЦ) с сетью Уральского отделения в Екатеринбурге на гарантированной скорости 1 Гбит/с и возможностью ее увеличения до 10 Гбит/с. Решена задача доступа всех институтов ПНЦ к public internet, централизованный доступ к которому осуществляется из Екатеринбурга. Институты КомиНЦ (Сыктывкар) подключены по арендуемому каналу 100 Мбит/с к сети ПНЦ.

– Разработан комплекс специализированного программного обеспечения middleware, обеспечивающий ввод в удаленный суперкомпьютер интенсивного потока экспериментальных данных [3]. Middleware используется при обработки потока данных от установки PIV в ИМСС (Пермь) на суперкомпьютере URAN в ИММ УрО РАН (Екатеринбург).

– На гарантированной скорости 10 Гбит/с соединены серверы хранения Supermicro в Перми и Екатеринбурге, образовав распределенную систему хранения данных (РСХ). В качестве основы для создания такой системы выбрано программное обеспечение dCache, которое активно используется в крупных международных проектах, таких как Worldwide LHC Computing Grid, European Grid Infrastructure, NorduGrid и др. [4]. Отличительной особенностью dCache является поддержка не только специфичных для грид-протоколов доступов к данным (SRM, dCap, xRoot), но и открытых протоколов (NFS, HTTP). В результате РСХ на основе dChache можно использовать как в составе грид, так и подключать к обычным вычислительным кластерам и персональным компьютерам.

– Соединены на гарантированной скорости 10 Гбит/с Ethernet интерконнект кластера (4,5 TFlops) на площадке ИМСС УрО РАН (Пермь) с ин-

терконнект суперкомпьютера URAN на платформе HP с ускорителями NVIDIA Tesla (260 TFlops). Сформированный распределенный интерконнект образует распределенную вычислительную среду нового поколения.

– Для разработки, тестирования и исследования каналов связи, протоколов, middleware и приложений созданы механизмы образования соединений на скорости 1–10 Гбит/с по оптической линии длиной 456 км или 900 км (петля) серверного оборудования.

Вышеперечисленная совокупность проектов иллюстрирует возможность создания принципиально новых инструментов проведения научных исследований посредством интеграции скоростных сетей с высокопроизводительными вычислительными системами.

Выводы

Опыт использования каналов связи DWDM тракта Пермь–Екатеринбург позволяет сформулировать следующие требования к научно-образовательным сетям. Сеть должна иметь трехкомпонентную структуру:

– опорная сеть для нужд проектов;

– исследовательская сеть для апробации сетевых технологий;

– пакетная сеть для скоростного доступа в Интернет.

Полоса пропускания сети должна постоянно увеличиваться не только в DWDM-тракте опорной сети, но и у всех узлов и систем, подключенных к скоростной магистрали.

Скоростной IP-доступ к глобальному Интернету является справедливым требованием образования международных коллабораций. Надежность сети является критическим параметром крупномасштабного эксперимента, когда отказов сети не должно быть.

Реализация инициативы GIGA UrB RAS на участке Пермь – Екатеринбург показала выполнимость подобного рода региональных проектов в научно-образовательном сообществе РФ.

Работа выполнена при поддержке РФФИ (грант № 14-07-96001-р_урал_а) и Уральского отделения РАН (код проекта 12-П-1-2012).

Список литературы

1. Silveste J. National LambdaRail (NLR) – Packet, Wave and Frame Services // GLIF 2006, Akihabara, Japan, September 11, 2006.

2. Масич А.Г., Масич Г.Ф. Инициатива GIGA UrB RAS // Вычислительные технологии. – Т. 13; Вестник КазНУ им. Аль-Фараби. Математика, механика, информатика. – №3 (58) (по материалам междунар. конф. «Вы-

числительные и информационные технологии в науке, технике и образовании». – Казахстан, Алматы). – 2008. – Ч. II. – С. 413–418.

3. Щапов В.А., Масич А.Г., Масич Г.Ф. Модель потоковой обработки экспериментальных данных в распределенных системах // Вычислительные методы и программирование. – 2012. – Разд. 2. – С. 139–145.

4. Distributed storage system with dCache / E.Y. Kuklin, A.V. Sozykin, A.Y. Bersenev, G.F. Masich // Distributed Computing and Grid-technologies in Science and Education: the 6th International Conference. – Dubna, 2014.

Г.Ф. Масич[1], А.Г. Масич[1], В.А. Щапов[1,2], С.Р. Латыпов[1], А.В. Созыкин[3], Е.Ю.Куклин[3]

[1]Институт механики сплошных сред УрО РАН, Пермь
[2]Пермский национальный исследовательский политехнический университет, Пермь
[3]Институт математики и механики УрО РАН, Екатеринбург

АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ УрО РАН

Описана архитектура распределенной вычислительной среды, состоящей из объединенных скоростной оптической сетью вычислителей и систем хранения суперкомпьютерного центра ИММ УрО РАН (Екатеринбург) и ИМСС УрО РАН (Пермь).

Введение

Известно, что суперкомпьютер представляет собой массивно параллельную вычислительную систему с распределенной памятью, включающей множество вычислительных узлов, соединенных между собой несколькими независимыми сетями (interconnect – интерконнект, например, InfiniBand/PCIe/Ethernet и т.д.), которые далее по тексту будем называть внутренний интерконнект. Совокупность внутренних интерконнектов суперкомпьютеров и скоростной сети, их объединяющей, назван нами «распределенный интерконнект». В распределенном интерконнекте основной идеей является параллелизм λ-каналов связи в оптической сети со спектральным управлением каналов.

Зарубежным прототипом является DAS-4 (Distributed ASCI Supercomputer)[1] – распределенный ASCI-суперкомпьютер из шести территориально распределѐнных разнородных кластеров в Нидерландах, расположенных в голландских университетах и используемых для проведения Computer Science-исследований. ASCI является аббревиатурой от Advanced Schoolfor Computing and Imaging – голландская магистерская школа, основанная в

1993 г. и аккредитованная в Нидерландской королевской академии наук и искусств (Royal Netherlands Academy of Arts and Sciences). DAS-4 разработана ASCI и финансируется NWO / NCF (Нидерландская организация по научным исследованиям). DAS-4 использует выделенный глобальный интерконнект, связывающий внутренние интерконнекты кластеров по скоростным выделенным λ-каналам, предоставляемым сетью SURFnet. SURFnet – голландская NREN, которая в рамках проекта StarPlane предоставила для DAS-4 выделенные 10 Гбит/с λ-каналы DWDM-магистрали SURFnet.

Компоненты распределенной вычислительной среды УрО РАН

Распределенная вычислительная среда УрО РАН базируется в двух центрах обработки данных (ЦОД): в ИМСС УрО РАН (Пермь) и ИММ УрО РАН (Екатеринбург).

В ЦОД ИММ располагаются суперкомпьютер «УРАН» пиковой производительностью 225.85 Тфлопс и три сервера распределенной системы хранения данных (РСХД) производства компании Supermicro.

В ЦОД ИМСС располагается вычислительный кластер Triton пиковой производительностью 4,5 Тфлопс и один сервер распределенной системы хранения данных.

Коммуникационная среда распределенной системы, показанной на рис. 1, сформирована посредством Ethernet-коммутаторов, соединенных

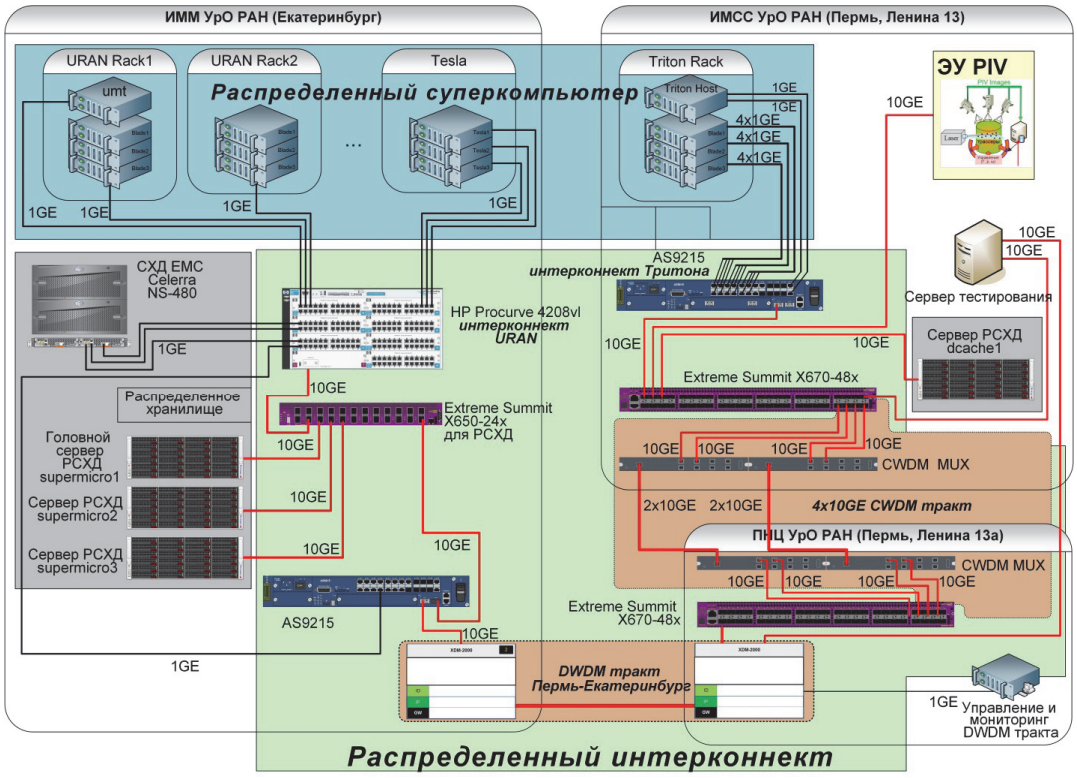


Рис. 1. Архитектура распределенной вычислительной среды УрО РАН

каналами связи DWDM тракта Пермь–Екатеринбург (30 Гбит/с) научно-образовательной сети GIGA URAL [2]. CWDM-тракт ИМСС-ПНЦ соединяет на скорости 40 Гбит/с вычислительные ресурсы ИМСС с местом окончания DWDM-тракта в Перми. Для большей гибкости проводимых исследований на площадках установлены L2-коммутаторы ECI AS9215 и L3-коммутаторы Extreme Summit X670-48x, образующие в совокупности с Ethernet-портами DWDM мультиплексоров гарантированные и негарантированные каналы связи 1–10 Гбит/с.

Архитектура сети передачи данных распределенного интерконнекта

Вычислительный кластер Triton состоит из трех базовых блоков HP BladeSystem 7000, каждый из которых состоит из 16 серверов HP Proliant BL 460c. Вычислительные узлы кластера для работы используют две сети передачи данных.

Основной сетью обмена данными MPI является InfiniBand 4xDDR пропускной способностью 20 Гбит/с. Дополнительная сеть Ethernet с пропускной способностью 1 Гбит/с предназначена для управления потоком задач и монтирования файловых систем на вычислительные узлы.

Каждый базовый блок оборудован встроенным Ethernet-коммутатором, к которому на скорости 1 Гбит/с подключен каждый вычислительный узел (см. рис. 1). Коммутаторы базовых блоков подключены к внешнему коммутатору ECI AS9215 при помощи 4 агрегированных по технологии Link Aggregation Control Protocol (LACP) соединений 1 Гбит/с (суммарная пропускная способность агрегированного канала – 4 Гбит/с; переподписка 1:4). Коммутатор ECI AS9215 подключен к магистральной сети на скорости 10 Гбит/с.

Суперкомпьютер «УРАН» Суперкомпьютерного центра ИММ УрО РАН в Екатеринбурге имеет аналогичную архитектуру.

Распределенный интерконнект соединяет Ethernet-интерконнект суперкомпьютера «УРАН» и сервера хранения Supermicro в ИММ УрО РАН (Екатеринбург) с Ethernet-интерконнектом вычислительного кластера Triton и сервер хранения Supermicro в ИМСС УрО РАН (Пермь) на скорости 2·10 Гбит/с.

Ethernet-интерконнекты суперкомпьютеров и узлы РСХД подключены к единой сети передачи данных и находятся в одном L2-сегменте сети, за исключением вычислительных узлов суперкомпьютера «УРАН», которые на текущий момент находятся в другом L2-сегменте, трафик между которыми маршрутизируется через управляющий узел суперкомпьютера «УРАН» с ограничением по скорости передачи данных 1 Гбит/с.

Адресация. В целях унификации был принят единый план распределения IP-адресов. Для основных Ethernet-интерфейсов была выделена подсеть 10.128.0.0/11. Диапазон адресов 10.128.0.0/16 выделен под общие нужды. Для вычислительных узлов кластера зарезервированы подсети 10.128+X.0.0/16, X >=1.

В настоящее время вычислительные узлы кластера Triton используют диапазон адресов 10.129.0.0/16, а узлы системы хранения данных располагаются в подсети 10.128.1.0/24.

Программная инфраструктура суперкомпьютеров

Суперкомпьютеры ИММ и ИМСС работают под управлением операционных систем Scientific Linux 6.5 архитектуры x86_64. Установлены одинаковые версии компиляторов GCC и MPI-библиотек, что позволит переносить исполняемые файлы приложений между вычислительными кластерами ИММ и ИМСС и использовать MPI для обмена сообщениями между экземплярами параллельных приложений на разных вычислительных кластерах. В каждом вычислительном кластере используется система пакетной обработки заданий SLURM, обеспечивающая возможность построения метапланировщика потока задач.

Распределенная система хранения данных

Узлы РСХД расположены в двух ЦОД: в ИММ УрО РАН г. Екатеринбурга (3 сервера) и в ИМСС УрО РАН г. Перми (1 сервер). В аппаратной платформе используются серверы производства Supermicro и 10GE сетевые карты Intel. РСХД построена на базе программного обеспечения dCache [3] из проекта European Middleware Initiative.

dCache ориентирован на хранение больших объемов экспериментальной информации. Он позволяет создавать несколько копий файлов для выравнивания нагрузки на узлы хранения, повышения надежности и оптимизации пути доступа к файлам. Так, межузловая репликация позволит узлу в ИМСС хранить локальную копию часто запрашиваемых у него данных, что приведет к уменьшению времени доступа к данным и уменьшит загрузку магистрального канала связи между вычислительными центрами. С другой стороны, распределение данных по разным серверам позволит избежать перегрузки одного узла.

dCache поддерживает большой набор протоколов для доступа к своему файловому пространству. Вместе с распространенными стандартными протоколами FTP, WebDAV, NFS могут использоваться грид-протоколы SRM и GRIDFTP, а также собственный протокол dCache DCAP.

В настоящее время вычислительные узлы суперкомпьютеров подключены к системе хранения по протоколу NFS версии 4.1 с использованием Parallel NFS. Parallel NFS позволяет организовать прямые обращения вычислительных узлов к узлам хранения для передачи данных. В этом случае NFS-сервер отвечает только за управление метаданными и не передает содержимое файлов, что является узким местом традиционных NFS-серверов. Предварительная загрузка данных из внешних систем возможна также по протоколам FTP, SCP или GridFTP.

Исследование производительности РСХД

На рис. 2 показаны результаты сравнения эффективности алгоритмов управления перегрузкой TCP при передаче данных между серверами РСХД, находящимися на площадках в Перми и Екатеринбурге. Наилучшие результаты показали алгоритмы bic и illinois. Причина, по которой алгоритмы не смогли достигнуть скоростей передачи данных, больших 6 Гбит/с, в настоящий момент не ясна. При этом на вдвое более протяженном тестовом кольце ИМСС УрО РАН, подключенном к серверам, оборудованным сетевыми картами Mellanox, достигается скорость передачи данных более 9 Гбит/с.

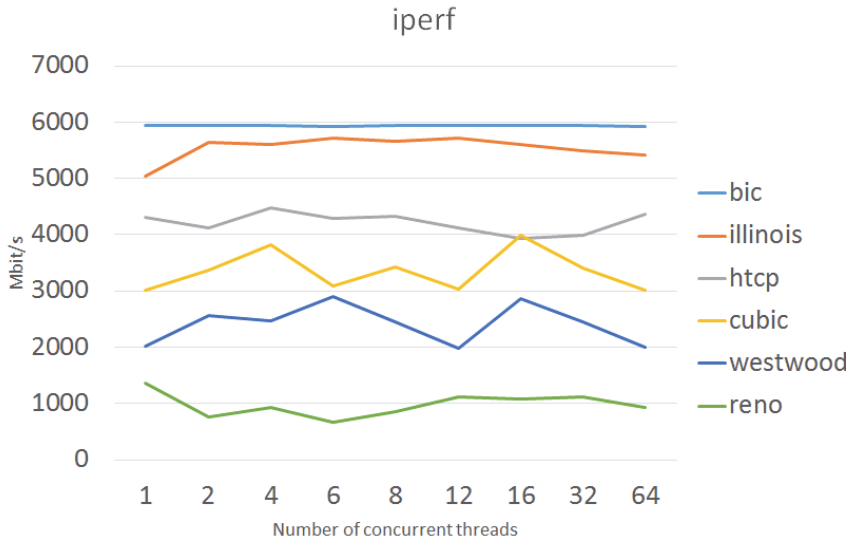


Рис. 2. Сравнение эффективности алгоритмов управления перегрузкой TCP между серверами Supermicro на площадках в ИМСС и ИММ

На рис. 3 показана производительность записи на РСХД клиентом в ИМСС до и после включения сервера dCache в ИМСС. Наличие близко расположенного к клиенту локального узла хранения позволяет увеличить скорость записи, так как dCache передает часть запросов на локальный сервер.

Рис. 3. Производительность записи на РСХД клиентом
в ИМСС до и после включения сервера dCache в ИМСС

На рис. 4 показана производительность записи на РСХД клиентом в ИМСС при включенной и выключенной репликации. Из графиков следует, что репликация не влияет на производительность, так как трансфер данных производится только при малой нагрузке на канал связи.

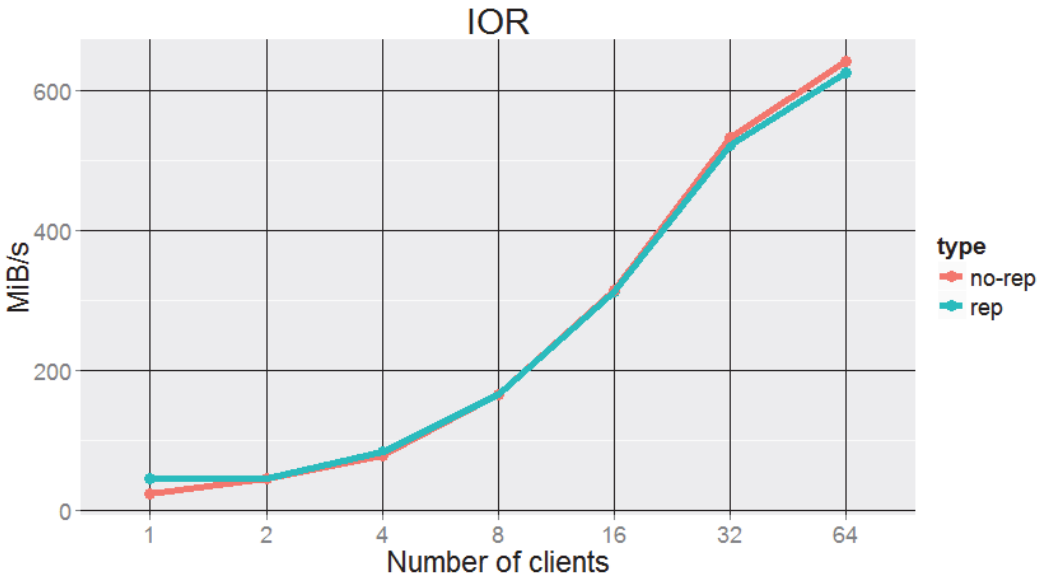


Рис. 4. Производительность записи на РСХД клиентом
в ИМСС при включенной и выключенной репликации

Заключение

Единое сетевое и системное администрирование распределенной вычислительной среды в УрО РАН, включая управление потоком задач, политику доступа к сервисам, мониторинг и диагностику, позволили реализовать приведенные архитектурные решения.

Создаваемая инфраструктура вычислительной среды предназначена для исследования различных аспектов параллельных, распределенных, сетевых и облачных вычислений. Основной целью разрабатываемых для такой среды приложений является обработка интенсивных потоков экспериментальных и визуализационных данных. Так, платформа распределенных вычислений позволит проводить их обработку в реальном времени, а распределенная система хранения данных – сохранять результаты обработки и исходные данные для последующего анализа.

Исследование проводится при поддержке грантов РФФИ №14-07-96001 и УрО РАН №12-П-1-2012.

Список литературы

1. Distributed ASCI Supercomputer. – URL: http://www.cs.vu.nl/das4/clusters.shtml

2. Масич Г.Ф., Масич А.Г. От «Инициативы GIGA UrB RAS» к Киберинфраструктуре УрО РАН // Вестник Пермского научного центра УрО РАН. – Пермь, 2009. – № 4. – С. 41–56.

3. Distributed storage system with dCache / E.Y. Kuklin, A.V. Sozykin, A.Y. Bersenev, G.F. Masich // Distributed Computing and Grid-technologies in Science and Education: the 6th International Conference. – Dubna, 2014.

Е.В. Мехоношина, В.Я. Модорский

Пермский национальный исследовательский политехнический университет,
Центр высокопроизводительных вычислительных систем, г. Пермь

ГАЗОДИНАМИЧЕСКИЙ РАСЧЕТ МОДЕЛЬНОГО КОМПРЕССОРА

Статья посвящена численному моделированию работы газодинамических процессов в модельном компрессоре с использованием системы инженерного анализа ANSYS 15.0. Проведена оценка времени расчета при работе на высокопроизводительном вычислительном комплексе ПНИПУ.

Современным средством исследования газодинамических процессов в компрессорах является вычислительный эксперимент [1–4].

Рассмотрена модельная задача. Построены физическая, математическая и сеточная модели, составлена методика проведения вычислительных экспериментов.

Подготовка, обеспечение и проведение вычислительных экспериментов. Расчеты проведены в системе компьютерного инженерного анализа ANSYS 15.0 с использованием мощностей высокопроизводитель-

ного вычислительного комплекса ПНИПУ, имеющего следующие технические характеристики:

– 112 вычислительных узлов на 128 четырехядерных процессорах AMD Opteron «Barcelona-3» (512 ядер);

– 48 восьмиядерных процессоров «IntelXeon E5-2680» (384 ядра) и 12 вычислительных модулей GPU NVIDIA Tesla M 2090;

– пиковая производительность 24 TFlops;

– производительность в тестовом пакете Linpack 78 % от пиковой;

– объем системы хранения информации 24 ТБ;

– объем оперативной памяти 128 Гбайт/узел.

Сформирована физическая модель:

– процессы рассматриваются в трехмерной постановке;

– рассматривается течение идеального газа постоянного состава с заданными свойствами;

– химические процессы не учитываются;

– поток однофазный;

– расчеты проведены без учета гравитации;

– стенки конструкции не поглощают и не выделяют тепло;

– коэффициенты теплоемкости не зависят от абсолютной температуры.

В соответствии с выбранной физической моделью разработана математическая модель газодинамического процесса, которая базируется на законах сохранения массы, импульса, энергии, уравнении состояния совершенного сжимаемого газа, начальных и граничных условиях.

В наиболее общем случае для задач газовой динамики требуется решить систему из четырех независимых уравнений – систему уравнений Навье–Стокса[1].

Уравнение неразрывности (сохранения массы)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{V} \right) = 0, \tag{1}$$

где ρ – плотность газа; t – время; V – вектор скорости газа.

Уравнение количества движения (сохранения импульса)

$$\frac{\partial \left(\rho \vec{V} \right)}{\partial t} + \nabla \left(\rho \vec{V} \otimes \vec{V} \right) = -\nabla p + \nabla \cdot \tau + \overrightarrow{S_M}, \tag{2}$$

где τ – тензор напряжений, записываемый в виде

$$\tau = \mu [\nabla \vec{V} + \left(\vec{V} \right)^T - \frac{2}{3} \delta \nabla \cdot \vec{V} \Big],$$

δ – дельта-функция Кронекера,

$$\delta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

p – давление; $\overrightarrow{S_M}$ – источниковый член для импульса.

Уравнение энергии (сохранения энергии)

$$\frac{\partial(\rho h_{tot})}{\partial t} - \frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{V} h_{tot}\right) = \nabla\left(\lambda \nabla T\right) + \nabla\left(\vec{V} \cdot \vec{\tau}\right) + \vec{V}\overrightarrow{S_M} + S_E, \qquad (3)$$

где h_{tot} – полная энтальпия газа,

$$h_{tot} = h_{\text{stat}} + \frac{\vec{V}^2}{2},$$

h_{stat} – статическая энтальпия газа,

$$h_{\text{stat}} = h_{\text{stat}}\left(T, p\right),$$

λ – коэффициент теплопроводности газа; S_E – источниковый член для энергии.

Уравнение состояния

$$\rho = \rho\left(T, p\right). \qquad (4)$$

Для замыкания в систему уравнений (1)–(4) необходимо добавить уравнения для модели турбулентности, чтобы система в целом могла быть решена. При проведении вычислительных экспериментов применялась SST-модель турбулентности. Расчетная схема – трехмерная.

При расчетах выделены две зоны: статическая (вблизи стенок корпуса) и динамическая (вблизи лопастей рабочего колеса). На рис. 1,*а* представлена твердотельная модель для проведения газодинамического анализа.

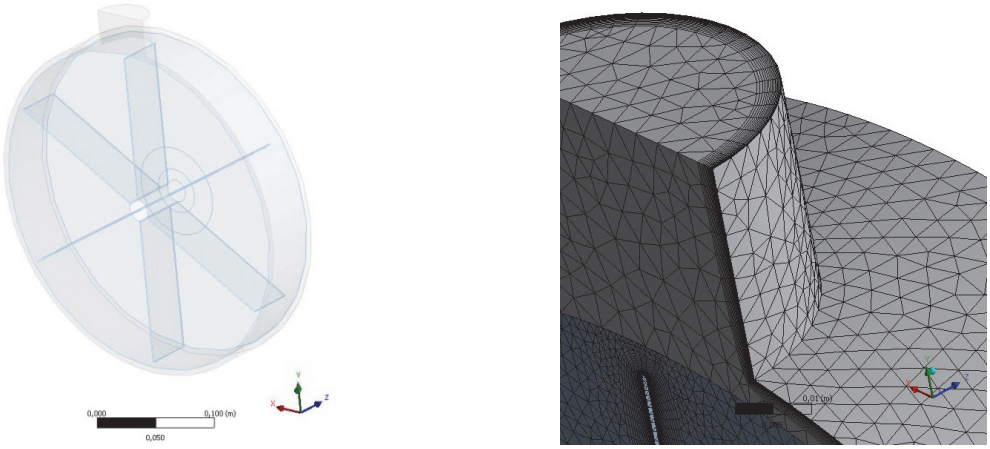
Для каждой зоны построена своя сетка. В расчете используется конечно-элементная сетка, состоящая из 1,7 млн ячеек. На рис. 1, *б* представлен фрагмент сеточной модели.

В результате расчетов получены поля давления и скорости газодинамического потока. Проведена оценка влияния скорости вращения рабочего колеса на поля давлений и скоростей потока в газодинамическом тракте.

Была проведена оценка масштабируемости расчетов. Расчеты проводились с использованием 4, 8, 16, 32, 64, 128 ядер центральных процессоров. Результаты представлены на рис. 2.

Из рис. 2 видно, что расчет данной модели выгодно проводить на числе ядер, не превышающем 64. Количество элементов расчетно-

сеточной модели должно быть увеличено, тогда, возможно, будет достигнуто масштабирование на большее число вычислительных ядер.



а *б*

Рис. 1. Модель газодинамического тракта:
а – твердотельная; *б* – сеточная (фрагмент)

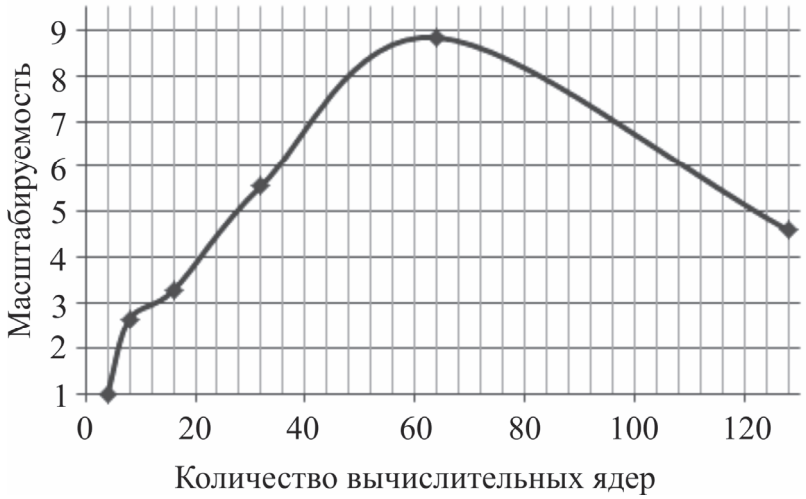


Рис. 2. Масштабируемость газодинамического расчета

Исследование выполнено за счет гранта Российского научного фонда (проект №14-19-00877).

Список литературы

1. Модорский В.Я., Соколкин Ю.В. Газоупругие процессы в энергетических установках. – М.: Наука, 2007 – 176 с.

2. Решение инженерных задач на высокопроизводительном вычислительном комплекс Пермского национального исследовательского политехнического университета / под ред. В.Я. Модорского. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2014. – 314 с.

3. Численное моделирование кавитационных эффектов при действии вибрации / Д.Ф. Гайнутдинова, А.В. Козлова, В.Я. Модорский, Е.В. Мехоношина // Научно-технический вестник Поволжья. – 2013. – № 6. – Казань, 2013. – 526 с.

4. Исследование влияния вибраций и давления на возникновение кавитации / Е.В. Мехоношина, В.Я. Модорский, А.В. Козлова, Д.Ф. Гайнутдинова // Научно-технический вестник Поволжья. – 2013. – № 6. – Казань: Научно-технический вестник Поволжья, 2013. – 526 с.

Ю.С. Митрохин

Удмуртский государственный университет, г. Ижевск

МОДЕЛИРОВАНИЕ ПРОЦЕССА ПЛАВЛЕНИЯ ТВЕРДЫХ ТЕЛ НА СУПЕРКОМПЬЮТЕРАХ ФИРМЫ INTEL

Методом первопринципной молекулярной динамики (ab initio MD) моделируется фазовый переход твердое тело–жидкость на современном многоядерном компьютере с использованием современных пакетов CP2K, CPMD, VASP, SIESTA, В качестве объекта исследования выбраны двойные и тройные сплавы на основе Ni_3Al. Результаты моделирования дают температуру плавления, слегка превышающую экспериментальное значение, в то время как классические методы дают значительное превышение расчетного значения по сравнению с экспериментом.

Методом первопринципной молекулярной динамики (ab initio MD) моделируется фазовый переход твердое тело–жидкость на современном многоядерном компьютере с использованием современных пакетов CP2K, CPMD, VASP, SIESTA, В качестве объекта исследования выбраны двойные и тройные сплавы на основе Ni_3Al. Результаты моделирования дают температуру плавления, слегка превышающую экспериментальное значение, в то время как классические методы дают значительное превышение расчетного значения по сравнению с экспериментом.

Моделирование физических и химических процессов в настоящее время приобретает все большее значение в связи с бурным развитием вычислительной техники. Центральное место в этих исследованиях занимает метод молекулярной динамики (MD). Первая работа по моделированию методом молекулярной динамики была выполнена в 1957 г. [1], вскоре после появления первых больших компьютеров того времени (UNIVAC и IBM 704), на которых был установлен первый транслятор с языка FORTRAN. До этого уже был накоплен необходимый теоретический материал, но именно появле-

ние транслятора с языка высокого уровня FORTRAN на самых мощных компьютерах того времени позволило писать и отлаживать большие программы за разумное время при разумных затратах на программирование. Поэтому 1957 г. считается началом эры MD и моделирования этим методом различных процессов в физике химии и биологии.

Одним из наиболее трудных объектов для моделирования является процесс фазового перехода, в частности, процесс плавления в твердом теле. Это связано как с чисто техническими трудностями (большое количество машинного времени), так и с методологическим трудностями. Дело в том, что классический двухчастичный потенциал типа Леннарда–Джонса или Морзе [2] в принципе не может описать многочастичное взаимодействие атомов в твердом теле или жидкости. Современные многочастичные потенциалы типа Саттона–Чена [3] частично решают эту проблему, но тем не менее дают точку плавления на 300–500 градусов выше наблюдаемого значения. Это явление известно в MD как эффект перегрева. В значительной мере эту проблему удалось решить после появления пионерской работы Car Parrinello [4], в которой был предложен метод первопринципной молекулярной динамики (ab initio MD). Часто этот метод сокращенно именуют CPMD. Суть этого метода достаточно проста, он основан на теореме Геллман-Фейнмана [5], которая гласит, что сила, действующая на каждый конкретный атом, определяется первой производной от электронной плотности окружающей этот атом. Таким образом, для правильного вычисления сил, действующих на атомы, нужно знать электронную плотность, создаваемую этими атомами. Она, в свою очередь, зависит он расположения атомов в расчетной области. Таким образом, в методе CPMD задействованы одновременно два метода – классический метод MD и квантовый метод расчета электронной плотности на основе уравнения Кона–Шэма [6]. Последний из них требует около 99 % машинного времени.

До появления современных многопроцессорных вычислительных систем число атомов в методе CPMD составляло несколько десятков. Этого, конечно, было недостаточно для набора хорошей статистики. Тем не менее этот метод давал более точные значения для температуры плавления как в чистых металлах, так и в сплавах. На современных параллельных кластерах, насчитывающих тысячи узлов, с использованием современных пакетов программ (VASP, CPMD, SIESTA, CP2K и др) число атомов в расчетной области могло доходить до 1000 и более. Правда, затраты машинного времени даже на самых мощных суперкомпьютерах составляли десятки и сотни часов. Очевидно, что такие затраты могут выходить за рамки разумных пределов. Однако ценность методов CPMD состоит в том, что

они позволяют моделировать процессы, которые трудно воспроизвести и невозможно моделировать другими методами. Например, в литературе появились работы по моделированию процессов, происходящих в земной мантии на большой глубине и при больших температурах и давлениях [7]. Метод CPMD также используется для моделирования процессов, происходящих при ядерных взрывах [7].

В лаборатории параллельных вычислений УдГУ автором данной статьи выполнены работы по моделированию процессов плавления двойных (Ni_3A) и тройных (Ni_3Al-X) сплавов на параллельном кластере УдГУ и на вычислительном комплексе, установленном в лаборатории математического моделирования. Этот комплекс имеет 64 ядерных процессоров XEON фирмы Intel, т.е. всего 24 ядра. Для решения научно-технических задач было решено установить на этот сервер достаточно большой объем оперативной памяти – 99Гб. Это решение полностью оправдало себя при решении задач моделирования методом ab initio MD. Кроме этого, в УдГУ еще имеется параллельный кластер также на процессорах XEON фирмы Intel. Он имеет 56 ядер (7 узлов по 8 ядер), но мало оперативной памяти (3 Гб на каждом из 7 узлов). Кластер относится к машинам с распределенной памятью. При этом важное значение имеет наличие на кластере одной из самых скоростных систем связи между узлами: шины infiniband. Однако если на данном кластере нет параллельной файловой системы, как в нашем случае, то это приводит к тому, что некоторые программы не могут записать большой файл, рассчитанный на разных рабочих узлах (slave) на центральном узле (master). В результате программа или стоит, или заканчивается аварийно. Другая проблема машин с распределенной памятью состоит в том, что при передаче больших файлов (десятки Гб) по сети с рабочих узлов на центральный сеть оказывается перегруженной. Это может привести к проблемам для других пользователей суперкомпьютера и к значительному замедлению расчета. С такой ситуацией автор столкнулся при работе на суперкомпьютере в Институте математики и механики – ИММ УрО РАН г. Екатеринбурга. Эти проблемы не возникают на машинах с общей памятью. Компьютер в лаборатории математического моделирования УдГУ относится к классу машин с общей памятью. Поэтому большинство расчетов было выполнено на этом компьютере.

Для создания параллельных программ использовались системы MPI и OpenMP. Такая комбинация систем параллельного программирования позволяет достигнуть максимальной эффективности, она широко используется на многих современных суперкомпьютерах. Если использовать только систему MPI, то даже на машинах с общей памятью, тратится зна-

чительное время на передачу информации между различными параллельными процессами. Например, при запуске задачи на всех 24 ядрах время вычисления одного шага в CP2K было в 2 раза больше, чем при запуске той же задачи на 12 ядрах при 2 тридах (thread). Максимальная производительность достигалась на 6 ядрах при 4 тридах. Это связано с тем, что такая комбинация MPI и OpenMP лучше всего соответствует архитектуре данного компьютера. Он имеет 6 узлов по 4 ядра на каждом.

Для моделирования сплава Ni_3Al были выбраны два варианта: 108 и 256 атомов в расчетной области. Второй вариант требовал примерно в 5 раз больше машинного времени, но давал заметно лучшую статистику. Были выполнены пробные расчеты с использованием разных пакетов программ: CP2K, CPMD, VASP и SIESTA. Наилучшие результаты были получены при работе с пакетом CP2K. Этот пакет разрабатывался в Германии на протяжении 10 лет. Он включает в себя почти все известные к настоящему времени варианты молекулярной динамики, как классические, так и современные варианты ab initio MD. Во всех этих пакетах используется приближение первопринципного псевдопотенциала. Основная идея этого метода состоит в том, что из расчета исключаются остовные электроны атомов, что значительно уменьшает вычислительные затраты, так как основных электронов в 2–3 раза больше, чем валентных.

Исходное положение атомов соответствовало ГЦК-решетке, использовался NPT-ансамбль с термостатом Нозье-Гувера. Нагревание проводилось с интервалом в 100 градусов, начиная с 300 и до 2000 К. Для релаксации выполнялось 1000 шагов, для набора статистики выполнялось еще 2000 шагов. В результате вычислялась радиальная функция распределения атомов (RDF), средне-квадратичное смещение атомов (MSD) и коэффициент диффузии D по формуле Энштейна. Точка плавления определялась по изменению формы RDF и резкому увеличению коэффициента диффузии. Также строился график полной энергии всех атомов в расчетной области в зависимости от температуры. В точке плавления этот график претерпевал разрыв. В результате моделирования для сплава Ni_3Al была получена точка плавления в интервале 1700–1800 К, что немного превосходит экспериментальное значение 1673 К (1400 ^oC).

Для сравнения аналогичный расчет с помощью пакета DL_POLY (в нем реализован классический вариант MD) дал значение для точки плавления Ni_3Al T_m = 2300 K, что значительно выше экспериментального значения. Этот результат показывает основное преимущество первопринципных методов расчетов. В них не используются подгоночные параметры, и они, как правило, дают более точные результаты по сравнению с другими мето-

дами. В литературе встречались сообщения об использовании ab initio MD для моделирования технологических процессов напыления микросхем в фирме Intel.

Список литературы

1. Alder B.J., Wainwright T.E. Phase transition for a hard sphere system // J. Chem. Phys. 1957. – Vol. 27. – P. 1208–1209.

2. Рапапорт Д.К. Искусство молекулярной динамики // РХД. – Ижевск, 2012. – 630 с.

3. Sutton A.P., Chen J. Long-range Finnis-Sinclair potentials // Pyilos. Mag. Lett. – 1990. – Vol. 61. – No. 3. – P. 139–146.

4. Car R., Parrinello M. Unified Approach for Molecular Dynamics and Density-Functional Theory // Phys. Rev. Lett. – 1985. – Vol. 55. – P. 2471–2474.

5. Майер И. Избранные главы квантовой механики. – М.: Бином, 2006. – 384 с.

6. Kohn W., Sham L.J. Self-Consistent Equations Including Exchange and Correlation Effects // Phys. Rev. – 1965. – Vol. 140. – No. 4A. – A1133–A1138.

7. Gygi F., Galli G. Ab Initio Simulation in Extreme Conditions // Materials Today, 2005. – Vol. 8. – No. 11. – P. 26–32.

В.Я. Модорский, А.Ф. Шмаков

Пермский национальный исследовательский политехнический университет

ПАРАЛЛЕЛЬНЫЙ РАСЧЕТ ГАЗОДИНАМИЧЕСКОГО ПРОЦЕССА В КРУПНОГАБАРИТНОМ ВЕНТИЛЯТОРЕ

Рассмотрены этапы газодинамического расчета крупногабаритного вентилятора. Проведен газодинамический расчет нескольких десятков вариантов конструкций вентиляторов градирен. Разработана методика расчета

В энергетике для охлаждения рабочего тела широко применяются градирни. На данный момент в мире используется большое количество таких устройств различных типоразмеров. Одним из элементов градирни является вентилятор. При этом каждый вентилятор используется для определенных задач, обладает определенными размерами, производительностью и энергопотреблением [1–8].

Рассчитана аэродинамика нескольких десятков вариантов конструкции. В соответствии с установленными требованиями из рассмотренных вариантов выбрана конструкция нагнетателя.

Сформулирована следующая физическая модель:

– исследовалось течение во внутренней области конструкции;

– исследуемая конструкция вентилятора вписана во внешний объем, который формирует внешние границы потока;

– не учитывается прилипание частиц к стенке;

– стенки являются адиабатическими.

На рис. 1 представлена модель расчетной области, а на рис. 2 конечно-элементная модель.

Приняты следующие граничные условия:

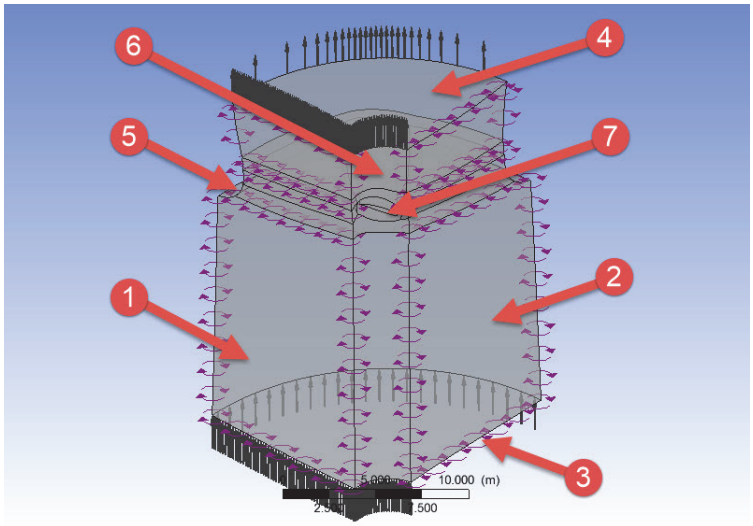


Рис. 1. Граничные условия: *1*, *2* – граничные условия периодичности расчетной области; *3* –граничные условия входа воздуха; *4* – граничные условия выхода воздуха; *5* – внутренняя стенка канала градирни; *6* – центральный канал градирни; *7* – внешняя стенка лопасти градирни

– в качестве рабочего тела используется идеальный газ;

– задано вращение расчетной области – 60 об/мин;

– на стенках расчетной зоны заданы условия периодичности для имитации расчета четырех лопастей;

– на входе *3* заданы условия «Входа». Давление на входе 101325 Па, температура газа 15 °С;

– на входе *4* заданы условия «Выхода». Давление на выходе 101325 Па;

– граница *5* используется для имитации внутренней стенки канала градирни. Заданы условия «Стенки». На стенке заданы условия вращения. Без прилипания;

– граница *6* используется для имитации центральной опорной площадки вентилятора для крепления лопастей;

– граница *7* используется для имитации наружной поверхности лопасти нагнетателя. Заданы условия «Стенки», без прилипания, гладкая стенка.

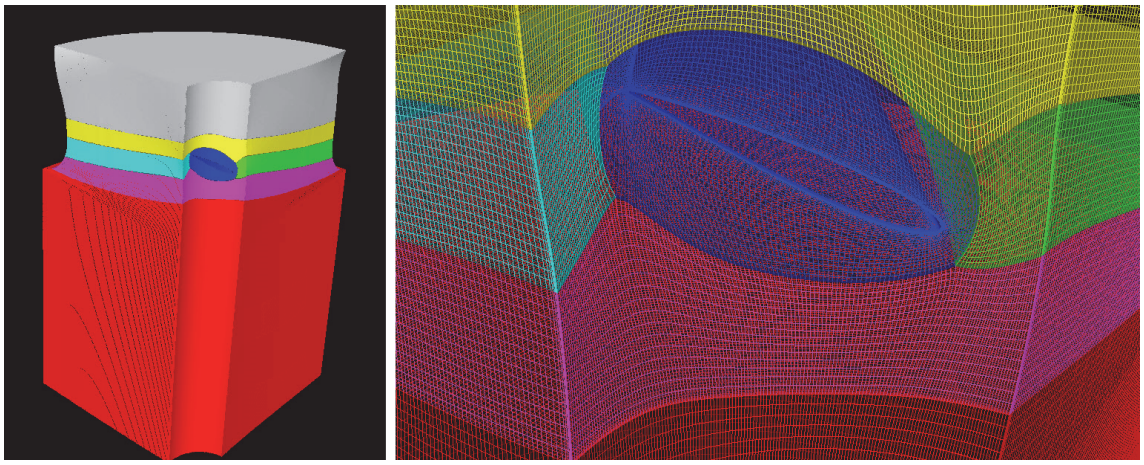


Рис. 2. Конечно-элементная модель

С учетом принятых допущений сформулирована следующая расчетная модель:

– конструкция полагается трехмерной (x, y, z);

– рабочее тело представляет собой несжимаемую жидкость;

– стенки конструкции не поглощают и не выделяют тепло:

В табл. 1 и 2 приводятся результаты вычислительных экспериментов в соответствии с номером расчетного варианта.

Таблица 1

Основные результаты вычислительного эксперимента по моделированию газодинамических процессов

Номер расчетного варианта	α_1, град. (угол атаки на конце лопасти)	α_2, град. (угол атаки на конце лопасти)	Ширина лопасти, мм	Расход млн $м^3/ч$	Мощность, кВт	Примечание
1	2	3	4	5	6	7
1	8	25	2250	17,2	586,7	Clark-Y, одно сечение по длине, с криволинейной закруткой
2	8	25	1900	10,9	226,6	Clark-Y, одно сечение по длине, с криволинейной закруткой
3	12	45	2100	11,7766	310,9	Clark-Y, одно сечение по длине, без закрутки уголок у комеля на 1/2 длины лопасти

Окончание табл. 1

1	2	3	4	5	6	7
4	16	25	1900	9,35754	457	Clark-Y, одно сечение по длине, без закрутки уголок у комеля на 1/2 длины лопасти
5	8	25	2250	11,2	257,1	Профиль: Clark-Y на обоих концах, одного сечения Равномерная крутка
6	8	8	2000	8,81	189	Профиль: Clark-Y на обоих концах, одного сечения
7	12	12	2100	10,2	291,99	Профиль: Clark-Y на обоих концах, одного сечения
8	10	25	2000	10,3	216,42	Профиль: ЦАГИ-6-13 % на обоих концах, одного сечения Равномерная крутка
9	10	25	2000	11,98	316,74	Профиль: Clark-Y на обоих концах, одного сечения Равномерная крутка

Таблица 2

Информационная карта результатов газодинамического расчета

Номер расчетного варианта	Примечания
1	2
1	Давление и скорости распределены неравномерно. Наблюдаются вихри и обратные токи воздуха в центральной части диффузора. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – не удовлетворяет требованиям*
2	Давление и скорости распределены неравномерно. Наблюдаются вихри и обратные токи воздуха в центральной части диффузора. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*
3	У задней части лопаток давление повышенное. Скорости распределены равномерно. В центральной части есть вихри и обратные токи. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*

Окончание табл. 2

1	2
4	Давление распределено равномерно. Вблизи стенок диффузора в плоскости лопастей наблюдается пониженное давление. Скорости распределены неравномерно. В центральной части градирни возникают вихри. *По расходу воздуха не удовлетворяет требованиям, по энергопотреблению – не удовлетворяет требованиям*
5	Давление распределено неравномерно. В плоскости лопастей вблизи стенок диффузора давление повышено. У передней части лопастей высокие скорости потока воздуха. В центральной части градирни возникают вихри. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*
6	Давление распределено неравномерно. Вблизи стенок диффузора в плоскости лопастей наблюдается пониженное давление. Скорости потока воздуха распределены неравномерно. *По расходу воздуха не удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*
7	Давление распределено неравномерно. Вблизи стенок диффузора в плоскости лопастей наблюдается пониженное давление. Скорости потока воздуха распределены неравномерно. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*
8	Давление распределено равномерно. Вблизи стенок диффузора в плоскости лопастей наблюдается пониженное давление. Скорости распределены неравномерно. В центральной части градирни возникают вихри. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*
9	Давление распределено равномерно. Вблизи стенок диффузора в плоскости лопастей наблюдается пониженное давление. Скорости распределены неравномерно. В центральной части градирни возникают вихри. *По расходу воздуха удовлетворяет требованиям, по энергопотреблению – удовлетворяет требованиям*

Примечание. Предпочтительными являются 8-й и 9-й варианты.

При измельчении сетки до значений более 32 млн ячеек результат практически не меняется. Использование грубой сетки (приблизительно 1,5 млн ячеек) снижает затраты машинного времени приблизительно в 3 раза по сравнению с оптимальной сеткой (32 млн ячеек), но дает занижение расхода примерно на 30 процентов и завышение энергопотребления в 1,5 раза. Поэтому грубая сетка не использовалась. Для сокращения времени счета использовалась 1/4 часть модели (сектор). Таким образом, для ускорения расчета использовалась модель с 8 млн ячеек.

Оценка влияния угла атаки на конце лопасти показала, что увеличение угла на 4 градуса приводит к увеличению потребляемой мощности двигателя на 54,5 % и расхода воздуха на 15,8 % (рис. 3).

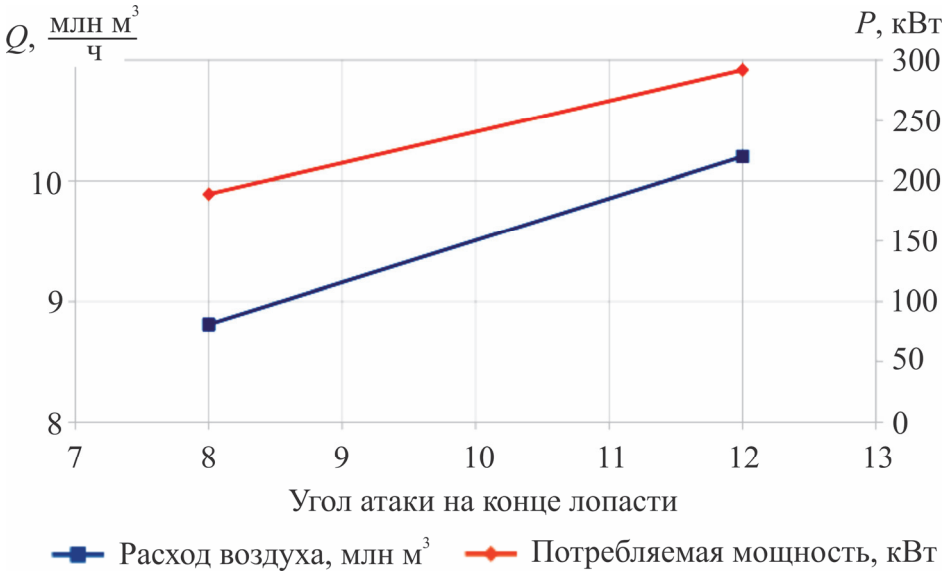


Рис. 3. Зависимость потребляемой мощности электродвигателя (P, кВт) и расхода воздуха (Q, млн м^3/ч) от угла атаки на конце лопасти (α_1)

Оценка влияния угла атаки вблизи комля лопасти показала, что увеличение угла на 17 градусов приводит к увеличению потребляемой мощности двигателя на 19,9 % и расхода воздуха на 6,2 %. Влияние α_2 на P и Q слабее, чем α_1 (рис. 4).

Рис. 4. Зависимость потребляемой мощности электродвигателя (P, кВт) и расхода воздуха (Q, млн м^3/ч) от угла атаки вблизи комля лопасти (α_2)

Оценка влияния ширины лопасти h показала, что увеличение ширины на 350 мм приводит к увеличению потребляемой мощности двигателя на 13,46 % и расхода воздуха на 2,8 % (рис. 5).

В расчетах использовались две модели газа. Первая модель – идеальный газ. Вторая модель – воздух при температуре 25 °С. Анализ влияния модели газа показал, что оно не превышает 1,5 %.

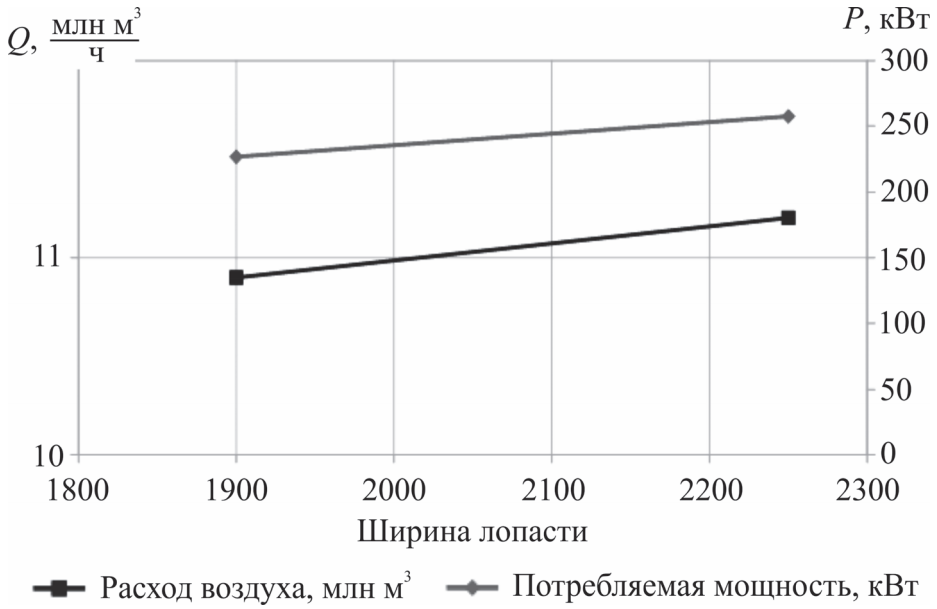


Рис. 5. Зависимость потребляемой мощности электродвигателя (P, кВт) и расхода воздуха (Q, млн м^3/ч) от ширины лопасти (h)

Наличие зазора между лопастью и корпусом градирни приводит к снижению потребляемой мощности электродвигателя и расхода воздуха менее, чем на 1 %.

Для ускорения расчетов был рассмотрен вариант конструкции градирни без конусной части конфузора. Однако это приводит к снижению потребляемой мощности электродвигателя и расхода воздуха на 20 % и не соответствует действительной конструкции. Поэтому для уменьшения ошибки расчетов во всех дальнейших расчетах конусная часть конфузора градирни учитывалась в обязательном порядке.

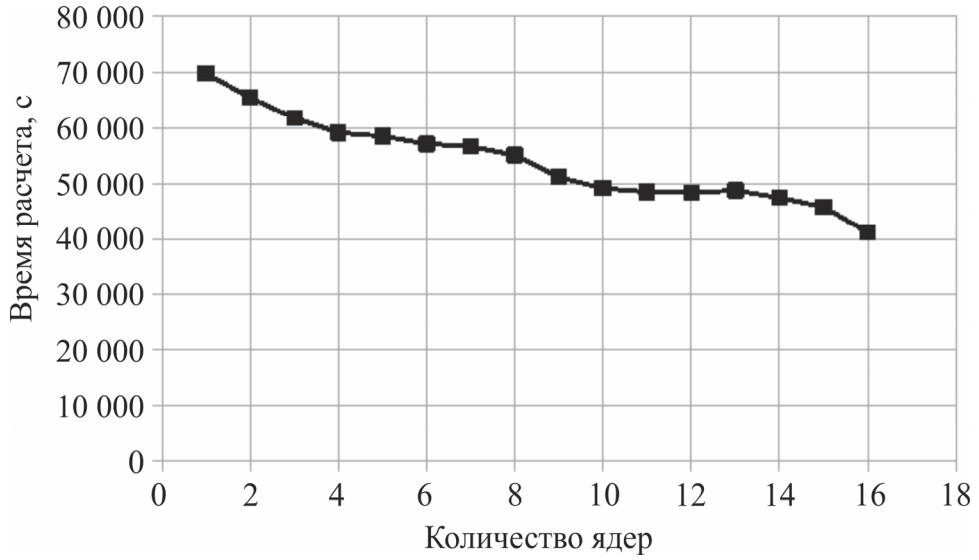


Рис. 6. Оценка скорости расчета в зависимости от количества ядер

Исключение из расчета застойной области вблизи оси вращения вентилятора путем введения вертикальной цилиндрической поверхности, ограничивающие расчетную область вблизи оси вращения, дает ошибку в пределах 2 %.

Также производилась оценка скорости расчета в зависимости от используемых ядер, результаты которого представлены на рис. 6. Оценка проводилась для количества ядер от 1 до 16. По графику видно, что с увеличением количества используемых ядер время расчета уменьшается.

Выводы

Аэродинамическая форма лопасти с профилем ЦАГИ-6 – 13 %, $\alpha_1 = 10°$, $\alpha_2 = 25°$, $h = 2000$ мм и равномерной круткой обеспечивает расход воздуха $Q = 10{,}2$ млн м^3/час и энергопотребление $P = 216{,}4$ кВт.

Для получения корректных результатов необходимо проводить учет влияния на движение газа конической части конфузора, так как это приводит к изменению результатов на 20–30 %.

Наибольшее влияние на расход газа и производительность оказывает изменение угла атаки на конце лопасти. Поэтому для изменения характеристик рекомендуем изменение этого параметра.

Требуются проверочные расчеты вентилятора на аэроупругость и обледенение

Для увеличения скорости расчета необходимо использовать 16 ядер (максимально допустимое для расчетного узла), количество ядер должно быть кратно 2.

Список литературы

1. Градирни промышленных и энергетических предприятий: справ. пособие / под общ. ред. В. С. Пономаренко. – М.: Энергоатомиздат: 1998. – 376 с.

2. URL: http://www.proficool.ru/

3. URL: http://gradirni.omul.org/

4. URL: http://www.xiron.ru/content/view/21342/28/

5. URL: http://www.rcdesign.ru/articles/avia/wings_profile

6. URL: http://avia-simply.ru/profil-krila/

7. Hensley C.J. Cooling Tower Fundamentals. Compiled from the knowledge and experience of the entire SPX Cooling Technologies staff. SPX Cooling Technologies. – Inc. Overland Park, Kansas USA, 2009.

8. Detlev G. Kröger. Air-cooled Heat Exchangers and Cooling Towers. – Tulsa, Oklahoma, PennWell Books, 2004.

А.А. Новиков, Р.Ф. Гибадуллин

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

ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ ПРОСТРАНСТВЕННЫХ ЗАПРОСОВ К ЗАЩИЩЕННОЙ КАРТОГРАФИЧЕСКОЙ БАЗЕ ДАННЫХ

Рассматривается создание модуля, исполняющего параллельные запросы к защищенной картографической базе данных (ЗКБД) в среде PostgreSQL, разработанного на языке C#. Приводятся тесты с измерением времени работы параллельного и последовательного модулей запросов с целью сравнения их эффективности по быстродействию.

Введение

В настоящее время во всем мире различными структурами и учреждениями широко используются средства пространственного анализа данных. В западных странах цифровые модели местности создаются в больших количествах и используются повсеместно. В нашей стране с вступлением в силу Федерального закона «О навигационной деятельности» от 14.02.2009, снимающего ограничения на точность определения координат объектов навигационной деятельности, общедоступные картографические данные в удобных для использования цифровых форматах только начинают появляться. С распространением навигационных устройств усилится процесс внедрения и распространения ГИС, рынок станет больше насыщаться цифровыми данными картографии.

Картографическая продукция имеет свои особенности. Она отличается высокой себестоимостью работ по ее получению: топографами и геодезистами затрачиваются немалые усилия на создание этой продукции. Кроме того, карты могут содержать конфиденциальные сведения, например новые места локализации нефтегазовых и урановых месторождений, цветных металлов. Поэтому задача их защиты актуальна.

Разработка модулей обработки запросов к защищенным картографическим базам данных может существенно повысить эффективность работы с такими базами данных, увеличить быстродействие при достаточном уровне защиты.

Архитектура сервера ЗКБД

Предложенная архитектура системы управления ЗКБД показана на рис. 1.Организация модулей в данной архитектуре является классикой [1], поэтому не нуждается в исчерпывающем описании. Рассмотрим отдельные модули.

Клиентская сторона.

Конвертер – это модуль, который преобразует результирующие отношения, получаемые от сервера СУБД, в формат ГИС-данных и проводит обратное преобразование для передачи ГИС-данных на сервер (и то, и другое выполняются опосредованно через клиентское хранилище).

Клиентский модуль выполняет два типа запросов к серверу ЗКБД: pull и push. Pull-запрос выполняет пространственную выборку данных, push-запрос выполняет изменение ЗКБД.

Серверная сторона.

Арбитр запросов – это модуль, который блокирует выполнение любых поступающих от клиента запросов в ходе выполнения pull-запроса.

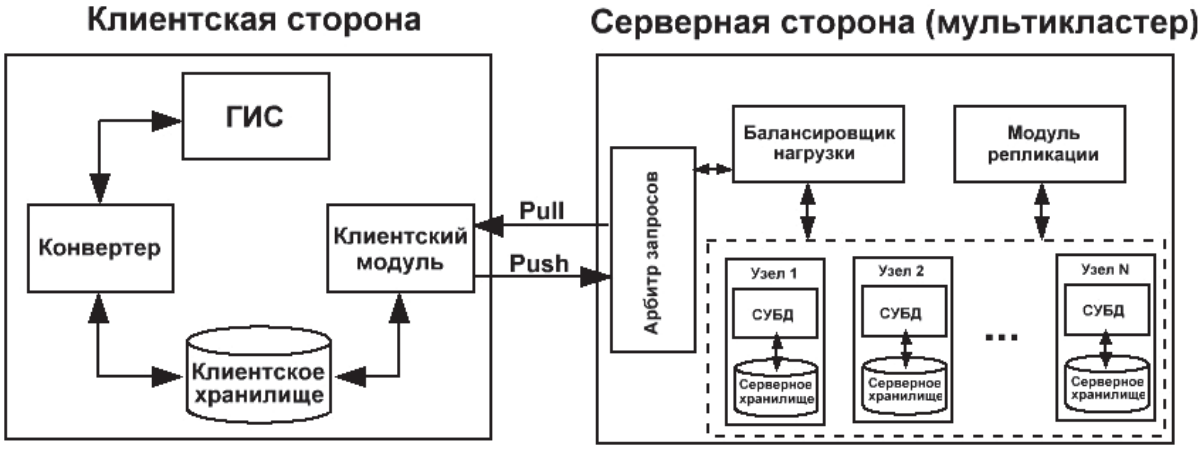


Рис. 1. Архитектура системы управления ЗКБД

При успешном обновлении ЗКБД выполняется процедура репликации баз данных на узлах, в противном случае выполняется откат базы данных до предыдущей версии.

Серверная сторона организована по принципу *мультикластера*. В случае такой организации каждый запрос, поступающий от клиентов, обрабатывается одним узлом сервера ЗКБД. За равномерное распределение запросов между узлами отвечает *балансировщик нагрузки*.

Библиотека TPL как инструмент распараллеливания запросов

Запросы от клиента формируются на языке стандарта SQL3, вводятся в специальное текстовое поле интерфейса веб-клиента и отправляются на сервер, где и происходит их параллельная обработка. Защищенная передача данных между клиентом и сервером реализована посредством SSL. Использование SSL решает задачу шифрования при передаче, а также затрудняет проведении атаки «человек посередине».

Параллельный модуль исполнения запросов реализован на языке C# с использованием библиотеки TPL. Одним из главных классов TPL является класс Parallel. Этот класс поддерживает набор методов, которые позволяют выполнять итерации по коллекции данных в параллельном режиме [2]. Применяя TPL, параллелизм в программу можно ввести двумя основными способами. Первый из них называется параллелизмом данных. При таком подходе одна операция над совокупностью данных разбивается на два параллельно выполняемых потока или больше, в каждом из которых обрабатывается часть данных. Так, если изменяется каждый элемент массива, то, применяя параллелизм данных, можно организовать параллельную обработку разных областей массива в двух или больше потоках. Нетрудно догадаться, что такие параллельно выполняющиеся действия могут привести к значительному ускорению обработки данных по сравнению с последовательным подходом.

Несмотря на то, что параллелизм данных был всегда возможен и с помощью класса Thread, построение масштабируемых решений средствами этого класса требовало немало усилий и времени. Это положение изменилось с появлением библиотеки TPL, с помощью которой масштабируемый параллелизм данных без особого труда вводится в программу. Второй способ ввода параллелизма называется параллелизмом задач. При таком подходе две операции или больше выполняются параллельно. Следовательно, параллелизм задач представляет собой разновидность параллелизма, который достигался в прошлом средствами класса Thread. А к преимуществам, которые сулит применение TPL, относится простота применения и возможность автоматически масштабировать исполнение кода на несколько процессоров. Библиотека TPL позволяет автоматически распределять нагрузку приложений между доступными процессорами в динамическом режиме, используя пул потоков CLR. Библиотека TPL занимается распределением работы, планированием потоков, управлением состоянием и прочими низкоуровневыми деталями. В результате появляется возможность максимизировать производительность приложений .NET., не имея дела со сложностями прямой работы с потоками. На рис. 2 показаны члены нового пространства имен .NET 4.0.

Для реализации параллельной обработки запроса в классе Parallel был выбран метод Invoke(). Данный метод позволяет выполнять один или несколько методов, указываемых в виде его аргументов. Он также масштабирует исполнение кода, используя доступные процессоры, если имеется такая возможность.

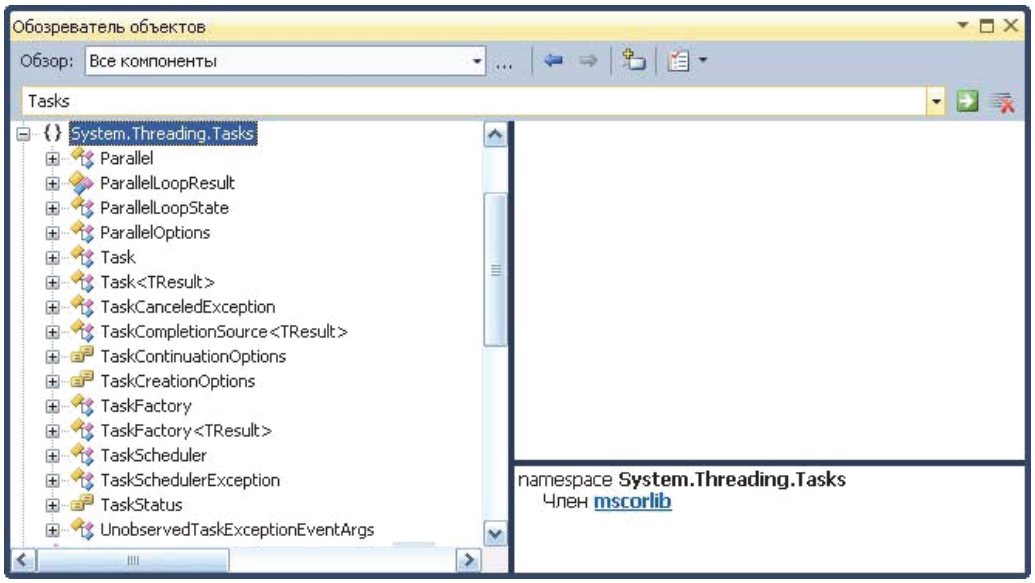


Рис. 2. Члены нового пространства имен .NET 4.0

Тестирование модуля запросов

Для измерения времени выполнения модулями запросов было проведено несколько различных по сложности тестов:

➢ простые селективные запросы (1):

- Select A.Name from public."Ozera, krupnyereki" A where A.waterway = 'riverbank' limit 20;
- Select A.Name from public."NaselennyePunkty" A where A.place = ‘city’ limit 20;

➢ запросы, усложненные функциями геометрической связи (2):

- Select R.Name, C.Name from public."Ozera, krupnyereki" R, public."NaselennyePunkty"C where ST_Crosses(R.geom,C.geom) = 't';
- Select C1.Name AS "Соседи" from public."NaselennyePunkty" C1, public."Naselennye Punkty_1" C2 where ST_Touches (C1.geom, C2.geom) = 't' and C2.Name = 'Москва';

➢ запросы с добавлением функции шифрования (3):

- Select pgp_sym_encrypt(cast(public."Gidroset"."geom" as text), cast('123' as text))from public."Gidroset" limit 20;
- Select pgp_sym_encrypt(cast(public."Sport"."geom" as text), cast('123' as text))from public."Sport" limit 20;

➢ добавление в запрос дешифрования (4):

- Select pgp_sym_decrypt(pgp_sym_encrypt(cast(public."Gidro-set"."geom" as text), cast('123' as text)), cast(‘123’ as text) from public."Gidroset" limit 20;
- Select pgp_sym_decrypt(pgp_sym_encrypt(cast(public."Sport"."geom" as text), cast('123' as text)), cast(‘123’ as text) from public."Sport" limit 20;

➢ самый сложный вариант с добавлением функций шифрования/дешифрования к функциям геометрической связи (5):

- Select C.name,ST_Centroid(pgp_sym_decrypt(pgp_sym_encrypt(cast (C.geom as text),cast('123' as text)), cast('123' as text))) from public."NaselennyePunkty" C;
- Select A.Name,ST_Length(pgp_sym_decrypt(pgp_sym_encrypt(cast (A.geom as text), cast('12345' as text)), cast('12345' as text))) as "Длина" from public."Zheleznyedorogi" A where A.Name<> ' '.

Для тестирования использовалась картографическая база данных Российской Федерации. Тестирование проводилось на одном узле с двумя ядрами на одном физическом процессоре. По каждой группе сложности было проведено по 20 тестов (по 10 тестов на запрос). В таблице приведены средние временные значения.

Выполнение запросов на параллельном и последовательном модулях

Группа сложности	Время обработки запроса при использовании последовательного модуля, с	Время обработки запроса при использовании параллельного модуля, с
1	1,39	0,61
2	1,12	0,34
3	1,57	0,28
4	0,23	0,22
5	2,23	0,59

Как видно из результирующей таблицы, использование параллельного модуля запросов дает нам прирост в производительности в два раза на самых простых селективных запросах и почти в шесть раз на сложных запросах. Используемая конфигурация предположительно не будет являться порогом масштабируемости для рассматриваемой системы. Следовательно, с увеличением количества процессоров и числа ядер на них предполагается дальнейшее повышение производительности.

Список литературы

1. Вершинин И.С., Гибадуллин Р.Ф., Пыстогов С.В. Использование кластерных технологий при решении задач защиты картографических данных // Высокопроизводительные параллельные вычисления на кластерных системах (HPC–2009): материалы IX Междунар. конф.; Владимир. гос. ун-т. – Владимир, 2009. – С. 68–72.

2. TaskParallelism (TaskParallelLibrary). – URL: http://msdn.microsoft.com/ru-ru/library/dd537609(v=vs.110).aspx (Последнее обращение: 04.10.2014 г.)

М. Новожилов, А. Белокаменская

Нижегородский государственный университет имени Н.И. Лобачевского

РАСПАРАЛЛЕЛИВАНИЕ МЕТОДА МАРШИРУЮЩИХ КУБОВ ДЛЯ ИНТЕРАКТИВНОЙ ТРЕХМЕРНОЙ ЦЕФАЛОМЕТРИИ

Разработан программный комплекс трехмерных инструментов стоматолога-гнатолога для диагностики заболеваний височно-нижнечелюстного сустава (ВНЧС). Инструментальная поддержка построена на использовании трехмерных исходных данных, которые предоставляет томография. Для трехмерной визуализации используются методы объемного рендеринга (DVR). Для измерений используется геометрическая реконструкция полигональной модели изоповерхностей методом марширующих кубов. Метод марширующих кубов распараллелен с помощью OpenMP, что позволяет реконструировать поверхность височно-нижнечелюстного сустава за время, близкое к секунде. На основе указанных методов реконструкции реализованы инструменты, обеспечивающие трехмерные методики измерений в практике стоматолога-гнатолога.

Введение

Патология височно-нижнечелюстного сустава (ВНЧС) затрагивает значительную часть населения, хотя и не рассматривается как проблема здравоохранения. Между тем 3–7 % населения пытается вылечить боль и дисфункции сустава или связанных с ним структур. Согласно литературным данным, распространенность клинических симптомов у населения разнится от 6 до 93 %, вероятно, в результате использования различных клинических критериев. Исследования последних двух десятилетий [6,7] констатируют примерно постоянный процент людей, наблюдающих симптомы дисфункции ВНЧС, что говорит о том, что пока не существует массовых эффективных методов лечения этого заболевания.

Важнейшим инструментом диагностики и оценки результатов ортодонтического лечения является цефалометрия. Цефалометрический анализ – это оценка размеров и пространственного взаимоотношения зубов, челюстей и лицевой части черепа в целом. Например, в 2002 г. 90 % ортодонтов в США регулярно применяли цефалометрический анализ [5]. Долгое время цефалометрический анализ проводили, используя боковые и фронтальные рентгенограммы. С распространением компьютерной томографии (КТ) и конусно-лучевой компьютерной томографии (КЛКТ или CBCT) стало возможным проводить цефалометрический анализ, используя трехмерные данные.

Для проведения цефалометрических измерений необходимо реконструировать поверхности черепа и мягких тканей. Для уменьшения времени

проведения цефалометрического анализа желательно приблизить скорость реконструирования к реальному времени.

Реализация методов анализа КЛКТ-томограмм

Для трехмерной визуализации компьютерной томограммы использовалась техника прямого объёмного рендеринга (Direct Volume Rendering, DVR) (рис. 1). Методом прямого объемного рендеринга вычисляется интеграл объёмной визуализации [1]:

$$I = \int_0^D \text{intensity}\left(\mathbf{x}(t)\right) \times \text{extinction}\left(\mathbf{x}(t)\right) \times \exp\left(-\int_0^t \text{extinction}\left(\mathbf{x}(t')\right) \mathrm{d}t'\right) \mathrm{d}t.$$

Луч $\mathbf{x}(t)$ параметризован расстоянием t от объектива виртуальной камеры, а излучаемая яркость intensity($\mathbf{x}$) и коэффициент затухания extinction($\mathbf{x}$) могут быть вычислены в любой точке $\mathbf{x}$ пространства с использованием передаточной функции. Интеграл объемной визуализации описывает процесс накопления цвета вдоль луча с учетом затухания по пути следования.

Через D обозначено расстояние, на которое луч проникает в объем (в большинстве случаев определяется временем выхода из ограничивающей оболочки).

Для устранения артефактов постклассификации, которые заключаются в искажениях цвета, хорошо видимых глазом, в виде колец или полос, реализован метод предынтегрированного объёмного рендеринга.

Рис 1. Визуализация методом DVR

Для последующей имплементации трехмерной цефалометрии реализована возможность построить трёхмерные объекты: точку, линию, плоскость и отобразить их вместе с объёмными данными.

Для геометрической реконструкции челюстно-лицевой системы используется метод марширующих кубов (рис. 2). Этот метод обладает важным свойством с точки зрения современной вычислительной техники – возможностью вполне равномерно распределить вычисления на несколько вычислительных устройств (ядер центрального процессора или GPU).

Впервые метод марширующих кубов был описан Лоренсеном и Клайном в 1987 г. [4]. Суть метода: в пространстве рассматривается трёхмерная прямоугольная сетка, линии которой параллельны осям координат. В узлах сетки известны значения поля. Для реконструкции изоповерхности достаточно по отдельности рассмотреть ячейки данной сетки – «кубики» с восемью вершинами. В алгоритме предполагается, что каждое ребро ячейки пересекает только одну изоповерхность и значение поля в вершине куба может быть больше или меньше изозначения (точка пересечения ребра и изоповерхности интерполируется исходя из значений поля в соответствующих соседних вершинах куба). Поэтому получаем 256 различных комбинаций пересечения ячейки с изоповерхностью. Первоначально эти 256 различных случаев сводили к 15 базовым триангуляциям, на чём основана реализация алгоритма [2]. Но при данном подходе после реконструкции изоповерхности часто возникали несоответствия. В 1995 г. Е.В. Черняев увеличил число базовых комбинаций до 33 [3].

Независимо от реализации алгоритма остается его основной принцип: ячейки обрабатываются независимо друг от друга. В задаче, где необходимо получить поверхность, заданную списком треугольников со своими координатами вершин, ячейки могут распределяться по вычислительным устройствам произвольно.

В задачах компьютерной графики часто требуется, чтобы вместо координат вершин треугольника использовались индексы – номера соответствующих вершин из некоторого списка. Такая структура применительно к поверхностям значительно сокращает объём данных и позволяет в один проход по списку треугольников определить соседей для каждой вершины. В данном случае при распределении кубов по вычислительным устройствам необходимо учесть, чтобы поиск для новой вычисленной вершины, совпадающей с ранее добавленной, производился минимальным перебором.

Например, можно рассмотреть перебор кубов во вложенных циклах по x, y, z. В данном случае область поиска вершины можно ограничить окрестностью по координате x, равной соответствующему приращению. Для распределения вычислений ось x можно поделить на количество вычислительных устройств и передать им соответствующие наборы кубов, а после реконструкции объединить получившиеся поверхности в одну.

Данный подход был реализован с помощью библиотеки OpenMP и протестирован на процессоре core i7 3820 3.6 ГГц. Было получено ожидаемое ускорение, близкое к линейному:

Поверхность	Время реконструкции (с)			Ускорение	
	1 ядро	2 ядра	4 ядра	2 ядра	4 ядра
91147 примитивов, 443232 кубов	1,419	0,736	0,41	1,92	3,46
296396 примитивов, 1481040 кубов	4,038	2,08	1,143	1,94	3,53

Рис. 2. Геометрическая реконструкция ВНЧС

Для удобства работы и ускорения измерений можно выделить зону интереса в томограмме, которую необходимо исследовать и построить полигональную модель только по выделенному массиву данных. Полигональная модель позволяет оперировать отдельными элементами челюстно-лицевой системы. Есть возможность проводить измерения. Например, можно измерить кратчайшее расстояние в суставе (рис. 3).

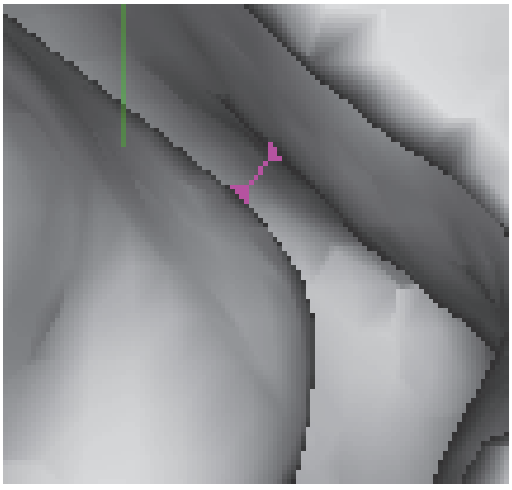


Рис. 3. Измерение зазора суставных поверхностей

Программа написана в среде MS Visual Studio на языках C++ , OpenCL и GLSL, с использованием библиотеки Qt.

Заключение

Разработан программный комплекс трехмерных инструментов стоматолога-гнатолога для диагностики заболеваний височно-нижнечелюстного сустава (ВНЧС). Инструментальная поддержка построена на использовании трехмерных исходных данных, которые предоставляет томография. Для трехмерной визуализации используются методы объемного рендеринга (DVR). Для измерений используется геометрическая реконструкция полигональной модели изоповерхностей методом марширующих кубов. Метод марширующих кубов распараллелен с помощью OpenMP, что позволяет реконструировать поверхность височно-челюстного сустава за время, близкое к секунде. На основе указанных методов реконструкции реализованы инструменты, обеспечивающие трехмерные методики измерений в практике стоматолога-гнатолога: диалоговый режим 3D-редактирования и дополнения цефалометрических ориентиров и измерение расстояний между элементами челюстно-лицевой системы. Для томограмм поддерживается входной формат DICOM.

Список литературы

1. Полноэкранное сглаживание в реализации предынтегрированного рендеринга для визуализации трехмерных скалярных полей на GPU / Д.К. Боголепов, И. Бугаев, А.А. Белокаменская, В.Е. Турлапов // Научная визуализация. – 2012. – Кв. 4. – Т. 4, № 4. – С. 2–16.

2. Bourke P. Polygonising a scalar field. – URL: http://paulbourke.net/geometry/polygonise/, свободный (дата обращения 8.10.2014).

3. Chernyaev E.V. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. Technical Report CERN CN 95-17, CERN, 1995

4. Lorensen William E., Cline Harvey E. Marching Cubes: A high resolution 3D surface construction algorithm // Computer Graphics. – 1987. – Vol. 21. – July.

5. The influence of cephalometrics on orthodontic treatment planning / P.G. Nijkamp [at al.] // The European Journal of Orthodontics. 2008. – No. 3.11. – 6 p.

6. Рабухина Н.А., Сёмкин В.А. Дисфункция височно-нижнечелюстных суставов (клиника, диагностика и лечение); ЗАО «Редакция журнала “Новое в стоматологии”». – М., 2000. – 56 с.

7. Review of temporomandibular joint pathology / Roda Rafael Poveda [at al.] // Part I: Classification, epidemiology and risk factors. Med Oral Patol Oral Cir Bucal. – 2007. – P. E292–298.

С. А. Носова, В. И. Турлапов

Нижегородский государственный университет им. Н. И. Лобачевского

СЕГМЕНТАЦИЯ И 3D-РЕКОНСТРУКЦИЯ МИТОХОНДРИЙ МОЗГА ПО СНИМКАМ С ЭЛЕКТРОННОГО МИКРОСКОПА

Статья посвящена новому подходу к параллельной алгоритмизации реконструкции клеток мозга по микрофотографиям с электронного микроскопа, основанному на использовании митохондрий как центров реконструкции, а также описанию алгоритма 3D-реконструкции митохондрий. Проанализирован характер изображений митохондрий, предложен алгоритм их сегментации, использующий, в свою очередь, ряд подходов и методов: лучевой подход, метод разрастания регионов, методы математической морфологии, оценку инвариантных признаков объекта. Предложенный алгоритм уверенно детектирует области митохондрий на серии микрофотографий. 3D-реконструкция и визуализация митохондрий осуществлена с использованием пакета ParaView

Введение

Задача исследования и моделирования поведения мозга на микроуровне является весьма актуальной. Одним из подходов к моделированию клеток мозга является их геометрическая реконструкция по микрофотографиям, полученным электронным микроскопом (ЭМ). Для геометрической реконструкции клетки мозга и ее органелл специалисту необходимо детектировать их визуально и вручную разметить их границы на каждой микрофотографии [1, 2]. До сих пор автоматически реконструировать удавалось только клетки, которые были предварительно покрашены непосредственно в препарате. Ручная реконструкция занимает большое количество времени специалиста, требует высокой квалификации в чтении микрофотографий и, самое главное, резко ограничивает интенсивность исследований клеток мозга. Инструментарий для автоматической 2D-, 3D-сегментации и последующей 3D-геометрической реконструкции данных ЭМ отсутствует. А задача повышения степени автоматизации с накоплением данных электронной микроскопии мозга становится все более актуальной.

Микрофотографии ЭМ представляют собой зашумленные изображения, достаточно сложные для детектирования отдельной клетки и ее органелл: границы клеток слабоконтрастны и часто неразличимы на основе только одной фотографии (приходится привлекать соседние слои, неявно включая механизмы 3D-детектирования). В то же время отдельные органеллы, такие как митохондрии, более контрастны и могли бы детектироваться автома-

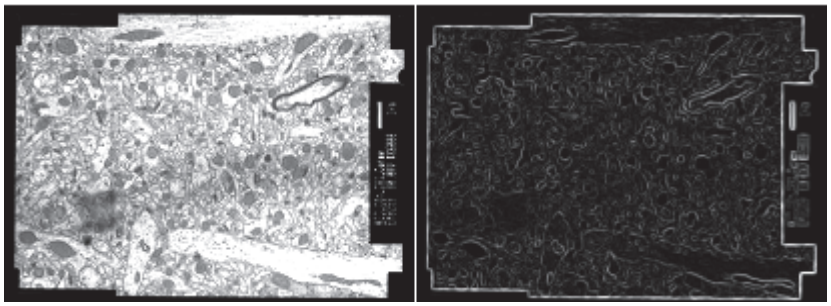
тически. Предлагается использовать митохондрии в качестве центров в параллельном алгоритме геометрической 3D-реконструкции клетки. Первым этапом такой реконструкции должна быть реконструкция митохондрий.

Анализ исходных данных

Исходные данные предоставлены Научно-нейрональным центром Института живых систем ННГУ им. Н.И. Лобачевского и представляют собой набор микрофотографий мозга крысы: 73 изображения в формате bmp размером 4756×6916. Исходные изображения имеют высокое разрешение (менее 10 нм/пиксел), которое является избыточным в задаче сегментации митохондрий. Экспериментальным путем установлено, что сжатие изображения в 8 раз по каждой размерности не снижает точность сегментации, но примерно в 50 раз снижает время сегментации.

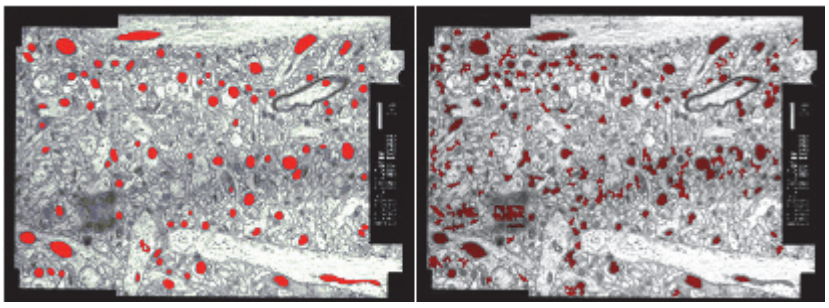
Для статистического исследования изображений митохондрий была произведена ручная сегментация (рис. 2, *а*) на нескольких снимках. Для удаления шума использовался простейший фильтр усреднения. Для дальнейшего анализа строится карта градиентов отфильтрованного изображения (рис. 1, *б*).

Для начала на одном слое были выделены все различимые митохондрии.



а *б*

Рис. 1. Карта градиентов отфильтрованного изображения:
а – слой исходных данных; *б* – карта градиентов



а *б*

Рис. 2. Изображение митохондрий: *а* – ручная сегментация митохондрий;
б – результат автоматической сегментации

1. Митохондриям соответствуют области с интенсивностью в диапазоне [30,170].

2. На обработанном специальным фильтром (см. рис. 1, *б*) снимке области митохондрий представлены интенсивностями в диапазоне [0,15].

3. Длина главных осей описывающего митохондрию эллипса колеблется от 2 до 140.

4. Каждое сечение митохондрий близко по форме к одному из нижеперечисленных объектов с приведенными характеристиками выпуклости области (solidity), заполненности области (extent) и эксцентриситета (eccentricity):

а) круг (0.9781, 0.7822, 0);

б) эллипс (0.9665, 0.6602, 0.8769);

в) прямоугольник 1:12 (0.8816, 0.2071, 0.9971);

г) S-образный объект (0.4595, 0.4208, 0.8728).

5. Размер области (количество пикселей, принадлежащих области) на уменьшенном снимке находится в диапазоне [30,700].

Данные параметры верны для конкретных тестовых данных, однако общий принцип остается следующим: митохондрии сегментируются по интенсивности их изображения, форме и размеру.

Сегментация

Алгоритм сегментации митохондрий состоит из следующих основных шагов:

1. Предобработка данных.

2. Первичная сегментация: тестирование каждой точки изображения на возможность принадлежности к митохондрии.

3. Анализ полученных областей на соответствие митохондрии.

4. Постобработка.

Этап предобработки данных заключается в фильтрации обыкновенным box-фильтром ($r = 3$) и построении карты градиентов для него (см. рис. 1, *б*).

Первичная сегментация состоит в тестировании принадлежности к митохондрии каждой точки отдельного слоя. Для этой цели используются лучевой метод [3]. Перечислим основные шаги первичной сегментации.

1. Для каждого пикселя изображения:

а) проверить, попадает ли интенсивность пикселя в область допустимых значений (п. 1 и 2 из подразд. «Анализ исходных данных»). Если да, то

б) сгенерировать набор лучей, с началом в месте расположения данного пикселя (используем генерацию 4 лучей).

2. Для каждого луча:

а) двигаться вдоль луча до тех пор, пока не будет найдено пересечение с пикселем заданной интенсивности T = 40. Сохранить координаты пикселя, в котором это пересечение произошло;

б) координаты пересечения с двумя противоположно направленными лучами образуют «диагональ». Рассчитать ее длину;

в) если длина каждой «диагонали» находится в допустимом диапазоне (п. 5 подразд. «Анализ исходных данных»), то пометить пиксель как принадлежащий объекту.

После проведения приведенных выше шагов получаем изображение, на котором отмечены некоторые пиксели как принадлежащие объекту. На этом этапе можно отметить следующее:

1. Существует множество одиночных пикселей, помеченных как принадлежащих митохондриям (своего рода точечный шум на бинарном изображении объект–фон, где объект – пиксели, помеченные как митохондрии, а фон – все остальное).

2. Внутри митохондрий помечены только те пиксели, которые расположены на некотором расстоянии *r* от границы митохондрий. Кроме того, эти группы пикселей образуют области с пустотами внутри.

Для удаления подобных артефактов используются методы математической морфологии (в пределах 4-связности) [4]. Для исправления первого артефакта применяется морфологическое «открытие», а для исправления второго – морфологическое «расширение».

Далее анализируются полученные объекты. При помощи алгоритма разрастания регионов [5] выделяются отдельные связные области, и если они не соответствуют необходимым условиям (п. 4, 5 подразд. «Анализ исходных данных»), то удаляются из области рассмотрения.

После этого мы переходим к рассмотрению трехмерного набора данных. В трехмерном случае сегментируем связные участки и отсекаем те из них, размер которых меньше наперед заданного значения. Это сделано из следующих соображений:

1) убрать на каждом слое из рассмотрения те регионы, продолжения которых нет ни на одном из соседних слоев (ошибки сегментации);

2) оставить только те митохондрии, которые представляют наибольший интерес для использования (в том числе в качестве центров реконструкции).

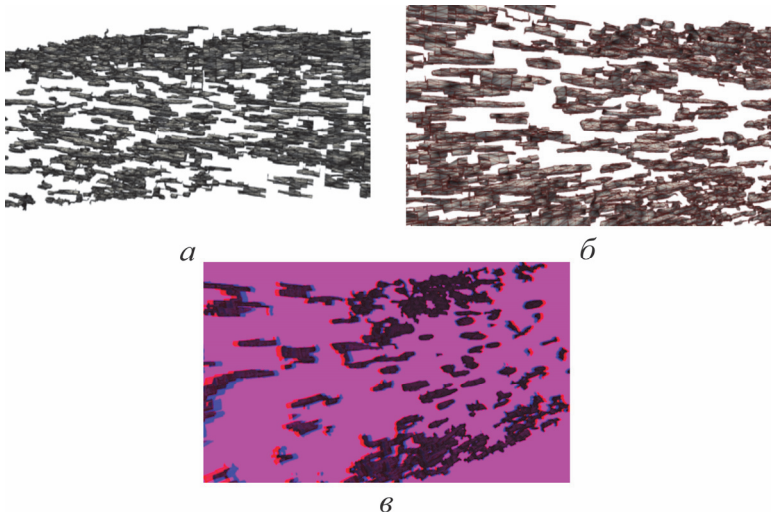


Рис. 3. Реконструированные митохондрии: *а* – общий вид массива тел митохондрий; *б* – в окне измененного масштаба, демонстрирующего сетку на поверхности митохондрий; *в* – анаглифное стереоизображение фрагмента

После этого, если необходимо, можно также применить методы матморфологии для расширения или замыкания областей.

3D-реконструкция

Для визуализации и реконструкции данных использована открытая система параллельной научной визуализации ParaView [6]. ParaView предоставляет пользователю возможности интерактивной визуализации и исследования больших массивов данных для качественного и количественного анализа, используя в качестве основы библиотеку VTK.

Для визуализации данных достаточно сохранить их в формате VTK. В данном случае используется запись с использованием структурированной сетки. Результаты реконструкции можно увидеть на рис. 3.

Выводы

В статье предложен новый подход к параллельной алгоритмизации реконструкции клеток мозга по микрофотографиям с ЭМ, основанный на использовании митохондрий как центров реконструкции. Подробно рассмотрен алгоритм 3D-реконструкции митохондрий как основа и начальный этап подхода. Проанализирован характер изображений митохондрий, предложен алгоритм их сегментации, использующий, в свою очередь, ряд подходов и методов: лучевой подход, метод разрастания регионов, методы математической морфологии, оценку инвариантных признаков объекта. В эксперименте показано, что предложенный алгоритм уверенно детектирует области митохонд-

рий на серии микрофотографий в реальном времени. 3D-реконструкция и визуализация митохондрий выполняется также в реальном времени силами открытой системы параллельной научной визуализации ParaView.

Работа поддержана грантом (соглашение от 27 августа 2013 г. № 02.B.49.21.0003 между МО РФ и ННГУ).

Список литературы

1. Open Connectome Project. – Collectively reverse-engineeringthe brainone synapseat time. – URL: http://www.openconnectomeproject.org/#!links/cgc4

2. Synapse Web hosts a suite of tools for analysis and reconstructionof three-dimensional objects from serial sections. – URL: http://synapses.clm.utexas.edu/tools/index.stm

3. Gavrilov N.I., Turlapov V.E. Novel approach to development of direct volume rendering algorithms based on visualization quality assessment // Programming and computer software. – 2014. – Vol. 40. – Iss. 4. – P. 174–184.

4. Patra J., Moulick H.N., Manna A.K. Biomedical image processing with morphology and segmentation methods for medical image analysis. – American Journal of Engineering Research (AJER). – 2013. – Vol. 02. – Iss. 07. – P. 227–244.

5. Najarian K., Splinter R. Biomedical Signal and Image Processing. – CRC Press, 2006. – 448 p.

6. Официальный сайт открытой параллельной системы научной визуализации ParaView. – URL: http://www.paraview.org.

Л.Ф. Нурисламова, И.М. Губайдуллин

Институт нефтехимии и катализа РАН, Уфа

РЕДУКЦИЯ МАТЕМАТИЧЕСКОЙ МОДЕЛИ РЕАКЦИИ ОКИСЛЕНИЯ ВОДОРОДА

Предложена методика упрощения математической модели химической реакции, основанная на анализе чувствительности целевой функции к изменению параметров модели. Представлены результаты сравнительного анализа моделирования реакции окисления водорода по детализированной и редуцированной схемам.

Введение

Компьютерное моделирование играет решающую роль в понимании сложных химических реакций. При этом качество моделирования напря-

мую зависит от выбора схемы химических превращений реакции. Использование детальных кинетических схем (включают десятки веществ и сотни стадий) при моделировании процесса важно для полного и точного описания процессов в широком диапазоне температур и условий. Однако использование детальных кинетических схем требует значительных вычислительных затрат при моделировании, в то время как кинетику реакции можно описать, используя гораздо меньшее число стадий, которые важны для рассматриваемой задачи при заданных реальных условиях. Кроме того, задача редуцирования кинетических схем механизма реакции является необходимым этапом на пути проведения многомерных расчетов реакторов. По этим причинам упрощение математической модели (уменьшение числа уравнений и вместе с этим числа параметров, определяющих поведение системы) за счет сокращения числа стадий и веществ схемы реакции является центральной проблемой в исследовании сложных реакций.

Первые работы, направленные на сокращение схемы механизма реакции, появились еще в начале XX столетия [1, 2]. В последние годы получил распространение метод анализа чувствительности численных моделей, что связано с развитием вычислительной техники и компьютерного моделирования [3, 4]. В химической кинетике при проведении анализа чувствительности математической модели реакции исследуют чувствительность следующих выходных параметров: расчетные значения температуры, концентрации всех либо одного из веществ (наиболее важного для исследователя), скорости стадий, скорость тепловыделения и пр. [5]. В статье предлагается проводить анализ чувствительности целевой функции к изменению констант скоростей стадий, где целевая функция характеризует меру близости расчетных значений по имеющейся и измененной схемам реакции. В качестве метода анализа чувствительности применяется глобальный метод И.М. Соболя [6]. Объект исследования данной работы – модельная реакция окисления водорода. Целью работы является получение редуцированной схемы данной реакции, которая описывает такое же поведение основных веществ реакции, как и детальная, но имеет значительно меньшее число стадий реакций.

Математическая модель реакции

Моделируемая система представляет собой замкнутую, идеально теплоизолированную емкость постоянного объема, в начальном состоянии заполненную смесью газообразных веществ, способных вступать в систему суммарно-экзотермических реакций. Математическая модель данного процесса включает в себя дифференциальные уравнения для описания поведения температуры смеси и концентраций реагентов [7]

$$\frac{dT}{dt} = -\frac{\sum_{i=1}^{M} \frac{dc_i}{dt} H_i(T)}{\sum_{i=1}^{M} c_i \cdot C_{p,i}(T)},$$

$$\frac{dc_i}{dt} = \sum_{j=1}^{N} S_{ij} w_j,\ w_j = k_j \prod_{i=1}^{M} (c_i)^{|\alpha_{ij}|} - k_{-j} \prod_{i=1}^{M} (c_i)^{\beta_{ij}} \qquad 1)$$

с начальными условиями: $c_i(0) = c_i$, $T(0) = T$, где T – температура; t – время; $H_i(T)$ – энтальпия; $C_{p,i}(T)$ – теплоемкость; c_i – концентрации веществ, участвующих в реакции; M – количество веществ; N – количество стадий; S_{ij} – стехиометрическая матрица; w_j – скорость j-й стадии; k_j, k_{-j} –константы скорости прямой и обратной реакции соответственно; α_{ij} – отрицательные элементы S_{ij}; β_{ij} – положительные элементы S_{ij}.

В расчетах рассматривались разбавленные стехиометрические смеси H_2:O_2 = 1:2, 2:1, 1:1, варьируемый диапазон начальной температуры – 1000–1500 К, нормальное атмосферное давление. Для определения индукционного периода вычислялся момент достижения максимума скорости тепловыделения $\max \partial T / \partial t$.

Схема реакции окисления водорода

В литературе представлено большое количество детальных схем механизма реакции окисления водорода. Среди различных детальных схем наиболее часто встречается схема, представленная в таблице [8], которая показала хорошее соответствие с экспериментальными данными по температуре окисления потока, времени индукции и др. Данная схема и была положена в основу расчетов.

Детальная схема реакции окисления водорода и ее параметры

№ п/п	Стадия	k_i			k_{-i}		
		k_0	n	E	k_0	n	E
1	2	3	4	5	6	7	8
1,2	$H+O_2 \leftrightarrow O+OH$	1,92E+14	0	1,64E+04	5,48E+11	0,39	–2,93E+02
3,4	$O+H_2 \leftrightarrow H+OH$	5,08E+04	2,67	6,29E+03	2,67E+04	2,65	4,88E+03
5,6	$OH+H_2 \leftrightarrow H+H_2O$	2,16E+08	1,51	3,43E+03	2,30E+09	1,4	1,83E+04
7,8	$O+H_2O \leftrightarrow 2OH$	2,97E+06	2,02	1,34E+04	1,47E+05	2,11	–2,90E+03
9,10	$H_2+M \leftrightarrow 2H+M$	4,58E+19	–1,4	1,04E+05	1,15E+20	–1,68	8,20E+02
11,12	$O_2+M \leftrightarrow 2O+M$	4,52E+17	–0,64	1,19E+05	6,17E+15	–0,5	0,00E+00
13,14	$OH+M \leftrightarrow O+H+M$	9,88E+17	–0,74	1,02E+05	4,71E+18	–1	0,00E+00
15,16	$H_2O+M \leftrightarrow H+OH+M$	1,91E+23	–1,83	1,19E+05	4,50E+22	–2	0,00E+00
17,18	$H+O_2+M \leftrightarrow HO_2+M$	1,48E+12	0,6	0,00E+00	3,09E+12	0,53	4,89E+04

Окончание таблицы

1	2	3	4	5	6	7	8
19,20	$HO_2+H \leftrightarrow H_2+O_2$	1,66E+13	0	8,23E+02	3,16E+12	0,35	5,55E+04
21,22	$HO_2+H \leftrightarrow 2OH$	7,08E+13	0	2,95E+02	2,03E+10	0,72	3,68E+04
23,24	$HO_2+O \leftrightarrow OH+O_2$	3,25E+13	0	0,00E+00	3,25E+12	0,33	5,33E+04
25,26	$HO_2+OH \leftrightarrow H_2O+O_2$	2,89E+13	0	-4,97E+02	5,86E+13	0,24	6,91E+04
27,28	$H_2O_2+O_2 \leftrightarrow 2HO_2$	4,63E+16	–0,35	5,07E+04	4,20E+14	0	1,20E+04
29,30	$H_2O_2+M \leftrightarrow 2OH+M$	2,95E+14	0	4,84E+04	3,66E+08	1,14	–2,58E+03
31,32	$H_2O_2+H \leftrightarrow H_2O+OH$	2,41E+13	0	3,97E+03	1,27E+08	1,31	7,14E+04
33,34	$H_2O_2+H \leftrightarrow H_2+HO_2$	6,03E+13	0	7,95E+03	1,04E+11	0,7	2,40E+03
35,36	$H_2O_2+O \leftrightarrow OH+HO_2$	9,55E+06	2	3,97E+03	8,66E+03	2,68	1,86E+04
37,38	$H_2O_2+OH \leftrightarrow H_2O+HO_2$	1,00E+12	0	0,00E+00	1,84E+10	0,59	3,09E+04

Примечание. $[k_0]$ = см^3/(моль·с), $[k]$ = см^3/(моль·с), $[E]$ = кал/моль.

Процедура проведения расчетов

Предлагается анализировать чувствительность целевой функции к изменению констант скоростей стадий, где целевая функция характеризует меру близости расчетных значений по имеющейся и измененной схемам реакции в различные моменты времени:

$$F_{obj} = \sum_{i=1}^{M}\sum_{j=1}^{N}\left(c_{ij}^{sh1} - c_{ij}^{sh2}\right)^2, \qquad (2)$$

где c_{ij}^{sh1} – расчетные значения концентраций, полученные по имеющейся схеме (таблица); c_{ij}^{sh2} – расчетные значения концентраций, полученные изменением (возмущением) параметров в схеме; N – количество точек времени; M – количество веществ.

Анализ чувствительности целевой функции позволяет выявить те параметры, которые обеспечивают наибольший (наименьший) вклад в погрешность при моделировании реакции. Математическая модель может содержать параметры, изменение значений которых не влияет на качественное и количественное описание процесса. Вклад таких параметров в значение целевой функции не будет иметь большого значения. Поэтому стадии, которые не служат для моделирования кинетических кривых веществ, можно исключить из рассмотрения.

Для вычисления глобальных коэффициентов чувствительности был выбран метод, предложенный И.М. Соболем, так как исследуемые системы содержат нелинейные зависимости. Предположим, что исследуемая математическая модель описывается функцией $y = f(x_1,...,x_n)$. Функция $f(\bar{x})$ определена и интегрируема в единичном n-мерном кубе $K^n = \{0 \le x_1 \le 1,...,0 \le x_n \le 1\}$. Глобальными показателями чувствительности

называются отношения дисперсий $S_{i_1,\ldots,i_s} = D_{i_1,\ldots,i_s} / D$. Величины $D_{i_1,\ldots,i_s} = \int_K f^2_{i_1,\ldots,i_s} dx_{i_1} \ldots dx_{i_s}$ называются дисперсиями, а $D = \int_K f^2(x)dx - f_0^2$ – полной дисперсией.

Интегралы для определения дисперсий вычислялись методом Монте-Карло. Генерировались различные значения констант скоростей (2^{14} различных наборов). Константа каждой стадии варьировалась в пределах 5% от своего значения, вычисленная по данным, приведенным в таблице. Предполагалось, что константы скорости являются независимыми друг от друга, поскольку отсутствовала информация о распределении входных параметров, значения параметров для вычисления коэффициентов чувствительности отбирались из равномерного распределения. Для равномерного распределения возможных значений констант скоростей использовались точки ЛПτ-последовательности, предложенные И.М. Соболем [9]. По сравнению с другими методами глобального анализа чувствительности метод Соболя имеет ряд преимуществ. Все дисперсии могут быть вычислены с помощью обычного метода Монте-Карло, не требующего явного нахождения членов разложения функции в отличие от методов FAST и E-FAST. Также не требуется информация о статистическом распределении каждого входного фактора модели. Следует признать, что данный метод является вычислительно затратным, требует большого числа запусков модели, в нашем случае каждый раз решается система уравнений (1). Однако метод Монте-Карло легко и эффективно распараллеливается, поскольку константы скорости являются независимыми друг от друга и решение задачи (1) с различным набором констант не требует синхронизации. Все вычисления производились в Matlab версии R2010a. Для распараллеливания вычислений между процессорами использовался цикл parfor. Задача запускалась на компьютере со следующими характеристиками: AMD Phenom II X4 940, nVidia GeForce gts 450.

Результаты вычислительных экспериментов

При вычислении коэффициентов чувствительности целевой функции к изменению параметров модели учитывалось изменение абсолютной разности значений концентраций для следующих веществ: H_2, O_2, H_2O, H, O, OH. В качестве c_{ij}^{sh1} были взяты значения, полученные численным решением системы (1) в 6 моментах времени со следующими с начальными условиями: концентрации H_2– 2.93e-06 моль/см^3, O_2 – 1.4e-06 моль/см^3, давление P = 1 атм, температура – 1000 К.

На рис. 1 приведены только те стадии, для которых значения коэффициента чувствительности составляли не менее 0,001. Анализ коэффициентов чувствительности показывает, что возможно также исключение 7, 8,

34-й стадий (см. рис. 1). Отметим небольшое число стадий, к которым действительно чувствительна исследуемая система. Элементарная реакция $H+O_2 \rightarrow OH+O$ является ярко выраженной лимитирующей стадией, поскольку это стадия развития цепи.

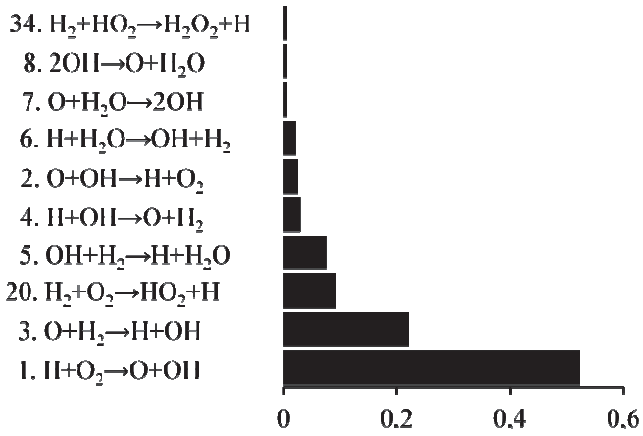


Рис. 1. Полные глобальные коэффициенты чувствительности целевой функции F_{obj} к изменению констант скоростей стадий

Редуцированная схема (содержит стадии 1–6, 20) представляет собой минимальную схему реакции окисления водорода, показавшую хорошие результаты для температуры 1000 К. Как видно из рис. 2, упрощение схемы механизма реакции до 7-й стадий не изменило общую динамику изменения концентраций целевых веществ во времени.

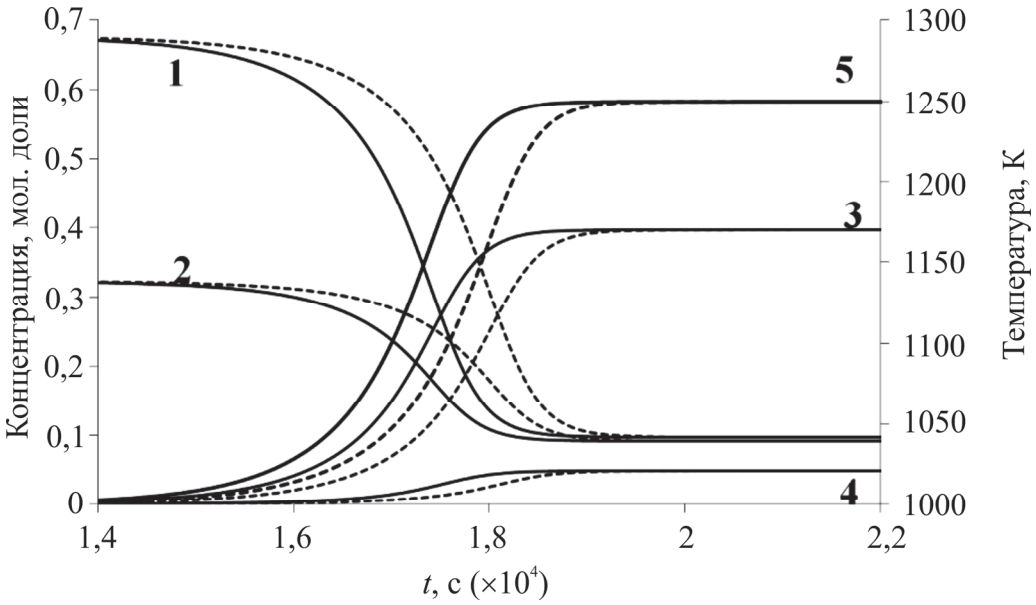


Рис. 2. Изменения концентраций основных веществ и температуры во времени в процессе окисления смеси водород-кислород (2:1), полученных с использованием детальной схемы (сплошная линия) и редуцированной схемы (пунктирная линия); начальная температура 1000 К: *1* – H_2, *2* – O_2, *3* – H_2O, *4* – O, *5* – температура

Для проверки точности описания процесса редуцированной схемой были проведены расчеты зависимости времени индукции горения от начальной температуры при различных стехиометрических соотношениях смеси водород-кислород соотношением 1:2, 1:1, 2:1. Совпадение по времени индукции для сокращенной схемы с детальной практически идеальное (рис. 3).

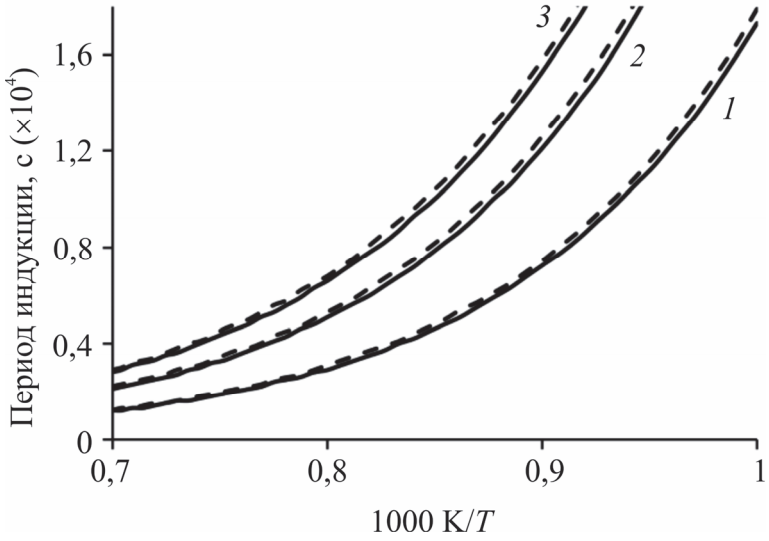


Рис. 3. Зависимость периода индукции от температуры при различных стехиометрических соотношениях смеси водород–кислород: *1* – 2:1, *2* – 1:1, *3* – 1:2

Заключение

Разработана методика, позволяющая определять сокращенные схемы химических превращений, обеспечивающие предсказание концентраций основных выходов реакции с требуемой точностью и отличающихся небольшими требованиями к вычислительным ресурсам компьютера. Эффективность применения методики продемонстрирована сравнительными результатами моделирования по детализированным и сокращенной схемам реакции окисления водорода.

Работа выполнена при поддержке РФФИ (грант № 12-07-00324).

Список литературы

1. Pepiot P., Pitsch H. Systematic Reduction of Large Chemical Mechanisms. 4th Joint Meeting of the U.S. Sections of the Combustion Institute. – Philadelphia, PA, 2005. – P. 4–16.

2. Brown N.J., Guoping Li. Mechanism Reduction Via Principal Component Analysis // Int J Chem Kinet. – 1997. – Vol. 29. – P. 393–414.

3. Sensitivity Analysis for Chemical Models / A. Saltelli, M. Ratto, S. Tarantola, F. Campolongo // Chem Rev. – 2005. – Vol. 105 (7). – P. 2811–2828.

4. Turányi T. Sensitivity analysis of complex kinetic systems. Tools and applications // Jounal of Mathematical Chemistry. – 1990. – Vol. 5 (3). – P. 203–248.

5. An explicit reduced mechanism for H2-air combustion / P. Boivin, C. Jiménez, A.L. Sánchez, F.A. Williams // Proceedings of the Combustion Institute. – 2011. – Vol. 33 (1). – P. 517–523.

6. Соболь И.М. Об оценке чувствительности нелинейных математических моделей // Математическое моделирование. – 1990. – Т. 2. – Вып. 1. – С. 112–118.

7. Димитров В.И. Простая кинетика. – Новосибирск: Наука, 1982. – 383 с.

8. A Comprehensive Modeling Study of Hydrogen Oxidation / M. O'Connaire, H.J. Curran, J.M. Simmie, W.J. Pitz, C.K. Westbrook // Int. J. Chem. Kinet. – 2004. – Vol. 36. – P. 603–622.

9. Соболь И.М. Равномерно распределенные последовательности с дополнительным свойством равномерности // Журнал вычислительной математики и математической физики. – 1976. – Т. 16, № 5. – P. 1332–1337.

С.А. Орлов

Национальный исследовательский Томский государственный университет

СУПЕРКОМПЬЮТЕРНЫЙ ЦЕНТР КАК КОНКУРЕНТНОЕ ПРЕИМУЩЕСТВО НАЦИОНАЛЬНОГО ИССЛЕДОВАТЕЛЬСКОГО УНИВЕРСИТЕТА

Работа посвящена вопросам функционирования суперкомпьютерного центра в Национальном исследовательском университете. Освещаются принципы взаимодействия с научными подразделениями, лабораториями, факультетами с целью эффективного использования ресурсов общего доступа. Рассматриваются вопросы построения системы постоянного внебюджетного центра для ежегодной модернизации ресурсов и наращивания вычислительных мощностей в условиях постоянного стремительного устаревания вычислительной техники.

В конце 2013 г. Министерством образования и науки РФ были названы 15 вузов России, которые стали победителями конкурсного отбора на право получения специальной субсидии, направленной на продвижение вузов в международных рейтингах. В соответствии с указом президента, к 2020 г. не менее пяти российских университетов должны войти в сотню лучших в мире, не менее 15 – в TOP-200. В число отобранных вузов вошел

Томский государственный университет, старейший университет Сибири. Стратегической целью ТГУ является формирование научно-образовательного, инновационного, культурного центра, входящего в число ведущих университетов мира.

В качестве целевых показателей, необходимых для достижения поставленной стратегической цели, руководством университета были выдвинуты следующие: количество научных статей, публикуемых в ведущих мировых изданиях из списков Scopus и Webof Science, на каждого научно-педагогического работника, количество цитирований опубликованных статей, доля зарубежных профессоров, преподавателей и исследователей, работающих в университете, доля доходов от внебюджетных источников в структуре доходов университета и др. Очевидно, что для выполнения всех этих показателей необходимо формирование эффективной модели управления существующими научными активами университета, а также их развитие и модернизация. Одним из таких активов являются высокопроизводительные вычислительные ресурсы, объединенные в суперкомпьютер. В настоящий момент в большинстве областей науки и техники моделирование является неотъемлемой частью научного процесса. В Томском государственный университете с богатой научной школой в области моделирования физических, химических, биологических процессов, применения математического аппарата к задачам физики, химии, важным инструментом для выполнения исследований является суперкомпьютер, использование которого позволяет готовить качественные и высокоцитируемые статьи для публикации в рейтинговых журналах. Для привлечения квалифицированных зарубежных научных работников, профессоров необходима хорошая материальная база, которая может быть использована ими для научных исследований в университете. В то же время одним из показателей является доля внебюджетных доходов университета от различных видов деятельности и их инвестирование в развитие университета. Все эти факторы привели к тому, что была разработана новая для нашего университета концепция управления высокопроизводительными вычислительными ресурсами.

Для оптимизации финансирования суперкомпьютерного центра была предложена новая политика предоставления доступа к вычислительным мощностям для всех пользователей. Основным ресурсом суперкомпьютера, представляющим интерес для пользователей, является процессор, выполняющий расчеты, поэтому единицей измерения использования вычислительных мощностей является *процессоро-час*. В отношении всех пользователей ведется постоянный учет количества использованных процессоро-часов, и, соответственно, может быть проведен учет финансовых затрат на каждый расчет исходя из установленного для данного пользователя тарифа.

Расчет стоимости работы одного шестиядерного процессора IntelXeon 5670 в течение одного часа с учетом текущих затрат показал, что номинальный тариф составляет 0,89 руб/процессоро-час. Однако, помимо текущих расходов, в цену должна закладываться инвестиционная составляющая, которая позволит в дальнейшем обновлять программное и аппаратное обеспечение. Для оценки инвестиционной составляющей, которую необходимо закладывать в цену с целью сохранения текущих позиций с точки зрения решения актуальных физико-математических задач, используется эмпирический закон Мура, который гласит, что каждые 18 месяцев происходит удвоение вычислительных мощностей при одних и тех же геометрических параметрах систем. Требования актуальных задач с каждым годом растут пропорционально данному закону. Это значит, что для сохранения конкурентоспособности вычислительной системы необходимо каждые полтора года удваивать мощность суперкомпьютера. Иначе неизбежно устаревание, а значит, снижение возможностей научных групп для решения актуальных задач. Исходя из того, что текущая вычислительная мощность 62 Тфлопс, а стоимость железа производительностью 1 Тфлопс на мировом рынке сегодня составляет около 300 тыс. руб., для удвоения мощности необходимо заложить инвестиционную составляющую в размере 18,6 млн руб. в год, а с учетом обновления только системного программного обеспечения сумма составит 20 млн руб. в год. Таким образом, тариф с инвестиционной составляющей составит 4,07 руб/процессоро-час.

При этом выделяются 4 группы пользователей, в отношении которых вводятся разные виды финансовой политики.

1. Сотрудники университета, проводящие научные исследования и имеющие финансовую поддержку своих проектов от различных фондов в виде грантов или договоров (РФФИ, РНФ, ФЦП, Госзадание, др.), в которых основной организацией-исполнителем является ТГУ. Поскольку по каждому проекту, в котором основной организацией-исполнителем является ТГУ, в бюджет университета перечисляются накладные расходы, то для *первой* группы пользователей в качестве источника финансирования расчетов предлагается использовать те *накладные расходы*, которые уже перечислены в бюджет университета.

2. Сотрудники лабораторий мирового уровня университета, проводящие научные исследования в рамках выполнения Программы повышения конкурентоспособности ТГУ и имеющие финансовую поддержку в рамках программы. Финансирование расчетов второй группы пользователей проводится из средств тех лабораторий, которые созданы в рамках выполнения Программы повышения конкурентоспособности ТГУ. Оплата

производится путем перераспределения из лаборатории в суперкомпьютерный центр и может быть использована только для оплаты текущих расходов Центра.

3. Лица (физические или юридические), проводящие научные исследования по проектам, в которых основной организацией-исполнителем является другая организация (не ТГУ). Для финансирования расчетов *третьей* группы пользователей предлагается заключить долгосрочные договоры с основными партнерами ТГУ, использующими ресурсы суперкомпьютера. С другими организациями заключаются отдельные договоры на проведение отдельных расчетов. Для долгосрочных договоров счета выставляются ежемесячно и оплачиваются согласно договору по установленному в договоре тарифу. Для прочих организаций счета выставляются по факту выполнения работ.

4. Студенты и аспиранты ТГУ, проводящие исследования в рамках подготовки магистерских и кандидатских диссертаций. Для *четвертой* группы пользователей предлагается установить нулевой тариф для использования ресурсов, при этом ввести ограничение на использование самых мощных процессоров, пользующихся спросом у предыдущих групп.

При этом отдаются в безлимитное использование те вычислительные процессоры, которые уже не приносят большого прироста для сложных научных расчетов. Они могут быть использованы при подготовке магистерских и кандидатских диссертаций, а также для отладки больших приложений.

Для всех групп пользователей предлагается обязательное упоминание в публикуемых статьях того, что расчеты проведены с использованием суперкомпьютера СКИФ Cyberia на базе Томского государственного университета. Все эти группы описаны в тех приоритетах, которые установлены для пользования вычислительными ресурсами.

Кроме финансовой политики была разработана модернизированная система доступа к суперкомпьютерным ресурсам. Большинство исследователей, использующих программное обеспечение для моделирования либо разрабатывающих его самостоятельно, привыкло работать исключительно на локальных рабочих станциях, управляемых ОС Windows. При этом наибольшую эффективность прикладное ПО, используемое на суперкомпьютере СКИФ Cyberia, показывает под управлением ОС Linux (CentOS 5.5). Возникла проблема обучения всех исследователей навыкам работы с Linux, а также передачи данных, компилирования, использования прикладного программного ПО. К тому же при использовании новой финансовой политики возникли вопросы о методиках подсчета расходуемых процессоро-часов. В большинстве случаев проблема решалась путем консультаций пользователей с системными администраторами, однако была предложена новая концепция доступа.

На базе сайта Межрегионального супервычислительного центра был создан личный кабинет пользователя, в который вошли такие функции, как загрузка файлов непосредственно с компьютера пользователя в папку на СХД суперкомпьютера, постановка задачи в очередь по нажатию кнопки в личном кабинете, реализованная на Flashконсоль в личном кабинете для тех, кто умеет и привык работать с putty под Windows, реализованный на Flashграфический интерфейс для большинства программных пакетов. Была внедрена система биллинга, в результате использования которой каждый пользователь может видеть, какое количество процессоро-часов и на какую задачу было израсходовано. Использование концепции личного кабинета избавило пользователей от необходимости устанавливать дополнительное ПО на свои компьютеры, тратить время на освоение командной строки ОС Linux, а вместо этого использовать удобный интерфейс и выполнять свои исследовательские проекты с большей эффективностью.

Поскольку в наличии находятся не только относительно новые процессоры, но и старые IntelXeon5150, которые уже не пользуются спросом у научных групп, но при этом могут быть использованы под другие задачи, такие как расчеты трехмерной графики, обработка данных для коммерческих организаций, в 2014 г. были заключены договоры на проведение работ для нескольких коммерческих организаций по таким расчетам. В результате ТГУ получил дополнительные средства для дальнейшего оборудования, что является эффективным использованием имеющихся ресурсов.

Внедрение новой политики доступа к ресурсам привело к тому, что финансирование, которое может быть направлено на постоянную ежегодную модернизацию, поступает от выполнения научных проектов конечных пользователей, а также коммерческих заказчиков. При этом возникает постоянное стимулирование пользователей к привлечению новых грантов, а также переводу своих грантов из других организаций под аффилиацию ТГУ. Таким образом, объем выполняемых проектов научными группами, использующими суперкомпьютер за первое полугодие 2014 г., увеличился вдвое. Количество публикаций, подготовленных с использованием суперкомпьютера, увеличилось в 3 раза по сравнению с аналогичным периодом прошлого года. При этом университет получил возможность модернизировать суперкомпьютерные ресурсы из внебюджетных источников. Таким образом, система управления суперкомпьютерным центром как инструментом для получения качественных научных результатов дала возможность переориентироваться на достижение ключевых показателей Национального исследовательского университета.

Д.А. Павлов, Р.Ш. Минязев

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

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ДЛЯ ПЛАТФОРМЫ С ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

Предлагается архитектура GPU СУБД и алгоритмы исполнения SQL-запросов с использованием графического ускорителя как сопроцессора в рамках подхода GPGPU на основе технологии CUDA. Рассматривается случай, когда база данных полностью размещается в памяти GPU.

Введение

Использование графических ускорителей обладает большим потенциалом повышения производительности путем уменьшения времени обработки запросов в параллельных СУБД. Реализация этого потенциала требует развития теории таких систем для выработки практических рекомендаций к их построению. Это возможно путем детального анализа протекающих в них процессов. Только знание особенностей динамики параллельных СУБД с утяжеленными GPU-акселераторами узлами позволит дать объективные рекомендации по их построению. Для проведения подобных исследований была разработана GPU СУБД. Программные модули системы, названной Clusterix-G, написаны на языке C++ с включением функций CUDA в модулях, отвечающих за взаимодействие с GPU. СУБД является колоночно ориентированной, только такой подход позволяет эффективно манипулировать данными при копировании их и последующей обработке на GPU. Архитектура СУБД показана на рисунке.

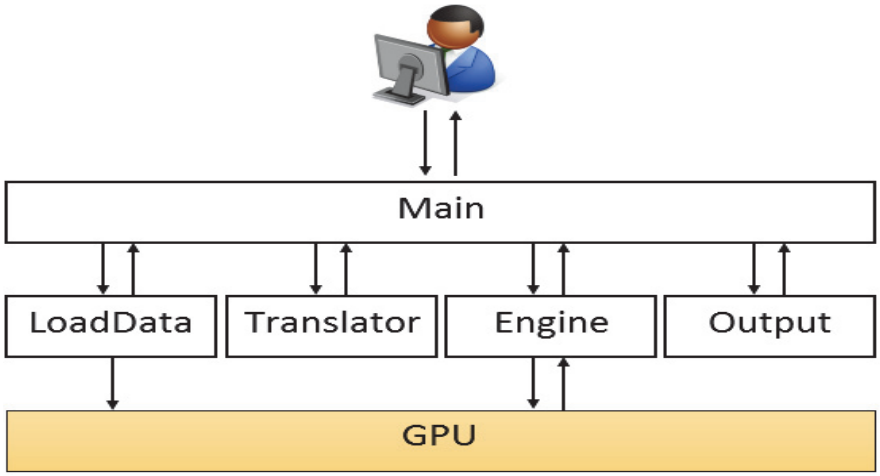


Рис. Архитектура системы Clusterix-G

Основные модули системы:

Main – управляющий модуль, с его запуска начинается работа системы. Реализует командный интерфейс СУБД, отвечает за хранение БД в оперативной памяти и по соответствующим командам вызывает остальные модули.

LoadData – отвечает за загрузку таблиц БД из файлов сначала в оперативную память, а затем в память GPU. Возможны варианты реализации парсера в этом модуле для чтения файлов различных форматов (например, XML).

Translator – отвечает за ввод SQL-запроса, проверяет его на корректность (синтаксическая правильность, соответствие типов данных, существование указанных в запросе таблиц и полей) и преобразует строку запроса во внутреннее представление программы.

Engine – «ядро» программы, он отвечает непосредственно за отбор записей по условиям запроса и использование вычислительных мощностей графического процессора. Результатом его работы является массив с индексами отобранных записей.

Output – преобразует индексы из массива, созданного модулем Engine, в номера записей в конкретных таблицах и выводит результат запроса на экран либо в указанный пользователем файл.

Поскольку графический ускоритель – узкоспециализированное вычислительное устройство, создание модуля Engine потребовало учета как специфики архитектуры GPU и технологии CUDA, так и условий, диктуемых самой предметной областью – работой с реляционной базой данных. К модулю предъявлялись следующие требования: сведение к минимуму передачи данных между памятью GPU и хоста; минимизация требуемого для работы модуля объема памяти GPU; максимальное использование параллелизма при вычислениях; максимальное упрощение алгоритма каждого отдельного потока; соблюдение принципов объединения запросов к памяти (coalescing) [1]; учет характеристик конкретного вычислительного узла и графического ускорителя. Последовательное выполнение этих требований при разработке привело к следующим ключевым решениям: идентичность таблиц БД, хранящихся в памяти GPU и оперативной памяти хоста, использование маркера для отбора записей по условиям запроса, многопоточность, пофрагментная обработка данных, максимальное уменьшение объема выгружаемых с GPU данных, применение оригинального параллельного алгоритма вычисления кумулятивной суммы [2].

Вычислительный эксперимент был проведен с использованием фрагмента базы данных теста TPC-H и пакета специально созданных запросов. Его цель – сравнение быстродействия созданной системы, исполь-

зующей GPU, и обычной СУБД MySQL. При тестировании использовался ПК следующей конфигурации: 4-ядерный процессор AMD Phenom II 3,3 ГГц; 4 ГБ оперативной памяти DDR3; графический ускоритель GeForce 210 (16 ядер CUDA) с 512 МБ памяти DDR2. Результаты экспериментов представлены в таблице.

Исходя из полученных результатов можно сделать следующий вывод: GPU-СУБД показывает наибольший прирост быстродействия по сравнению с обычной СУБД при обработке средних и «тяжелых» запросов без использования дополнительной информации (например, первичных ключей). Их наличие заметно ускоряет обработку в системе MySQL, тогда как в Clusterix-G время исполнения остается неизменным. Для наиболее «легких» запросов быстродействие примерно одинаково у обеих систем. Достигнуто ускорение исполнения запроса до 32 раз по сравнению с традиционной СУБД. Также система Clusterix-G показала хорошую стабильность и экономичность по памяти.

Результаты эксперимента

Номер запроса	Объем БД, МБ	Кол-во кортежей	Время, Clusterix-G, мс	Время, MySQL, мс	Ускорение, раз
1	0,002	25	1,06	1	0,94
2	0,002	125	1,03	2	1,94
3	1,266	250 000	11,71	20	1,71
4	1,266	250 000	5,78	121	20,93
5	1,266	250 000	7,76	251	32,35
6	1,266	250 000	6,66	23	3,45
7	1,267	1 250 000	28,41	122	4,29
8	20,685	5 000 000	69,28	794	11,46
9	20,685	5 000 000	75,28	1951	25,92
10	21,947	2 млрд	29455,1	Ошибка: недостаточно памяти	

Список литературы

1. CUDA C Programming Guide. – URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#device-memory-accesses (Последнее обращение: 14.09.2014 г.)

2. Prefix sum – Wikipedia. – URL: http://en.wiki-pedia.org/wiki/Prefix_sum (Последнее обращение: 14.09.2014 г.)

С.В. Панков

Южный федеральный университет, г. Ростов-на-Дону

ФОРМАЛЬНАЯ СПЕЦИФИКАЦИЯ ОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Для класса асинхронных однородных вычислительных систем (ОВС) предлагается метод построения логико-математической спецификации. Исследуется вопрос применения этой спецификации для обоснования функциональной корректности ОВС. Демонстрируется независимость вида спецификации ОВС с регулярной топологией связей (а также её анализа) от размера системы.

Введение

Рассматривается класс асинхронных ОВС, использующих произвольное (сколь угодно большое) число однотипных вычислительных устройств с регулярной топологией связей между ними. При этом ОВС являются либо специализированными системами с фиксированной структурой, либо универсальными с программируемой структурой, но уже настроенными на решение определённой задачи. Проблема обоснования функциональной корректности (правильности работы с точки зрения вычисляемой функции) для таких систем является актуальной.

Предлагается метод спецификации ОВС на основе *L*-программ, логико-математической модели асинхронных параллельных вычислений, описанной впервые в [1]. Строящаяся спецификация ориентирована на применение логических методов верификации *L*-программ [2] для анализа корректности ОВС. (*L*-программы, также достаточно подробно описывались в [3], где подобный подход применялся к системам параллельных вычислений над общей памятью.) Благодаря однотипности вычислительных процессов, регулярности структуры связей в ОВС модель не зависит от числа процессов, что обеспечивает также и независимость анализа спецификации от их числа.

Описание ОВС

ОВС определяется в виде, ориентированном на моделирование *L*-программой, как система $S=<P,C>$, где P – множество процессов, а C – множество линий связи. Множество P разбивается на три подмножества $P=P\mathrm{in} \cup P\mathrm{calc} \cup P\mathrm{out}$. Здесь $P\mathrm{in}$ – множество процессов *ввода*, имеющих по одному выходу (называемому *входом системы*) и ни одного входа. $P\mathrm{calc}$ – это множество однотипных *вычислительных* процессов, имеющих конечное не-

нулевое число входов – *n*, выходов – *m*, и вычисляющих набор функций <*f1*,...,*fm*>. Входные данные процесса снимаются со входов, вычисляются функции на этом наборе данных, а затем результаты помещаются на соответствующие выходы <*1*,…,*m*>. (Предполагается, что все функции определены на входном наборе данных.) *P*out – это множество процессов *вывода*, имеющих по одному входу (называемому *выходом системы*) и ни одного выхода. Выходы процессов из *P*calc соединены с входами соседних процессов этого же множества линиями связи так, что образуется регулярная структура. (Имеется в виду структура графа, вершины которого соответствуют процессам из *P*calc, а рёбра – линиям связи между ними). На *C* также накладываются следующие ограничения: 1) каждый выход процесса из *P* связан точно с одним входом некоторого процесса; 2) каждый вход процесса из *P* связан только с одним выходом некоторого процесса. ОВС также характеризуется потоками данных на её входах и выходах.

Процесс *p* из *P*calc может находиться в одном из двух состояний: в состоянии *ожидания* (в нём все выходы *p* свободны) или в состоянии *выполнения*. Процесс *p* может перейти из состояния ожидания в состояние выполнения, если справедливо следующее условие *готовности к выполнению*: все входы *p* содержат данные. Само выполнение считается одномоментным и сопровождается снятием данных со всех входов, заполнением результатами вычисления всех выходов *p*. Передача данного с выхода процесса *p* на вход приемника *p*' одномоментна и может произойти, если выполняется следующее условие *готовности к передач*е: выход процесса *p* заполнен, вход процесса *p*' свободен, а сам процесс *p*' (если *p'* не совпадает с *p*) находится в состоянии ожидания.

Состояние системы *S* определяется состоянием потоков данных на её входах и выходах, наличием данных на входах и выходах процессов, а также самими данными. В начальном состоянии все входы системы содержат первые по порядку данные из входных потоков, а выходы системы не содержат данных. Все процессы-вычислители находятся в состоянии ожидания.

Шаг работы системы *S* состоит в следующем: 1) происходит заполнение данными произвольного множества входов системы с непустыми входными потоками; 2) произвольное множество готовых к выполнению процессов переходит из состояния ожидания в состояние выполнения; 3) осуществляется передача данных с произвольного множества готовых к передаче выходов процессов; 4) происходит снятие данных с произвольного множества заполненных выходов системы. (Объединение перечисленных множеств должно быть непустым.) Переход вычислительного процесса из состояния выполнения в состояние ожидания осуществляется после передачи всех результатов процессам-приемникам. Пусть *Q* – это класс

всех состояний системы S. Запись $q \rightarrow q'$ будет означать *достижимость* q' из q за один шаг системы S. Состояние называется *заключительным*, если в нём система S не может сделать ни одного шага. Пусть $I,E \subseteq Q$. Система S *корректна* относительно предусловия I и постусловия E ($[I]\ S\ [E]$), если любое заключительное состояние, достижимое из I, принадлежит E.

Пример 1. В качестве примера ОВС рассмотрим систему СУММА, осуществляющую суммирование элементов строк матрицы A размером $M \times N$, где $N = 2^K$ для некоторого целого k. В этой системе $P_{in}=\{p_1,p_2,...,p_N\}$, $P_{out}=\{p\}$, каждый процесс из P_{oa1o} имеет два входа ($n = 2$) один выход ($m = 1$) и вычисляет одну функцию x_1+x_2 (где x_1,x_2 – переменные, обозначающие содержимое 1-го и 2-го входов, соответственно). На рис. 1 представлена система СУММА для K=3.

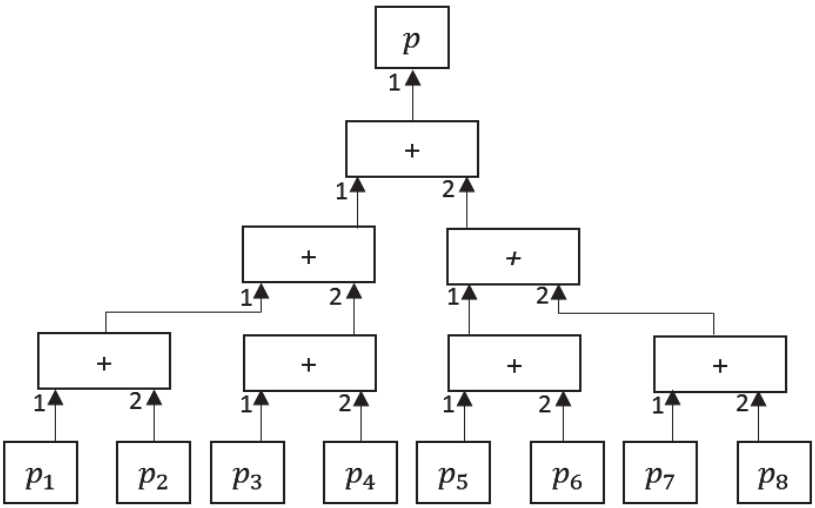


Рис. 1. Система СУММА для K=3

Потоки данных на входах системы задаются элементами столбцов матрицы A. На выходе системы формируется поток значений, в котором первое значение совпадает с суммой элементов первой строки матрицы A, второе значение – с суммой элементов второй строки и т.д. В начальном состоянии входы системы содержат элементы первой строки матрицы A, а остальные входы и выходы процессов свободны.

Спецификация ОВС

Для спецификации L-программой работы системы S сначала определяется язык предметной области – L как язык логики предикатов первого порядка.

Сорта: *ПР* – процессы из P;

ВХ – номера входов процессов $\{1, 2,...,n\}$;

ВЫХ – номера выходов процессов $\{1, 2,...,m\}$;

ДАН – данные;

НАТ – натуральные числа и 0.

Функции: +: *НАТ*×*НАТ* → *НАТ* – сложение;

кол: *ПР*→*НАТ* – задает количество срабатываний процесса с момента начала работы системы;

*данное*in: *ВХ*×*ПР*×*НАТ* → *ДАН* – *данное*in(*k*,*p*,*i*) задаёт данное, содержащееся на *k*-м входе процесса *p* перед (*i*+1)-м срабатыванием *p*, где 0 ≤*i*≤ *кол*(*p*), третий аргумент введён для удобства анализа спецификации; так, выходной поток данных системы на текущий момент для *p*∈ *P*out представляется набором значений *данное*in(1, *p*, *кол*(*p*)),,...,*данное*in (1,*p*,0);

*данное*out: *ПР*×*ВЫХ*×*НАТ* → *ДАН* – *данное*out(*p*,*k*,*i*) задаёт данное, содержащееся на *k*-м выходе процесса *p*∈ *P*calk после *i*-го срабатывания *p*, где 1 ≤*i*≤ *кол*(*p*), также определяет остаток входного потока данных системы на текущий момент для *p*∈ *P*in набором значений *данное*out(*p*,1,*кол*(*p*)), *данное*out (p,1,*кол*(*p*)+1),....

Предикаты: *связь*: *ПР*×*ВЫХ*×*ВХ*×*ПР* – задаёт линии связи;

ввод: *ПР* – задаёт процессы ввода;

вычисление: *ПР* – задаёт вычислительные процессы;

вывод: *ПР* – задаёт процессы вывода;

*заполнен*in: *ВХ*×*ПР* – задаёт наличие данных на входах процессов;

*заполнен*ou: *ПР*×*ВЫХ* – задаёт наличие данных на выходах процессов.

Переменные: *пр*,*пр*':*ПР*; *вх*, *вх*':*ВХ*; *вых*, *вых*':*ВЫХ*; *у*,*х*1,...,*х*n:*ДАН*; *i*,*i*':*НАТ*.

Язык *L* также содержит обычные арифметические функции и отношения. Знаки ∧, ∨, ¬, →, ∀ и ∃ обозначают обычные логические связки и кванторы. Далее под *Q* понимается класс всех *L*-структур (представляющих состояния системы *S*). Специфицирующая *L*-программа [3] действует по шагам, преобразуя одну *L*-структуру в другую. Произвольный шаг этой *L*-программы будет реализовывать шаг работы системы *S*. Согласно своему определению [3] *L-программа* задаётся конечной совокупностью *правил* вида *условие*⇒*действие* (где *условие* – произвольная *L*-формулы, а *действие* – *L*-формула специального вида [3]). Эти *L*-формулы зависят от набора свободных переменных. Правило может исполниться одновременно для нескольких (выбираемых недетерминированно) наборов значений свободных переменных, для которых истинно условие. Кроме того, одновременно могут сработать несколько (также выбираемых недетерминированно) правил. В *L*-программе может использоваться средство синхронизации **$** работы как в рамках одного правила, так и на уровне нескольких правил [3].

Введём следующие обозначения формул:

READYload(*пр*): $\exists y$(*данное*out(*пр*,1,*кол*(*р*)+1)=*y*) – выражает существование данных в остатке входного потока для процесса ввода *пр*;

READYcalc(*пр*): $\forall$*вх*'(*заполнен*in(*вх*',*пр*)) – выражает условие готовности вычислительного процесса к выполнению;

READYsend(*пр*,*вых*,*вх*',*пр'*): *связь*(*пр*,*вых*,*вх*',*пр*')$\wedge$*заполнен*out(*пр*,*вых*) $\wedge\neg$*заполнен*in(*вх*',*пр*')$\wedge$(*пр*'$\neq$*пр* $\rightarrow$ $\forall$*вых*'($\neg$*заполнен*out(*пр*',*вых*')) – выражает условие готовности к передаче данного с выхода *вых* процесса *пр* на вход *вх*' процесса *пр*'.

Специфицирующая *L*-программа состоит из четырех правил, моделирующих соответственно: 1) загрузку данных на входы системы; 2) переход вычислительных процессов из состояния ожидания в состояние выполнения (обратный переход происходит автоматически, после передачи всех результатов приемникам); 3) передачу данных приемникам; 4) снятие данных с выходов системы.

***L*-программа:**

1) *ввод*(*пр*)$\wedge$READYload(*пр*)$\wedge\neg$*заполнен*out(*пр*,1)$\wedge$*i*=*кол*(*пр*)+1 $\Rightarrow$
*заполнен*out(*пр*,1)$\wedge$*кол*(*пр*)=*i*

2) **$** *вх*,*вых*,*y*
вычисление(*пр*)$\wedge$READYcalc(*пр*)$\wedge$
*x*1=*данное*in(*вх*,*пр*,*кол*(*пр*))$\wedge\ldots\wedge$*x*n=*данное*in(*вх*,*пр*,*кол*(*пр*))$\wedge$
(*y*=*f*1(*x*1,...,*x*n)$\wedge$*вых*=1$\vee\ldots\vee$*y*=*f*m(*x*1,...,*x*n))$\wedge$*вых*=m$\wedge$*i*=*кол*(*пр*)+1 $\Rightarrow$
$\neg$*заполнен*in(*вх*,*пр*)$\wedge$*заполнен*out(*пр*,*вых*)$\wedge$*данное*out(*пр*,*вых*,*i*)=*y*$\wedge$*кол(пр)=i*

3) READYsend(*пр*,*вых*,*вх*',*пр'*)$\wedge$*y*=*данное*out(*пр*,*вых*,*кол*(*пр*))$\wedge$*i*'=*кол*(*пр*') $\Rightarrow$
$\neg$*заполнен*out(*пр*,*вых*)$\wedge$*заполнен*in(*вх*',*пр*')$\wedge$*данное*in(*вх*',*пр*',*i*')=*y*

4) *вывод*(*пр*)$\wedge$*заполнен*in(1,*пр*)$\wedge$*i*=*кол*(*пр*)+1 $\Rightarrow\neg$*заполнен*in(1,*пр*)$\wedge$*кол*(*пр*)=*i*

Первое правило может исполниться одновременно для нескольких процессов ввода (значений переменной *пр*) с непустыми входными потоками и незаполненными выходами. При этом значение переменной *i* подбирается так, чтобы увеличить с её помощью количество срабатываний процесса на 1. В результате исполнения действия правила заполняются выходы процессов, и изменяется количество срабатываний.

Второе правило может исполниться одновременно для нескольких вычислительных процессов (значений переменной *пр*), готовых к выполнению. Переменные *x*1,...,*xn* используются для упрощения записи условия правила. Запись вида *fk*(*x*1,...,*xn*) – это *L*-терм, вычисляющий значение функции, соответствующей *k*-му выходу процесса *пр*, где $1\leq k\leq m$. Вместо формулы вида *y* = *fk*(*x*1,...,*xn*) может использоваться *L*-формула, определяющая значение *y* функции *fk*. Синхронизатор **$** *вх*, *вых*, *y* обеспечивает

одновременное вычисление значений всех функций и заполнение этими значениями всех соответствующих выходов процесса *пр*, освобождение всех его входов.

Третье правило моделирует передачу данного с выхода *вых* процесса *пр* на вход *вх*' процесса *пр*' и может исполниться для нескольких связей одновременно (нескольких наборов значений переменных *пр*, *вых*, *вх*', *пр'*, для которых истинно условие правила). Четвёртое правило снимает с входа 1 процесса вывода *пр* (выхода системы) и может это сделать для нескольких процессов вывода (значений переменной *пр*) с заполненным входом.

Пример 2. Обсудим корректность системы СУММА относительно предусловия *I* и постусловия *E*. Дополним язык предметной области *L* функцией *A*: $\{1,2,\ldots,M\}\times\{1,2,\ldots,N\}\to$*ДАН*, задающей входную матрицу системы СУММА. Предусловие *I* описывает возможные начальные состояния системы СУММА и имеет вид

$$\forall пр\ (\neg ввод(пр) \to$$
$$\neg заполнен\text{in}(1,пр)\wedge\neg заполнен\text{in}(2,пр)\wedge\neg заполнен\text{out}(пр,1)\wedge кол(пр)=0)\wedge$$
$$\forall пр\ (ввод(пр) \to заполнен\text{out}(пр,1)\wedge кол(пр)=1)\wedge$$
$$\bigwedge_{1\le i\le M,\,1\le j\le N} данное\text{out}(p\text{j},1,i)=A(i,j)\wedge ввод(p\text{j})$$

Эта формула утверждает следующее: у каждого процесса, отличного от процесса ввода, все входы и выходы свободны, и этот процесс ещё ни разу не срабатывал; выходы всех процессов ввода содержат данные; и элементы первого столбца матрицы *A* образуют входной поток данных, соответствующий процессу ввода *p*1, элементы второго столбца задают входной поток данных, реализуемый процессом *p*2, и т.д. Постусловие *E* предъявляет требования к заключительному состоянию системы СУММА и имеет вид

$$\bigwedge_{1\le i\le M} (данное\text{in}(1,p,i)=(\sum_{j=1}^{N} A(i,j)).$$

Эта формула требует, чтобы первое значение в выходном потоке (*p* – это процесс вывода) совпадало с суммой элементов первой строки матрицы *A*, второе значение в выходном потоке – с суммой элементов второй строки и т.д.

Для анализа корректности системы СУММА может быть применён метод инварианта, описанный, в частности, в [2 и 3]. (Инвариант системы – это множество её состояний, войдя в которое она в нём остаётся во всё время выполнения.) Формула *W* лежит в основе инварианта, позволяющего доказать [*I*] СУММА [*E*], и имеет вид

$$\bigwedge_{1\le i\le M} (кол(p)\ge i \wedge (\sum_{j=1}^{N} A(i,j)) = данное\text{in}(1,p,i) \vee$$

$$кол(p)<i \wedge (\sum_{j=1}^{N} A(i,j)) = [+:x]\ (вычисление(пр) \wedge (кол(пр)+1=i \wedge$$

$$(заполнен\text{in}(1,пр) \wedge x=данное\text{in}(1,пр,i) \vee заполнен\text{in}(2,пр) \wedge x=данное\text{in}(2,пр,i)) \vee$$
$$кол(пр)=i \wedge заполнен\text{out}(пр,1) \wedge x=данное\text{out}(пр,1,,i)) \vee$$
$$ввод(пр) \wedge кол(пр)\le i \wedge x=данное\text{out}(пр,1,,i))$$

Эта формула утверждает для каждого номера строки i матрицы A следующее: либо длина выходного потока больше или равна i, и i-й элемент этого потока совпадает с суммой элементов i-й строки матрицы A либо длина выходного потока меньше i, и тогда сумма элементов i-й строки матрицы A совпадает с суммой значений, содержащихся на входах вычислительного процесса $пр$ (если $кол(пр)+1=i$), значений, содержащихся на выходах вычислительного процесса $пр$ (если $кол(пр)=i$), а также значений входного потока процесса ввода $пр$ (если $кол(пр)\le i$).

Список литературы

1. Крицкий С.П. Модель асинхронных вычислений в структурах и языки программирования // Методы трансляции. – Ростов н/Д., 1981. – С. 92–100.

2. Крицкий С.П., Панков С.В. О верификации асинхронных программ продукционного типа // Программирование. – 1994. – № 5. – С. 40–52.

3. Панков С.В. Формальная спецификация асинхронных параллельных вычислений над общей памятью // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2013): материалы XIII Всерос. конф. г. Нижний Новгород, 14–16 ноября. – Н. Новгород: Изд-во Нижегород. гос. ун-та. – 2013. – С. 195–199.

А.В. Пачин, Ба Хла Тхан, М.В. Коваленко, Д.А. Федяшин

Национальный исследовательский университет «МИЭТ», г. Москва

РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ ВЫЧИСЛЕНИЙ НА ПРИМЕРЕ ОТКРЫТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ BOINC

Рассмотрен один из методов решения проблемы интеграции вычислительных ресурсов, использующий преимущества grid-технологии, а также создание и тестирование grid-среды на основе программного обеспечения BOINC на базе кафедры вычислительной техники НИУ «МИЭТ».

Введение

Необходимость разделять вычислительные задачи и выполнять их одновременно (параллельно) возникла задолго до появления первых вычислительных машин.

Многие современные прикладные задачи настолько сложны, что для их решения необходимо объединять вычислительные мощности нескольких суперкомпьютеров, серверов, рабочих станций или простых персональных компьютеров пользователей.

Такие задачи нередко встречаются в различных сферах исследований или производства: нанотехнологиях; науках о материалах; биоинформационных технологиях; криптоанализе; бизнес-анализе.

Обычно в распоряжении пользователя имеются различные вычислительные ресурсы. Это ставит перед ним задачу организации вычислительной системы с разнородными типами оборудования, что само по себе сопряжено с некоторыми сложностями, обусловленными следующими факторами:

- большим различием в архитектуре используемого оборудования;
- высокой вероятностью потери связи с одним из компонентов системы;
- различием в производительности отдельных компонентов системы, усложняющей рациональное распределение нагрузки между ними.

Данная проблема интеграции разнородных вычислительных ресурсов имеет два основных метода решения.

Первый метод заключается в создании универсального ПО системного уровня, обеспечивающего интеграцию для приложений. Примерами являются средства организации grid-среды, такие как BOINC или Globus Toolkit, позволяющие переносить распределенные программы, которые предназначены для выполнения на многопроцессорных кластерах.

Другой подход заключается в непосредственной интеграции на программном уровне. Примером такого подхода является сервис ориентированная архитектура, в рамках которой приложение, выполняющееся на вычислительном ресурсе, рассматривается как сервис, предоставляющий определенные возможности в рамках заданного протокола.

Проект на основе BOINC

Для исследования решений проблемы интеграции вычислительных ресурсов в компьютерной аудитории кафедры вычислительной техники «МИЭТ» была создана высокопроизводительная вычислительная среда, использующая преимущества grid-технологии. В качестве связующего программного обеспечения был выбран BOINC, система, разработанная в Ка-

лифорнийском университете командой под руководством Дэвида Андерсона, ответственной за создание легендарного проекта SETI@home.

BOINC состоит из программы-клиента, общей для всех BOINC-проектов, составного сервера (термин «составной сервер» означает, что физически сервер может состоять из нескольких отдельных компьютеров) и программного обеспечения. Для выполнения распределенных вычислений используется архитектура клиент-сервер.

Каждая программа-клиент подключается к серверу и получает задание на обработку. Через некоторое время, когда задание выполнено, программа-клиент снова подключается к серверу и передает ему результаты (а также получает новое задание).

В данной схеме существует ряд проблем:

1) программа-клиент, получившая задание, может по различным причинам никогда не выйти снова на связь с сервером;

2) участники проекта могут посылать неверные результаты.

Первую проблему система BOINC обходит, назначая конечный срок, к которому программа-клиент должна вернуть результат вычислений.

Вторая проблема решается за счет передачи одного и того же задания нескольким пользователям. Для каждого задания количество дублирующих друг друга исполнителей индивидуально, но значение по умолчанию платформы BOINC равняется пяти. Когда программа-клиент отчитывается о проведенных вычислениях, сервер сравнивает результаты, присланные разными участниками. Результат, принятый «кворумом» программ-клиентов, считается правильным, а неверные ответы отвергаются.

Для участия в проектах и предоставления ресурсов своего ПК необходима установка клиента BOINC. Данный клиент может быть установлен на любую из существующих сегодня платформ, таких как Windows, Linux, Mac OS, а также мобильный Android. Получить установочный пакет самой новой версии программы можно со страницы официального сайта либо введя соответствующие команды в терминале управления системой (Linux).

Для создания и тестирования проекта на основе BOINC был настроен сервер, а также взят алгоритм распределенного приложения: вычисление определенного интеграла при заданных данных (интервал интегрирования, шаг, количество подзадач). В качестве узлов использовались компьютеры учебной аудитории в количестве 25 штук.

Данные тестов и результаты представлены в таблице. Также на двух графиках (рис. 1, 2) представлено наглядное сравнение результатов тестов 7–9 и 10–12.

Тестирование работы проекта

Номер теста	Интервал интегрирования	Шаг	Количество подзадач	Время выполнения расчетов (с)
1	0…0,01	0,001	10	95
2	0…0,01	0,001	5	45
3	0…0,1	0,01	5	75
4	0…0,1	0,01	7	125
5	0…2	0,1	10	90
6	0…2	0,1	20	130
7	-1…1	0,1	10	105
8	-1…1	0,1	15	113
9	-1…1	0,1	20	130
10	0…4	0,0000001	100	16096
11	0…4	0,0000001	200	14576
12	0…4	0,0000001	350	12187

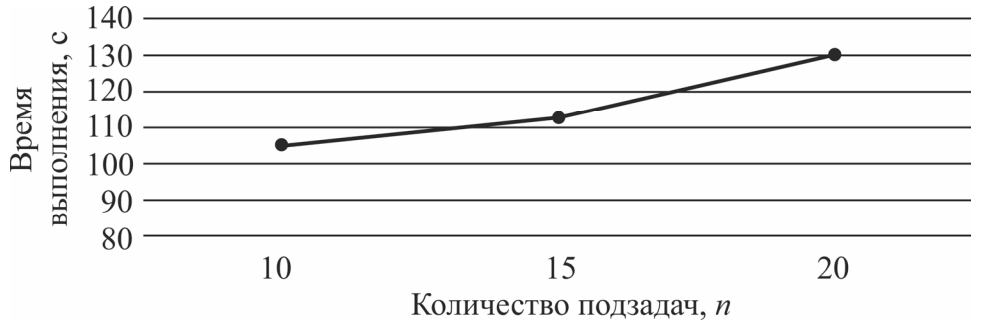


Рис. 1. Сравнение тестов 7–9

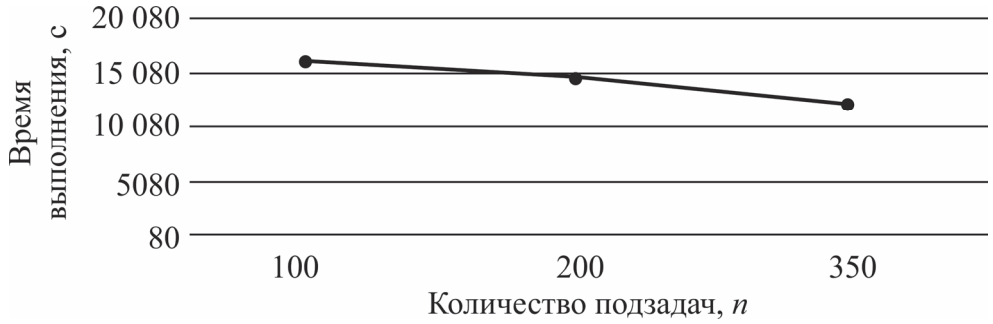


Рис. 2. Сравнение тестов 10–12

Анализируя результаты проделанной работы, можно с уверенностью сказать, что создание небольшой локальной системы распределенных вычислений на основе платформы BOINC вполне осуществимо.

Нам удалось настроить сервер, развернуть на его основе проект с распределенным приложением и подключить к нему небольшое количество узлов, которые обеспечили выполнение поставленной задачи (расчет интеграла).

Результаты тестов, в свою очередь, открыли некоторые особенности и проблемы, связанные с вычислительной системой, основанной на BOINC.

1. Сравнивая время выполнения некоторых тестов, можно заметить, что с увеличением количества подзадач, а следовательно, и количества задействованных узлов скорость выполнения операции вычисления падает (тесты 7–9). Это противоречит общему положению о том, что производительность должна увеличиваться с увеличением количества узлов. Возможно, это связано с тем, что сложность самой задачи недостаточно высока, чтобы задействовать преимущества распределенного метода, и некоторое количество времени тратится на пересылку и ожидание ответа от конкретного узла. После того как входные данные были изменены на подходящие объемным вычислениям, было отмечено снижение времени выполнения задачи (тесты 10–12).

2. Не все тесты были удачными. Эта проблема связана с сильной зависимостью GRID-технологии от качества и работы сети. В нашем случае если конкретный узел, получив задание, отключался от сети, то его результат вычислений (work unit) не мог быть получен сервером. Это приводило либо к некорректному итоговому результату, либо к бесконечному ожиданию завершения процесса вычисления.

3. Не до конца понятно, как происходит раздача подзаданий на стороне сервера BOINC. Скорее всего не существует никакой оценки производительности системы, выступающей узлом, поэтому распределение подзаданий может быть не совсем рациональным в целях увеличения производительности. Наиболее сложные по вычислениям фрагменты общей задачи вполне могут вычисляться не самыми производительными ПК, а высокопроизводительные, наоборот, могут быстро завершить расчеты и «простаивать».

Все эти проблемы и вопросы требуют дальнейшего исследования работы распределенных систем вычислений и, в частности, открытой программной платформы BOINC. Создание более крупного и комплексного проекта могло бы дать больше ответов и понимания принципа работы этого метода.

Список литературы

1. Посыпкин М.А. Решение задач глобальной оптимизации в среде распределенных вычислений // Программные продукты и системы. – 2010. – № 1. – С. 23–29.

2. Посыпкин М.А. Методы и распределенная программная инфраструктура для численного решения задачи поиска молекулярных кластеров с минимальной энергией // Труды ПаВТ. – 2009. – С. 269–281.

3. Хританков А.С., Посыпкин М.А. Анализ эффективности распределенных вычислительных систем при решении больших задач // Современные информационные технологии и ИТ-образование: материалы конференции. – М.: МАКС ПРЕСС, 2008. – С. 528–536.

4. Посыпкин М.А., Сухорослов О.В., Храпов Н.П. Технологии распределенных вычислений: учеб. пособие. – М., 2012.

5. McGilvary G. How to create a BOINC Projec Manual. – URL: www.ibm.com/developerworks/ru/library/l-grid/index.html

А.Ю. Пирова

Нижегородский государственный университет им. Н.И. Лобачевского

РАЗРАБОТКА ПЕРЕУПОРЯДОЧИВАТЕЛЯ СИММЕТРИЧНЫХ РАЗРЕЖЕННЫХ МАТРИЦ ДЛЯ СИСТЕМ С ОБЩЕЙ ПАМЯТЬЮ

Рассматривается задача уменьшения заполнения фактора разреженной матрицы при разложении Холецкого. Предлагается параллельная реализация алгоритма переупорядочения разреженных матриц, основанного на многоуровневом методе вложенных сечений. Реализация ориентирована на системы с общей памятью и выполнена с помощью технологии OpenMP. Приводятся результаты вычислительных экспериментов, показывающие приемлемую масштабируемость реализации.

Введение

При использовании прямых методов решения систем линейных уравнений с разреженной матрицей (разложения Холецкого в случае симметричной положительно определенной матрицы) возникает *заполнение*: число ненулевых элементов в факторе матрицы много больше числа ненулевых элементов исходной матрицы. Для того чтобы снизить заполнение фактора, к исходной матрице применяется процедура переупорядочения ее строк и столбцов. Эффективность его выполнения критична для последующего решения системы, поскольку определяет затраты памяти для хранения фактора матрицы и затраты времени, необходимого на факторизацию. Качество переупорядочения также влияет на потенциал параллелизма при выполнении факторизации. Нахождение перестановки, минимизирующей размер фактора или число вычислительных операций для ее факторизации, является NP-полной задачей [11]. В практических приложениях для ее решения применяются две группы эвристических методов – методы минимальной степени и методы вложенных сечений. Обзор методов переупорядочения разреженных матриц представлен в работе [13].

В настоящий момент существует ряд успешных реализаций указанных алгоритмов, работающих как последовательно, так и параллельно для систем с распределенной памятью. Среди широко распространенных пакетов отметим библиотеки METIS [44], Scotch [8] и их аналоги для систем с распределенной памятью ParMETIS [5] и PT-Scotch [1]. Однако вопрос о возможности создания параллельных инструментов переупорядочивания для систем с общей памятью, сочетающих эффективное использование вычислительных ресурсов современных многоядерных и мультиядерных систем и качество решения задачи, является открытым. Последние работы по данному направлению посвящены решению смежной задачи – разработкам параллельных алгоритмов для разделения графов [6, 7].

В данной работе представлены текущие результаты по разработке нового параллельного алгоритма для переупорядочения разреженных матриц, ориентированного на системы с общей памятью. Работа выполняется в рамках проекта по созданию прямого решателя СЛАУ на кафедре МО ЭВМ факультета ВМК ННГУ им. Н.И. Лобачевского [12].

Постановка задачи

Рассмотрим постановку задачи переупорядочения в следующем виде [11].

Пусть дана разреженная симметричная матрица $A=\left(a_{ij}\right)$ размером $n\times n$. Построим *граф матрицы* $G=(V,E)$, в котором каждая вершина v_i, $i=\overline{1,n}$ сопоставлена строке матрицы *i*, а ребро $\left(v_i,v_j\right)\in E$ сопоставлено недиагональному ненулевому элементу матрицы $a_{ij}: a_{ij}\neq 0$, $i,j\in\{1,2,...,n\}, i\neq 1$. 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 $\pi=\left(\pi_1,\pi_2,...,\pi_n\right)$ – перестановка множества вершин *V*. *Заполнение* $F(\pi)$, порожденное перестановкой π, – это множество ребер, добавленных при исключении вершин $\pi_1,\pi_2,...,\pi_n$. Нахождение перестановки π^* такой, что число добавленных ребер минимально, является NP-полной задачей [11]:

$$\pi^*=\arg\min\{|F(\pi)|\}.$$

Под *качеством перестановки* будем понимать число ненулевых элементов фактора Холецкого матрицы после применения к ней данной перестановки.

Многоуровневый метод вложенных сечений

Метод вложенных сечений состоит в следующем: граф матрицы разделяется на два несвязных подграфа при помощи вершинного разделителя, вершины разделителя нумеруются и удаляются из графа, а процесс переупорядочения продолжается рекурсивно для порожденных подграфов до тех пор, пока не будут занумерованы все вершины. Модификации метода различаются способом нахождения разделителя.

С середины 1990-х гг. широко используются модификации метода вложенных сечений, основанные на многоуровневой процедуре разделения графа. Такой подход был применен в работах Кариписа и Кумара [3], Пеллигрини [8], Хендриксона и Ротберга [2] и др. В многоуровневом методе выделение разделителя графа выполняется в три этапа. На этапе огрубления с помощью паросочетания ребер строится последовательность сжимающихся графов, в которой структура каждого следующего в последовательности графа отражает структуру предыдущего. На этапе разделения выполняется поиск разделителя с несвязных компонент самого маленького грубого графа. На этапе развертывания выполняется проекция разделителя самого грубого графа на исходный граф через всю последовательность промежуточных графов.

Программная реализация

Ранее была разработана библиотека MORSy, в которой выполнена последовательная программная реализация новой модификации алгоритма переупорядочения симметричных разреженных матриц, основанного на многоуровневом методе вложенных сечений. Подробно библиотека MORSy описана в работе [9].

В рамках данной работы выполняется разработка параллельной версии библиотеки MORSy для систем с общей памятью. Для распараллеливания используется технология OpenMP.

Параллельный алгоритм переупорядочения основан на следующих идеях. Нахождение разделителя в конкретном подграфе соответствует вычислительной задаче, которую выполняет свободный поток. Входные и выходные данные таких задач – подграфы и найденные для них разделители – хранятся в общей для потоков очереди. Потоки, находясь в активном ожидании, выбирают нерешенные задачи из очереди. После нахождения разделителя текущего подграфа, графы, на которые он распадется, заносятся в конец очереди задач. Процесс заканчивается, когда найден разделитель для всех порожденных подграфов. Таким образом, вычислительный процесс соответствует обходу дерева подграфов в ширину, которое неизвестно заранее, а формируется по мере вычислений.

Результаты вычислительных экспериментов

Для текущей программной реализации был проведен ряд вычислительных экспериментов по переупорядочению матриц из широко распространенной коллекции университета Флориды [10]. Параметры тестовых матриц и аппаратного окружения приведены в табл. 1, 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	726713	2903837	0,000010
boneS10	914 898	40 878 708	0,000097
Emilia_923	923 136	40 373 538	0,000094
audikw_1	943 695	77 651 847	0,000173
bone010	986 703	36 326 514	0,000074
ecology2	999 999	2 997 995	0,000005
thermal2	1 228 045	4 904 179	0,000006
StocF-1465	1 465 137	11 235 263	0,000010
Hook_1498	1 498 023	59 374 451	0,000052
Flan_1565	1 564 794	114 165 372	0,000093
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)

Параметры переупорядочивания в MORSy позволяют настроить работу алгоритма с приоритетом на минимизацию времени работы и на максимизацию качества переупорядочения в смысле числа ненулевых элементов фактора. В данной серии вычислительных экспериментов для каждой матрицы были выбраны параметры, дающие наилучшее качество работы. Табл. 3 показывает полученные размеры факторов матриц.

Таблица 3

Число ненулевых элементов фактора матриц при лучшем по качеству переупорядочении

Название матрицы	Порядок	Число ненулевых элементов в факторе
Pwtk	217 918	46 784 875
msdoor	415 863	52 085 831
parabolic_fem	525 825	24 923 337
tmt_sym	726713	28 741 732
boneS10	914 898	295 723 050
Emilia_923	923 136	1 650 689 751
audikw_1	943 695	1 200 984 910
bone010	986 703	1 035 907 995
ecology2	999 999	30 081 507
thermal2	1 228 045	51 356 445
StocF-1465	1 465 137	1 072 623 748
Hook_1498	1 498 023	1 617 131 106
Flan_1565	1 564 794	1 415 670 363
G3_circuit	1 585 478	91 292 425

Время работы и масштабируемость параллельной версии переупорядочения на тестовых матрицах при работе в 1–8 потоков приведены в табл. 4, 5.

Таблица 4

Время работы переупорядочивателя MORSy при работе в несколько потоков. Для каждой матрицы параметры переупорядочения лучшие по качеству

Название матрицы	1 поток	2 потока	4 потока	8 потоков
Pwtk	0,67	0,42	0,31	0,30
Msdoor	1,22	0,77	0,52	0,41
parabolic_fem	12,28	7,28	4,56	3,48
tmt_sym	19,70	11,31	7,52	5,61
boneS10	17,69	10,74	7,80	6,75
Emilia_923	7,55	4,88	3,47	3,08
audikw_1	8,34	5,31	4,13	3,89
bone010	16,22	9,47	6,44	5,36
ecology2	20,64	11,94	7,63	5,78
thermal2	38,36	23,22	15,03	11,38
StocF-1465	62,47	37,24	23,55	18,36
Hook_1498	34,22	20,45	17,16	16,55
Flan_1565	17,11	10,92	8,41	7,52
G3_circuit	20,53	12,02	7,81	6,28

Таблица 5

Масштабируемость переупорядочивателя MORSy при работе в несколько потоков. Для каждой матрицы параметры переупорядочения лучшие по качеству

Название матрицы	2 потока	4 потока	8 потоков
Pwtk	1,59	2,15	2,26
msdoor	1,59	2,36	3,00
parabolic_fem	1,69	2,69	3,52
tmt_sym	1,74	2,62	3,51
boneS10	1,65	2,27	2,62
Emilia_923	1,55	2,18	2,45
audikw_1	1,57	2,02	2,14
bone010	1,71	2,52	3,03
ecology2	1,73	2,71	3,57
thermal2	1,65	2,55	3,37
StocF-1465	1,68	2,65	3,40
Hook_1498	1,67	1,99	2,07
Flan_1565	1,57	2,04	2,28
G3_circuit	1,71	2,63	3,27

Как видно из результатов экспериментов, масштабируемость разработанной параллельной версии на 8 потоков более 3,5 для 3 матриц, от 3 до 3,5 для 5 матриц, менее 3 – для 6 матриц. Лучшие результаты распараллеливания были получены на матрицах, для которых подграфы, образованные при исключении разделителя в процессе переупорядочения, были близких размеров.

Заключение

В настоящий момент разработана параллельная программная реализация алгоритма переупорядочения разреженных матриц для систем с общей памятью, выполненная с помощью технологии OpenMP. Текущие результаты распараллеливания показывают приемлемую масштабируемость для данного класса задач.

Направлениями дальнейших исследований являются развитие алгоритма и программной реализации с целью уменьшения времени ее работы и увеличения масштабируемости.

Автор благодарит И.Б. Меерова, А.В. Сысоева, Е.А. Козинова за участие в обсуждении работы.

Работа выполнена при поддержке гранта РФФИ № 14-01-31455 и организационной поддержке лаборатории Intel-ННГУ «Информационные технологии».

Список литературы

1. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering // Parallel Computing. – 2008. – Vol. 34. – No. 6. – P. 318–331.

2. Hendrickson B., Rothberg. Improving the runtime and quality of nested dissection ordering // SIAM J. on Scientific Computing. – 1999. – Vol. 20. – P. 468–489.

3. 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.

4. 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].

5. Karypis G., Kumar V. ParMETIS: Parallel graph partitioning and sparse matrix ordering library // Technical report TR 97-060. University of Minnesota, 1997.

6. LaSalle D., Karypis G. Multi-threaded Graph Partitioning // Parallel & Distributed Processing (IPDPS), 2013. – P. 225–236.

7. Pellegrini F. Shared memory parallel algorithms in Scotch 6. – URL: [http://graal.ens-lyon.fr/MUMPS/doc/ud_2013/Pellegrini.pdf].

8. 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.

9. Pirova A., Meyerov I. MORSy – a new tool for sparse matrix reordering // An International Conference on Engineering and Applied Sciences Optimization M. Papadrakakis, M.G. Karlaftis, N.D. Lagaros (eds.) Kos Island, Greece, 4-6 June 2014. – P. 1952–1964.

10. The University of Florida Sparse Matrix Collection – URL.: [www.cise.ufl.edu/research/sparse/matrices/].

11. 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.

12. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей / Е.А. Козинов, И.Г. Лебедев, С.А. Лебедев, А.Ю. Малова [и др.] // Вестник ННГУ. – Н. Новгород: Изд-во ННГУ, 2012. – № 5(2). – С. 376–384.

13. Малова А.Ю. Обзор методов переупорядочения разреженных матриц // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2012): материалы 12-й Всерос. конф. Нижний Новгород, 26 – 28 ноября 2012 г. – Н. Новгород: Изд-во ННГУ, 2012. – С. 257–264.

П.В. Писарев, В.Я. Модорский

Пермский национальный исследовательский политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ГИДРОУПРУГИХ ПРОЦЕССОВ В СОЕДИНИТЕЛЬНОМ КАНАЛЕ ДВУХСТУПЕНЧАТОГО ЦЕНТРОБЕЖНОГО НАСОСА С ИСПОЛЬЗОВНИЕМ GPU

Цель исследования заключалась в изучении влияния конструктивных и технологических параметров на напряженно-деформированное состояние стенок соединительного канала, нагруженного динамическим потоком жидкости, и разработке рекомендаций по уменьшению интенсивности непрогнозируемых режимов в каналах наукоемких изделий с использованием бытовых видеокарт.

Сформулированы физическая и математическая модели гидроупругих процессов в модельном двухступенчатом центробежном насосе. Произведены численные эксперименты в связанной постановке по моделированию потока в соединительном канале с различной толщиной стенок, а также с учетом прилегающих свободных объемов. Построены зависимости максимальных напряжений в стенках соединительного канала от их толщины. В рамках вычислительных экспериментов рассматривался расчетный вариант при угле сдвига фаз $\varphi = 0$. Проведено исследование масштабируемости задачи на расчетных сетках различной размерности, при этом ускорение оценивалось относительно расчета, выполненного на CPU.

Надежность и безотказность двух ступенчатых центробежных насосов, как любых машин и механизмов, определяется целым рядом факторов, в том числе условиями их эксплуатации и способностью адаптироваться к изменяющимся условиям работы. Одной из основных тенденций проектирования и создания многоступенчатых центробежных насосов является повышение их производительности за счет роста числа оборотов рабочих колес. При этом могут проявляться непрогнозируемые динамические эффекты, связанные с взаимодействием рабочей жидкости и тонкостенной конструкции соединительного канала [1–4].

Разработка расчетных методов и специальных моделей, позволяющих определять амплитуды пульсаций давления и вибрационные нагрузки в конструкции соединительного канала путем численного моделирования нестационарных процессов в проточном тракте ступенчатого центробежного насоса, является весьма актуальной задачей. Вместе с тем моделирование гидроупругих динамических процессов в связанной постановке требует больших вычислительных ресурсов, таких как многопроцессорная техника, и высокопроизводительных вычислений на графических процессорах.

Разработка методик расчета гидроупругих процессов в соединительных каналах многоступенчатых центробежных насосов позволит получить необходимые данные для правильного выбора конструктивных и режимных параметров с целью снижения динамической нагруженности при минимальных материальных и временных затратах [5].

Наиболее доступным высокопроизводительным вычислением является вычисление на базе графических ускорителей. Вычислительные средства с многоядерными процессорами для параллельных векторных вычислений, используемых в основном в трехмерной компьютерной графике, способны достигать высокой пиковой производительности, которая с использованием универсальных процессоров может быть реализована только в кластерных системах, которые недоступны многим исследователям и специалистам на предприятиях. Конечно, максимальная скорость достигается лишь в ряде удобных задач и имеет некоторые ограничения, но такие устройства уже начали довольно широко применять в сферах, для которых они изначально и не предназначались. Среди преимуществ таких вычислений стоит отметить общедоступность и относительно невысокую стоимость при достаточно высокой производительности. В то же время существует ряд определенных трудностей с организацией таких вычислений. В качестве одной из основных можно выделить проблему, связанную с передачей данных из глобальной памяти компьютера в память устройства и наоборот. В настоящее время данная проблема является одним из самых узких мест в алгоритмах, разрабатываемых под GPU. Стоит отметить, что в новых релизах CUDA появилась возможность выполнять асинхронные переводы из памяти в память даже в то время, когда вычислительное ядро находится в состоянии счета.

В данной работе использовалась бытовая видеокарта NVIDIA GeForce 650, GPU с архитектурой Kepler. В расчете использовалась конфигурация из четырёх потоковых блоков с 768 ядрами CUDA и 64 блоками текстурирования. Совокупная ширина шины памяти составляет 128 бит. Также расчеты проводились на восьмиядерном процессоре AMD FX-8320 3.5GHz 16Mb DDR3-1866.

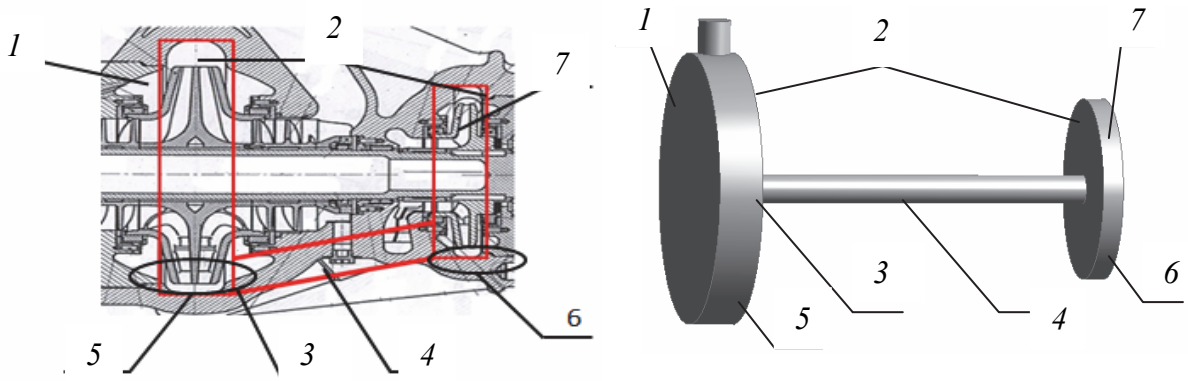
Моделирование взаимодействия осциллирующего потока жидкости и конструкции канала осуществляется с помощью междисциплинарного open-source пакета прикладных программ OpenFOAM с использованием технологии MPI. Для решения поставленной задачи не разрабатывался новый исходный код а использовались заголовочные файлы из стандартных библиотек OpenFOAM

Реальный исследуемый объект обладает следующими особенностями (рис. 1, *а*): *1* осевой двухсторонний встречный вход в первую ступень; *2* проточный тракт первой и второй ступени с учетом рабочих колес; *3* пло-

щадь поперечного сечения; соединительный канал *4* имеет определенную длину; *5, 6, 7* – пульсирующие подача и отвод жидкости в тракте центробежного двухступенчатого насоса, обусловленные воздействием лопаток рабочих колес первой и второй ступеней.

Для решения модельной задачи геометрия расчетной области (см. рис. 1, *а*) представляет собой совокупность трех объемов (рис. 1, *б*). Первый соответствует проточному тракту первой ступени, второй – проточному тракту соединительного канала с учетом его значимых геометрических размеров (длины и объема). Поперечное сечение модельного соединительного канала представляет собой круг. Третий объем соответствует проточному тракту второй ступени.

Проточный тракт первой ступени имеет один вход и два выхода. Первый выход – в соединительный канал и направлен вправо (см. рис. 1, *б*). Второй выход имеет большее сечение и направлен вверх (см. рис. 1, *б*). Этот выход имеет также круговое поперечное сечение. Проточный тракт второй ступени имеет один вход (см. рис. 1, *б* – слева) и один выход (см. рис. 1, *б* – справа) с круговыми поперечными сечениями.



а *б*

Рис. 1. Исследуемый двухступенчатый центробежный насос:
а – конструкция реального объекта;
б – общий вид проточного тракта модельного насоса

При описании граничных условий учитывалось, что крыльчатки насосов первой и второй ступеней жестко закреплены на одном валу, их вращение происходит с заданной частотой и моделируется пульсирующей подачей жидкости в проточный тракт, что можно описать синусоидальной функцией подачи.

При описании граничных условий учитывалось, что крыльчатки насосов первой и второй ступеней жестко закреплены на одном валу, их вращение происходит с заданной частотой и моделируется пульсирующей подачей жидкости в проточный тракт, что можно описать синусоидальной функцией подачи.

Угол сдвига фаз φ колебаний на входе относительно колебаний на выходе из соединительного канала определяется взаимным расположением крыльчаток друг относительно друга на валу и не изменяется в процессе работы.

Тогда граничное условие входа в первую ступень (см. рис. 1, *б*) в запишется в виде

$$V_1(t) = V_{\text{ном1}} + V_{A1}\sin(\omega t+\varphi), \tag{1}$$

где $V_{\text{ном1}}$ – номинальная скорость подачи жидкости в первую ступень; V_{A1} – амплитуда колебания скорости потока; ω – угловая частота вращения крыльчатки; t – время расчета,

$$\omega = 2\pi z f_r, \tag{2}$$

где z – количество лопаток крыльчатки; f_r = 142 Гц – частота вращения ротора.

Граничное условие выхода из второй ступени насоса (см. рис. 1, *б*) в запишется в виде

$$V_2(t) = V_{\text{ном2}} + V_{A2}\sin\omega t, \tag{3}$$

где $V_{\text{ном2}}$ – номинальная скорость выхода жидкости из второй ступени насоса; V_{A2} – амплитуда колебания скорости потока.

Для контроля пульсаций давления в расчетную область были помещены математические «датчики» давления. Все «датчики» располагаются на оси симметрии модели. Первый «датчик» располагается на входе в соединительный канал, второй, третий, четвертый – в соединительном канале, пятый – на выходе из соединительного канала (рис. 2).

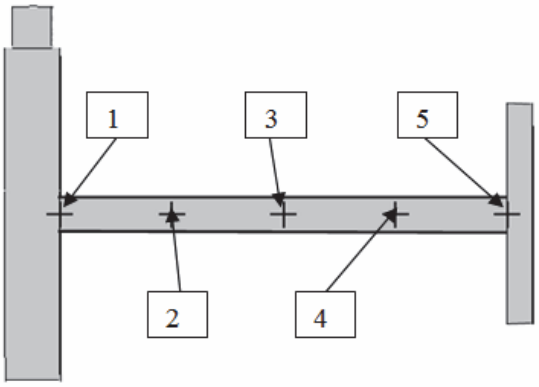


Рис. 2. Схема расположения контрольных точек

При рассмотрении результатов вычислительных экспериментов, полученных по модельному насосу, выяснилось, что амплитуда колебаний давления ΔP на в ходе в соединительный канал в контрольной точке *1*,

наибольшая в том случае, если угол сдвига фаз φ равен π. В то же время наименьшая амплитуда колебаний давления наблюдается при угле сдвига фаз φ, равном 0.

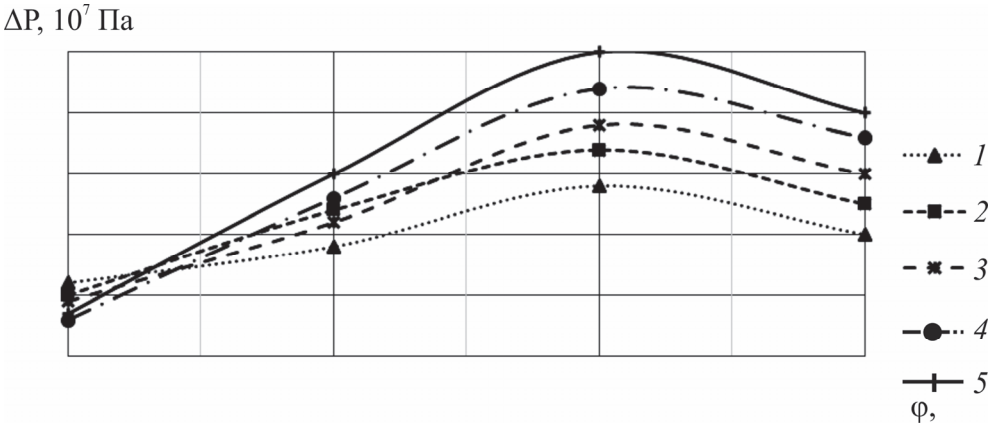


Рис. 3. График зависимости Δ*P* от φ в различных контрольных точках

Максимальное усиление обнаружилось в контрольной точке *5* (выход соединительного канала), а минимальное усиление в контрольной точке *1* (начало соединительного канала) (рис. 3).

Кроме того, в ходе проведения исследований выявлено, что амплитуда колебаний давления в контрольной точке зависит от места ее расположения в соединительном канале. Увеличение амплитуды наблюдается при движении жидкости в сторону второй ступени при φ = 90, 180, 270°. Ослабление колебательных процессов наблюдается при движении жидкости в сторону второй ступени при φ = 0°, как и в несвязанной постановке. По результатам численных экспериментов построена зависимость максимальных напряжений от номера положения рабочего колеса первой ступени (угла сдвига фаз) рис. 4.

Рис. 4. Зависимость максимальных напряжений в стенках соединительного канала при различных положениях рабочего колеса первой ступени

Вместе с тем в работе проведено исследование масштабируемости задачи на расчетных сетках различной размерности, ускорение оценивалось относительно расчета, выполненного на CPU (рис. 5).

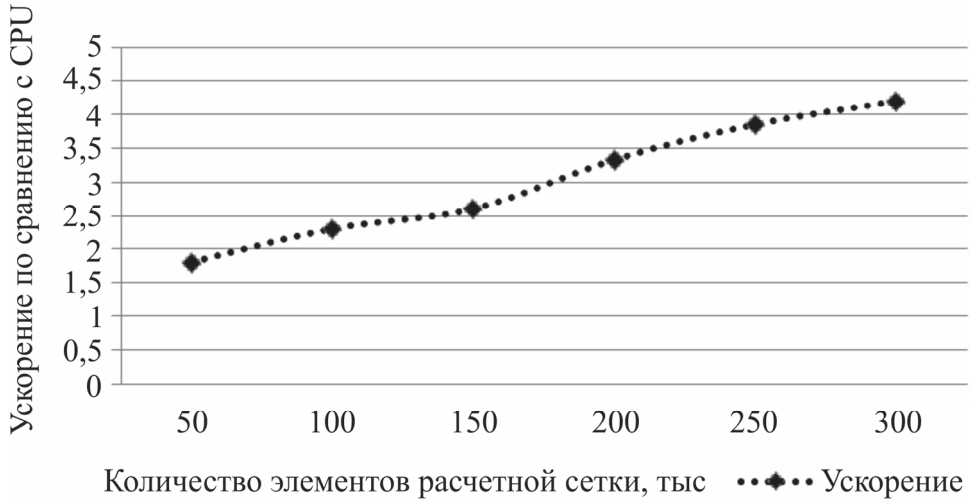


Рис. 5. График зависимости ускорения по сравнению с GPU от количества расчетных элементов

Как видно из графика, максимальное ускорение, как и предполагалось, наблюдается при максимальном количестве расчетных элементов по сравнению с расчетами, проводимыми на CPU. Таким образом, использование GPU для данного класса задач представляется весьма эффективным.

Выводы

1. Разработана методика расчета гидроупругих процессов в соединительном канале двухступенчатого центробежного насоса.

2. Проведено сравнение полученных результатов с результатами расчетов в несвязанной постановке.

3. Выявлено, что максимальные напряжения и перемещения наблюдаются в начале соединительного канала для $\varphi = 0$ и в конце соединительного канала для остальных углов сдвига фаз в области максимальных амплитуд колебаний давления.

4. Построена зависимость максимальных напряжений от номера положения рабочего колеса первой ступени.

Список литературы

1. Численное моделирование колебательных процессов в центробежном насосе / П.В. Писарев, В.Я. Модорский, Д.В. Щенятский, И.А. Арбузов, Р.В. Бульбович, Б.Е. Кириевский, А.А. Ташкинов // Научно-технический вестник Поволжья. – 2012. – № 3. – С. 44–49.

2. Численное моделирование колебательных процессов в соединительном канале модельного насоса / П.В. Писарев, И.А. Арбузов, А.А. Ташкинов, Д.В. Щенятский, Б.Е. Кириевский, Р.В. Бульбович, В.Я. Модорский // Научно-технический вестник Поволжья. – 2012. – № 6. – С. 100–103.

3. Писарев П.В., Модорский В.Я. Численный анализ динамического напряженно-деформированного состояния конечномерного цилиндра, нагруженного гидродинамическим потоком жидкости // Механика микронеоднородных материалов и разрушение: материалы VI Всерос. конф. Екатеринбург, 24 – 28 мая 2010; ИМАШ УрО РАН. – Екатеринбург, 2010. – С. 41.

4. Моделирование динамического напряженно-деформированного состояния конечномерного цилиндра, нагруженного гидродинамическим потоком жидкости на многопроцессорном вычислительном комплексе. Научный сервис в сети Интернет: экзафлопсное будущее / П.В. Писарев, А.А. Писарева, Д.В. Зимин, Л.Н. Бутымова, Д.Ф. Гайнутдинова, В.Я. Модорский // Труды международной суперкомпьютерной конференции (19–24 сентября 2011 г., г. Новороссийск). – М.: Изд-во МГУ, 2011. – 643 с.

5. Численное моделирование взаимодействия в динамической системе «газ–конструкция» / П.В. Писарев, Д.В. Зимин, В.Я. Модорский, Ю.В. Соколкин// Механика и процессы управления: материалы XXXXI Всерос. симпозиума. Т. 1. – М., 2011. – С. 127.

6. Писарев П.В., Модорский В.Я., Соколкин Ю.В. Решение междисциплинарных задач на примере взаимодействия струи горячего газа с металлической преградой // Механика и процессы управления: материалы XXXXI Всерос. симпозиума. Т. 1. – М., 2011. – С. 123.

Р.В. Писарев, П.В. Писарев, В.Я. Модорский

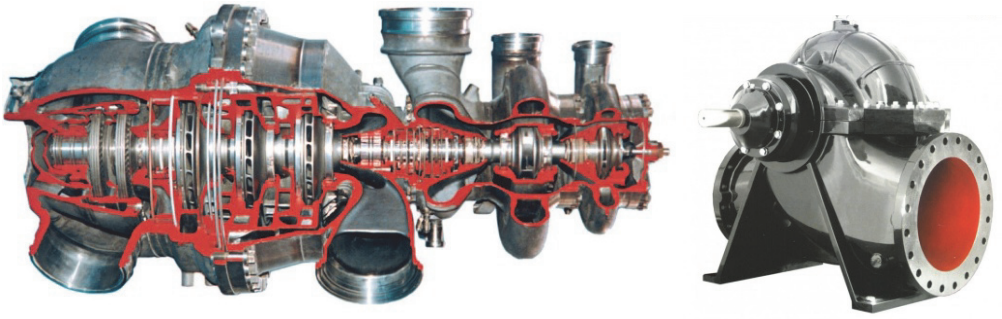
Пермский национальный исследовательский политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ВИБРАЦИЙ ЛОПАТОК ЦЕНТРОБЕЖНОГО НАСОСА, НАГРУЖЕННЫХ ПОТОКОМ ЖИДКОСТИ

При разработке центробежных насосов одной из основных является задача по расчету вибраций лопаток рабочих колес. Известно, что вибрации могут приводить к снижению рабочих характеристик и в некоторых случаях к разрушению элементов конструкции вследствие высокого уровня динамических напряжений, вызванных неравномерностью потока жидкости. На сегодняшний день большинство методик расчета резонансных явлений в лопатках центробежных насосов базируются на дорогостоящих натурных экспериментах. В связи с этим разработка методики расчета

вибраций лопаток является актуальной задачей. Для решения поставленной задачи был проведен ряд вычислительных экспериментов по расчету гидродинамических и прочностных характеристик модельного центробежного насоса в связанной постановке *Fluid-Structure Interaction*. В качестве программного обеспечения использовался междисциплинарный многопроцессорный пакет прикладных программ *ANSYS*.

В настоящее время центробежные насосы высокого давления являются наиболее распространенным типом насосного оборудования и используются во многих отраслях промышленности. Например, для транспортировки воды, перекачки вязких жидкостей в магистральных трубопроводах, агрессивных жидкостей в авиационной и космической технике, а также для перекачивания жидкостей различной плотности в технологических и производственных процессах (рис. 1).



а *б*

Рис. 1. Центробежные насосы: *а* – центробежный насос в составе ТНА РД-120; *б* – нефтяной центробежный насос

Несмотря на широкую распространенность и высокую степень изучения процессов, протекающих в центробежных насосах, конструкторы и инженеры достаточно часто сталкиваются с проблемами, возникающими в ходе их эксплуатации. Наиболее распространенные – это повышенный шум, вибрации лопаток рабочих колес и корпуса центробежного насоса, срыв напора. Все это может быть вызвано множеством факторов: отклонениями в размерах машины при ее изготовлении; увеличением потерь в лабиринтных уплотнениях, возникающих из-за увеличения зазоров в уплотнительных кольцах сверх нормативных; увеличением уровня вибрации насоса в результате некачественной сборки, монтажа, возникающих дефектов или кавитации в насосе; изменением диаметра рабочего колеса, погрешности при проведении балансировочных работ; смещением выходных кромок половинок рабочих колес относительно друг друга; неравенством щелевых зазоров уплотнения колеса; влиянием вязкости рабочей жидкости; содержанием газа в рабочей жидкости; изменением давления на входе и другие.

В связи с вышесказанным разработка методики расчёта вибрации лопаток, нагруженных динамическим потоком жидкости, с использованием современных пакетов прикладных инженерных программ является актуальной задачей.

Целью исследования является разработка методики расчета вибраций лопаток рабочего колеса центробежного насоса, нагруженного динамическим потоком жидкости. В рамках данной работы проводится численный эксперимент в связанной односторонней постановке по определению собственных частот и форм колебаний лопаток рабочего колеса центробежного насоса с использованием многопроцессорного программного комплекса *ANSYS*, а также анализ влияния нагрузки, передаваемой на рабочее колесо центробежного насоса, на собственные частоты и формы колебаний отдельно взятой его лопатки.

Проточный тракт исследуемого центробежного насоса ступени имеет один вход и один выход. На боковой поверхности (рис. 2, *а*) модели расположен кольцевой осевой вход. Рабочее колесо представляет собой твердое тело. Геометрическая модель построена с учетом тангенциального расположения выхода. Учитывалось геометрическое расширение отвода. Твердотельная модель для данной задачи была построена в инженерном пакете *Solid Works*; для проведения вычислительного эксперимента было необходимо построить геометрическую модель, состоящую из двух регионов:

1) жидкостной регион (см. рис. 2, *а*) представляет собой проточный тракт центробежного насоса;

2) твердотельный регион представляет собой рабочее колесо, которое помещено в проточный тракт насоса соосно.

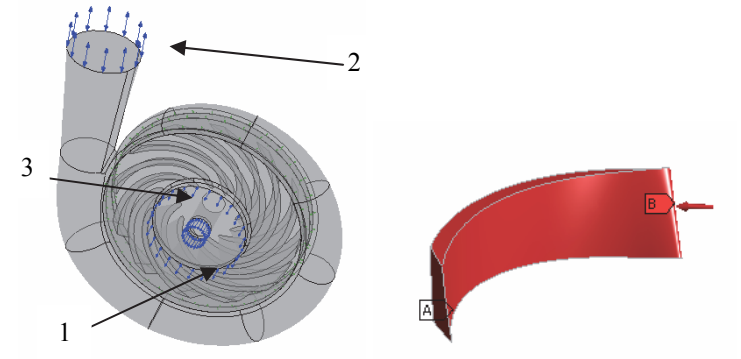
Для улучшения сходимости расчета и снижения погрешностей получаемых результатов создавалась расчетная сетка, элементы которой имеют равномерную призматическую форму. Количество элементов расчетной сетки составило 35 млн.

Для решения поставленной задачи сформулирована следующая физическая модель: конструкция полагается трехмерной (x, y, z); расчет нестационарный; стенки конструкции не поглощают и не выделяют тепло; сила трения между стенками и рабочей жидкостью не учитывается; в качестве рабочей жидкости используется вязкая вода; стенки проточного тракта не деформируются; сила тяжести не учитывается; давление на входе подается через осевой кольцевой вход; реализуется Лагранжев подход. Колесо представляет собой вращающееся тело, состоящее из 12 одинаковых спрофилированных лопаток. В качестве материала рабочего колеса назначалась сталь. Процессы, протекающие в конструкции, рассматривались в линейной постановке.

При описании граничных условий учитывалось, что рабочее колесо *3* (см. рис. 2, *а*) насоса жестко закреплено на валу, и его вращение происходит с заданной частотой:

$$\omega = 2\pi z f_r,$$

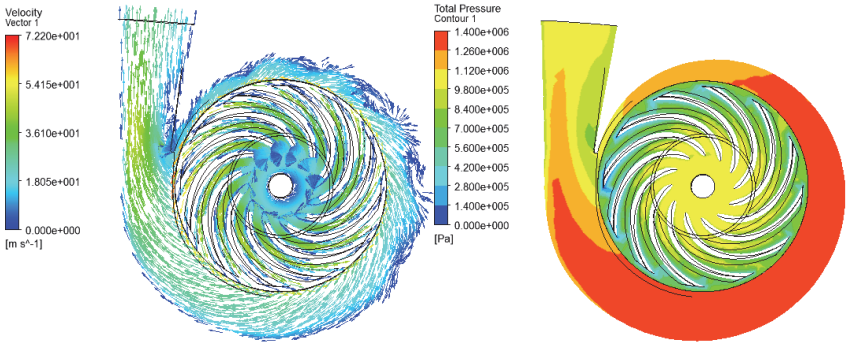
где z – количество лопаток крыльчатки; f_r = 75 Гц – частота вращения ротора.



а *б*

Рис. 2. Граничные условия: *а* – жидкостный регион: *1* – свободный вход-выход, давление на входе 3 атм.; *2* – выход; *3* – скорость вращения рабочих колес 4500 об/мин; *б* – передаваемая нагрузка на лопатку: *A* – консольное закрепление по нижней кромке; *B* – передаваемая через интерфейс нагрузка из гидродинамического расчета

В качестве нагрузок для конструкционного анализа использовались нагрузки из гидродинамического расчета, импортированные через интерфейс взаимодействия двух сред. По результатам вычислительных экспериментов получены поля распределения давления и скоростей (рис. 3). Схема организации распределенных вычислений в связанной постановке на двух вычислительных узлах кластера ПНИПУ представлена на рис. 4.



а *б*

Рис. 3. Результаты гидродинамического расчета:
а – векторы скоростей; *б* – распределение полного давления

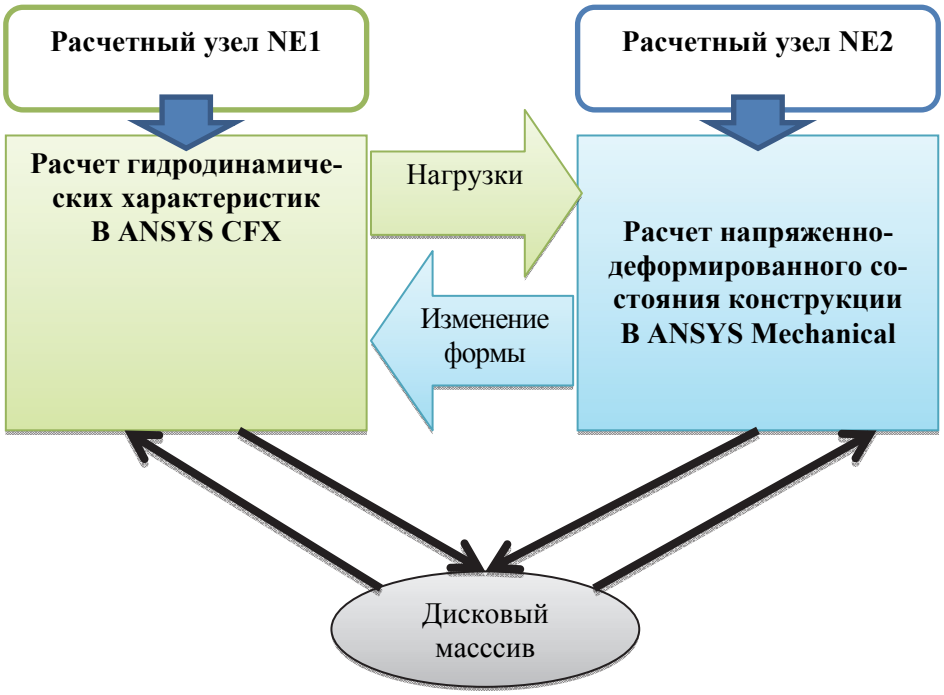


Рис. 4. Схема организации распределенных вычислений

По результатам проведения численных экспериментов определены нагрузки для конструкционного анализа. Проведен линейный статический анализ в связанной постановке. Определены напряжения и перемещения лопатки рабочего колеса центробежного насоса, общие максимальные перемещения на лопатках составили 1,2 мм, максимальные напряжения по критерию Мизеса 29.419 МПа.

В рамках вычислительных экспериментов был проведен модальный анализ, в ходе которого определены формы и частоты собственных колебаний при частоте вращения ротора $n = 0$ и $n = 4500$ об/мин, нагруженного потоком жидкости. Результаты представлены в таблице.

Результаты модального анализа

Номер частоты	Частота колебаний при $n = 0$ об/мин, Гц	Частота колебаний при $n = 450$ об/мин, Гц
1	557,67	583
2	1844,5	1860
3	5226,6	5347,3
4	6871,4	6871,1
5	8849,9	8892,7
6	8961,7	8975

Выводы

1. Разработана методика расчета вибраций лопаток рабочего колеса центробежного насоса, нагруженного динамическим потоком жидкости в

связанной однонаправленной постановке, с использованием многопроцессорной вычислительной техники.

2. Определены формы и частоты собственных колебаний при частоте вращения ротора $n = 0$ и $n = 4500$ об/мин, нагруженного динамическим потоком жидкости.

3. В ходе проведения вычислительного эксперимента определены напряжения и перемещения лопатки рабочего колеса.

Список литературы

1. Писарев П.В., Модорский В.Я. Численное моделирование ударно-волнового нагружения соплового аппарата на многопроцессорном вычислительном комплексе// Аэрокосмическая техника, высокие технологии и инновации – 2009: материалы XII Всерос. науч.-техн. конф. – 2009. – С. 357–359.

2. Писарев П.В., Модорский В.Я. Численный анализ динамического напряженно-деформированного состояния конечномерного цилиндра, нагруженного гидродинамическим потоком жидкости// Механика микронеоднородных материалов и разрушение: материалы VI Всерос. конф. Екатеринбург, 24 –28 мая 2010; ИМАШ УрО РАН. – Екатеринбург, 2010. – С. 41–41.

А.С. Порозов

Южно-Уральский государственный университет, г. Челябинск

ПРИМЕНЕНИЕ МНОГОЯДЕРНЫХ СОПРОЦЕССОРОВ INTEL XEON PHI ДЛЯ ВЫПОЛНЕНИЯ АГРЕГАТНЫХ ФУНКЦИЙ В ПАРАЛЛЕЛЬНЫХ СУБД

Работа посвящена исследованию эффективности использования многоядерных сопроцессов в Intel Xeon Phi при выполнении агрегатных функций в парралельных СУБД.

В результате выполнения поставленных задач были выявлены объемы данных, при которых использование сопроцесса является эффективным.

Применение вычислительных систем, построенных на базе многоядерных сопроцессоров, является одним из актуальных на сегодняшний день направлений исследований. Современные многоядерные сопроцессоры и графические ускорители могут быть с успехом применены для решения множества задач [2]. Одной из задач, способных эффективно использовать многоядерные сопроцессоры Intel Xeon Phi, является задача параллельной обработки баз данных [1,5]. Существующие на сегодняшний день работы показывают, что многоядерные сопроцессоры Intel Xeon Phi могут эффективно применяться для выполнения специфичных для СУБД операций, например соединения[1], поиска [5], сортировки [6].

Описывается незаконченное исследование, посвященное оценке эффективности использования многоядерных сопроцессоров Intel Xeon Phi, при параллельной обработке запросов, требующих вычисления агрегатных функций. На текущий момент был разработан эмулятор [3,4] параллельной СУБД, выполняющий запросы агрегатных функций к базе данных. В статье работа эмулятора рассматривается применительно к функциям SUM, AVG, MAX, MIN, часто используемых в реальных СУБД. Эмулятор включает в себя версии запроса для CPU и для многоядерного сопроцессора Intel Xeon Phi. Алгоритмы для CPU и для Intel Xeon Phi реализованы с помощью технологии OpenMP. Работа с многоядерным сопроцессором производится в режиме offload. Время передачи данных и время, требуемое на инициализацию потоков на ЦПУ и сопроцессоре, не учитывается. На рис. 1, 2 представлены результаты экспериментов, моделирующих агрегатные функции AVG и SUM.

На рис. 3, 4 представлены результаты экспериментов, моделирующих выполнение агрегатных функций MAX и MIN.

На основе проведенных экспериментов можно сделать вывод, что многоядерные сопроцессоры Intel Xeon Phi могут эффективно использоваться в СУБД для вычисления агрегатных функций. При размере обрабатываемого отношения базы данных менее 4 ГБ сопроцессор значительно выигрывает в скорости обработки у центрального процессора. Также во время данного исследования было выявлено, что первая передача данных на сопроцессор занимает гораздо больше времени, чем все последующие. Вероятно, это связано с особенностями архитектуры памяти многоядерного сопроцессора Intel Xeon Phi.

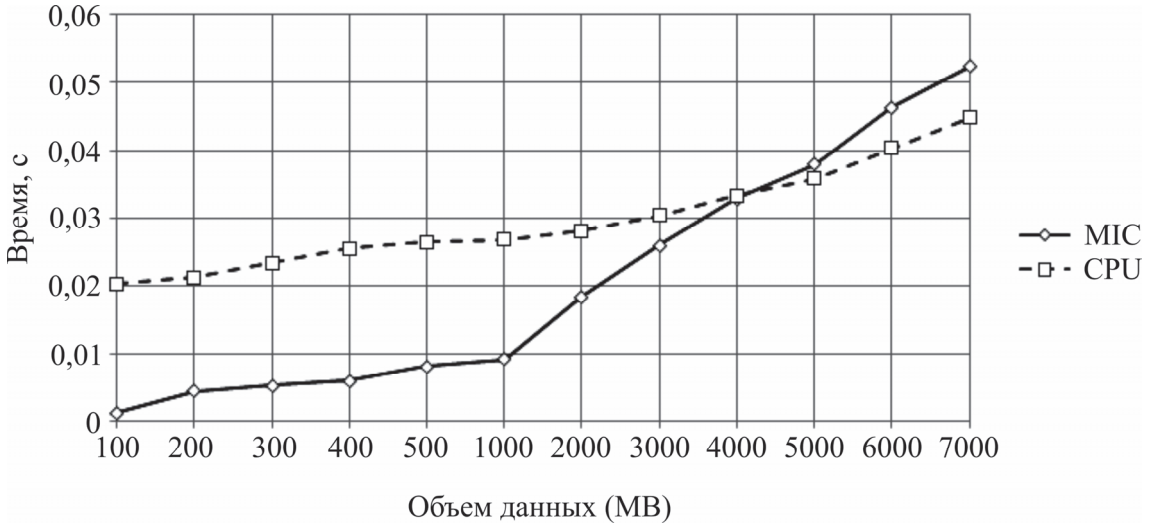


Рис. 1. Время выполнения функции AVG в зависимости от объема обрабатываемых данных

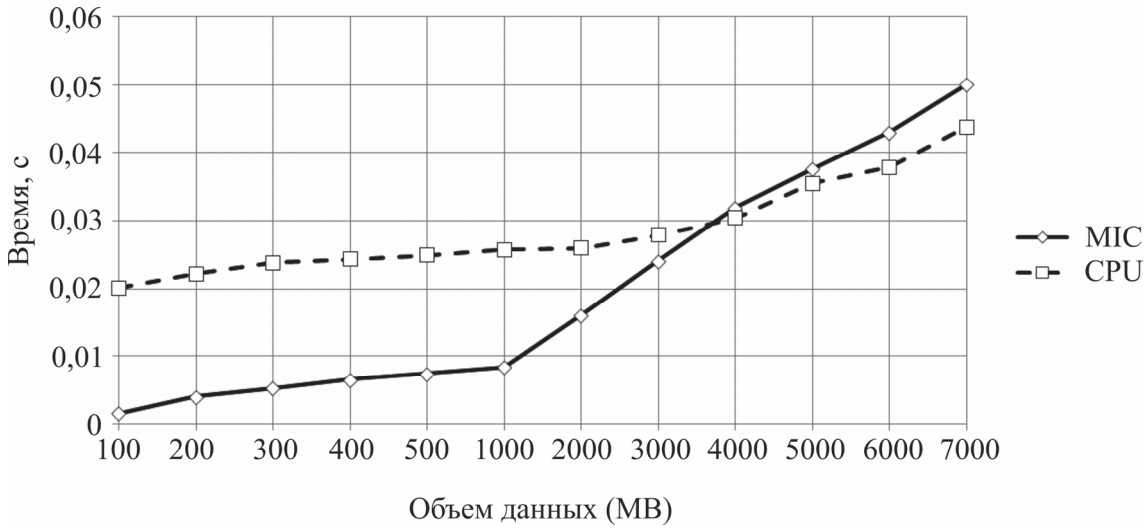


Рис. 2. Время выполнения функции SUM в зависимости от объема обрабатываемых данных

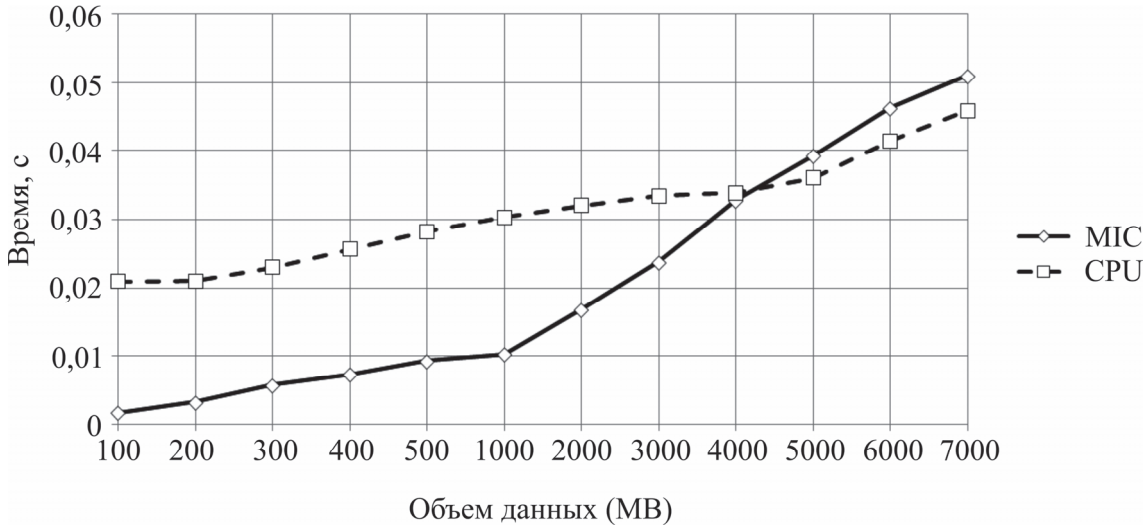


Рис. 3. Время выполнения функции MAX в зависимости от объема обрабатываемых данных

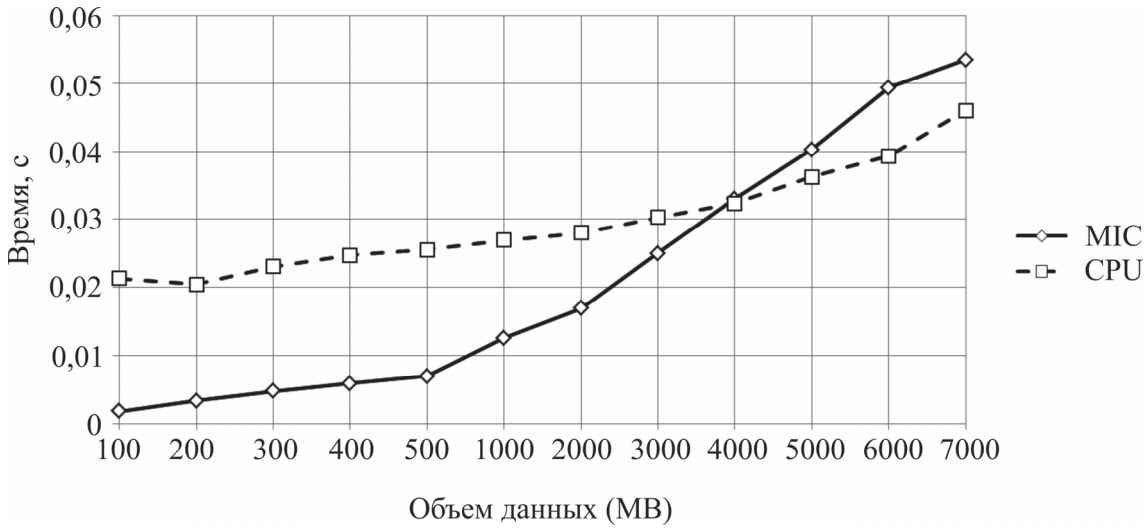


Рис. 4. Время выполнения функции MIN в зависимости от объема обрабатываемых данных

Направление дальнейших исследований будет связано с оценкой эффективности разработанных методов выполнения агрегатных функций в сочетании со сжатием данных и исследованием применения разработанных методов к базам данных, объем которых значительно превышает размер памяти сопроцессора, путем разбиения обрабатываемых данных на сегменты подходящего размера. Предполагается, что при сжатии и разбиении данных архитектура Intel MIC покажет гораздо лучшие результаты, чем ЦПУ, что позволит увеличить производительность при работе со сверхбольшими данными при выполнении запросов, использующих агрегатные функции.

Список литературы

1. Беседин К.Ю., Костенецкий П.С. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями // Программные системы: теория и приложения. Институт программных систем им. А.К. Айламазяна РАН. – 2014. – Т. 5, № 1(19). – С. 91–110.

2. Костенецкий П.С. Обработка запросов на кластерных вычислительных системах с многоядерными ускорителями // Вестник ЮУрГУ. Вычислительная математика и информатика. – 2012. – № 47 (306). – Вып. 2. – С. 59–67.

3. Костенецкий П.С., Соколинский Л.Б. Моделирование иерархических многопроцессорных систем баз данных // Программирование. – 2013. – Vol. 39.

4. Костенецкий П.С., Соколинский Л.Б. Моделирование параллельных систем баз данных: учеб. пособие. – Челябинск: Фотохудожник, 2012. – 78 с.

5. Designing fast architecture-sensitive tree search on modern multicore/many-core processors / C. Kim, J. Chhugani, N. Satish [et al.] // ACM Trans. Database Syst. – 2011. – Vol 36. – No. 4. – P 22:1–22:34.

6. Fast sort on CPUs GPUs and Intel MIC architectures / N. Satish, C. Kim, J. Chhugani [et al.]. – Intel Labs, 2010.

С.В. Пыстогов

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

ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПОЛНООБЪЕКТНЫМИ БАЗАМИ ДАННЫХ КАРТОГРАФИЧЕСКИХ СЦЕН С АССОЦИАТИВНОЙ ЗАЩИТОЙ

Анализируются две программно-перестраиваемые версии серверной части оригинальной системы полнообъектных баз данных картографических сцен с ассоциативной защитой: моно- и мультикластерная. Дается сравнительная оценка эффективности обеих версий по результатам программного эксперимента на платформе HPC-кластера.

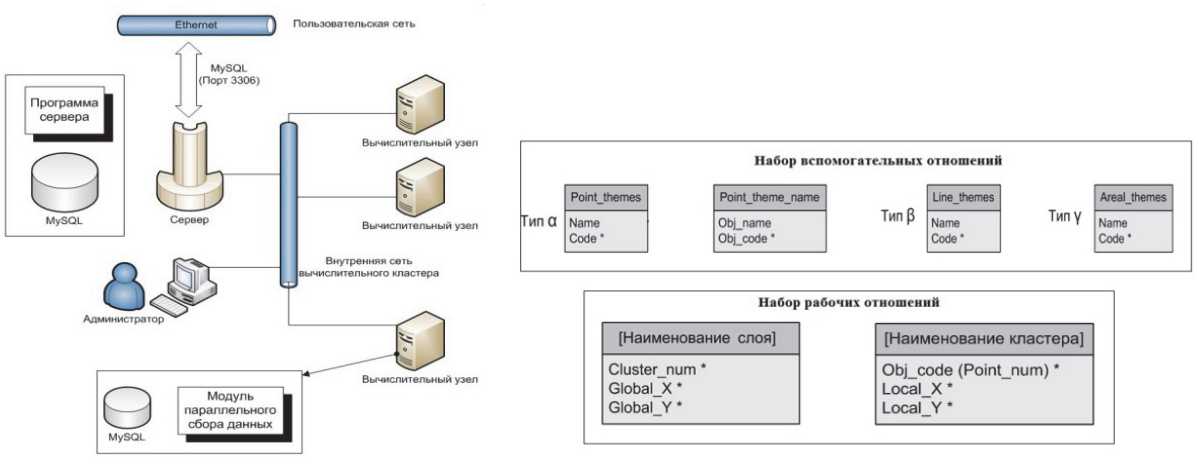
Введение

Кластерная система управления полнообъектными базами данных картографических сцен с ассоциативной защитой [ПБД КС АЗ] (далее система) служит для хранения в распределенном виде защищенной информации о картографических сценах и для обработки потока пользовательских (клиентских) запросов на получение этой информации. Под картографической сценой понимается результат дистанционного зондирования поверхности, преобразованный в векторное представление и описание расположенных на сцене объектов в терминах «объекты-координаты» [1]. Под полнообъектностью понимается возможность обрабатывать все типы объектов: точечные, линейные и площадные.

Для защиты данных применяется двумерно-ассоциативный механизм маскирования [2]. При генерации картографических баз данных используется кодирование информации о типах объектов и их координатах трехразрядными числами − от 000 до 999. При генерации ПБД КС АЗ выполняются операции по созданию тематического слоя, разбиению слоя (и объектов на нем) на кластеры, перевод глобальных координат точечных (узловых точек линейных и площадных) объектов к локальным координатам, добавление пустых (несущественных) объектов/узловых точек и т.д. [3].

Система (серверная часть) имеет архитектуру, показанную на рис. 1, *а*.

Основные принципы построения и клиент-серверного взаимодействия системы описываются в работе [3]. За основу берется «интеллектуальная» файл-серверная организация взаимодействия: функции хранения, функции предварительной обработки поступающих запросов и сбор результатов. Клиентская часть системы функционирует на рабочих станциях пользователей (клиентов).



а *б*

Рис. 1. Архитектура сервера (*а*); схема специализированной БД (*б*)

Полнообъектная база данных картографических сцен с ассоциативной защитой (ПБД КС АЗ) или база данных картографических сцен со свойствами полноты и ассоциативной защиты – база данных, описывающая в терминах «объект-координаты» точечные, линейные и площадные объекты векторизованных участков местности, в которой конфиденциальная информация об объектах сцены хранится в сокрытом виде.

Ранее разработанная схема полнообъектной БД картографических сцен с ассоциативной защитой [3] предполагает единообразный формат хранения данных для всех типов объектов (рис. 1, *б*). Несанкционированному пользователю невозможно определить тип описываемых объектов по хранимой в сокрытом виде информации.

Для реализации серверной части системы предлагаются два подхода [3,4]. Каждый подход имеет свои преимущества и недостатки.

1. Архитектура «*монокластер*»: в каждый момент времени все вычислительные узлы серверной части обрабатывают только один запрос. При этом каждая база данных равномерно распределяется по узлам вычислительного кластера. Присутствие фактора барьерной синхронизации ограничивает масштабируемость системы.

2. Архитектура «*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. При этом каждый узел содержит полную копию всех ПБД КС АЗ. Достаточно мощный вычислительный кластер разбивается на ряд монокластеров, каждый из которых обрабатывает свой запрос. Базы данных реплицируются по монокластерам. Число узлов во всех монокластерах одинаково и близко к порогу масштабируемости.

Экспериментальные исследования

Сравнительная оценка эффективности моно- и мультикластерных архитектур по критериям масштабируемости и производительности получена путем анализа экспериментальных данных. Эксперимент проводился на платформе вычислительного кластера Sun с 22 8-ядерными узлами.

Выявлялось суммарное время обработки пакета запросов $T_{общ}$ и для каждого клиентского запроса – время задержки $T_{зд}$ от момента его поступления в систему до момента отправки ответа.

$$T_{зд} = T_{обр} + T_{ожд} + T_{пробр},$$

где $T_{обр}$ – время обработки запроса сервером; $T_{ожд}$ – время простоя запроса в очереди запросов сервера; $T_{пробр}$ – время предобработки запроса, включающее раскрытие сокрытых в запросе параметров, синтаксический, семантический анализ, права пользователя на исполнение запроса и т.п.

Полученные данные используются для подсчета математического ожидания (M ($T_{зд}$)) и среднеквадратического отклонения σ ($T_{зд}$).

$$\mathrm{M}\,(T_{зд}) = (\sum T_{зд}) / n,$$

n – число обработанных запросов.

$$\sigma = \sqrt{\mathrm{M}[T_{зд}^{k} - \mathrm{M}(T_{зд}^{k})]^2},$$

k – номер запроса.

Для проведения экспериментов:

1. Разработан исследовательский прототип системы [3].

2. Предложен представительский тест (ПТ). Состоит из 20 пользовательских селективных запросов. Выборка объектов − по координатам. Например, запрос: select * from <*kbd_name*> where X>230000 and X<570000 and Y>1570006 and Y<718000;

Здесь *kbd_name* – имя соответствующей ПБД КС АЗ.

3. Для удобства проведения эксперимента полнообъектная база данных была представлена совокупностью трех наборов различных ПБД КС АЗ: точечных, линейных и площадных объектов.

Принципы генерации БД:

1. Генерируются по три ПБД КС АЗ для каждого типа объектов (всего 9) на области размером 2000 × 2000 км.

2. Базы данных генерируются в прямой зависимости от сформированного ПТ. Число объектов в генерируемых базах данных одного типа пропорционально друг другу в следующем соотношении: БД1 (90 Мб) : БД2 : БД3 = 1 : 2 : 4. Размещение объектов на плоскости при генерации БД

случайно, но строго отвечает условиям координатного поиска объектов для каждого из 20 запросов ПТ (условие *where*).

3. В целях упрощения допускаются самопересечения для линейных и площадных объектов.

4. Для генерации информации об объектах использовался стандартный ГПСЧ языка Си – функция *rand()*. Период генератора 2^{32}.

5. Присвоение реальных координат картографической сцены *X*, *Y* для точечных объектов и узловых точек линейных и площадных объектов происходит из диапазона $\overline{\{0, 1999999\}}$ в результате генерации. Полученные картографические сцены со множеством расположенных на них объектов точечного, линейного и площадного типов затем преобразовывались в ПБД КС АЗ согласно принятой схеме и используемой стратегии ассоциативной защиты картографических сцен [2].

Исследование было ограничено случаем селективных запросов, так как особенности выполнения запросов изменения в мультикластерном подходе приводят к появлению больших пиков на графиках временных задержек. Исследование мультикластера ограничивалось вариантом одноузловых компонентов-монокластеров.

Было проведено три эксперимента на исследовательском прототипе системы в обеих архитектурах: моно- и мультикластер.

1. Установление границы масштабируемости и характера изменения параметров $M(T_{зд})$ и $\sigma(T_{зд})$ с ростом *N* для архитектуры монокластера на ПТ из 20 запросов на всех сгенерированных ПБД КС АЗ для каждого типа объектов. Оценка значений M и σ проводилась для последней секции потока, полученного конкатенацией трех одинаковых ПТ.

2. То же для архитектуры мультикластера на том же ПТ и ПБД КС АЗ.

3. Сравнение производительности и характеристик $M(T_{зд})$ и $\sigma(T_{зд})$ обеих архитектур на непрерывном потоке из 1000 запросов.

Далее приводятся результаты только для БД точечных объектов.

Эксперимент 1. Результаты – на рис. 2, *а*. По оси абсцисс число узлов в кластере. По оси ординат – суммарное время обработки пакета в секундах для трех БД.

Согласно рисунку:

- система в архитектуре монокластер имеет явно выраженный порог масштабируемости;
- порог масштабируемости зависит от размеров ПБД КС АЗ, т.е. от числа обрабатываемых записей БД. Для БД1 порог равен 3 узлам. Для БД2 – 4, БД3 – 4,5.

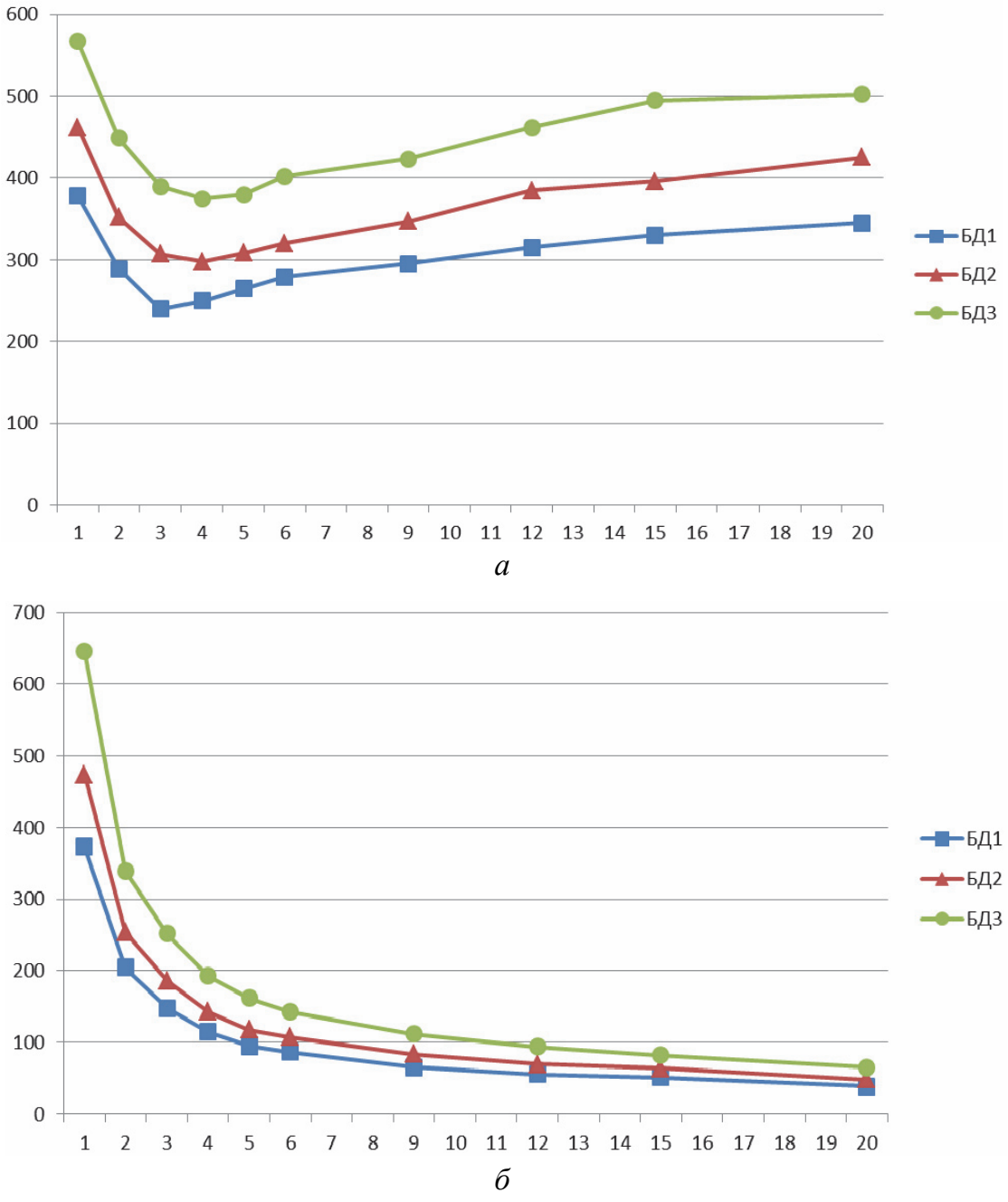


Рис. 2. Монокластер (*а*); мультикластер (*б*)

Эксперимент 2. Согласно графику (рис. 2, *б*: по оси ординат – суммарное время обработки ПТ на мультикластере) рост производительности системы при увеличении числа узлов близок к линейному. В табл.1 приведены показатели улучшения производительности системы «мультикластер» для разного числа монокластеров-компонентов (узлов) относительно системы с одним монокластером-компонентом ($T_{общ1}$) для БД1.

Изменения значения параметра М($T_{зд}$) для разных N аналогичны изменениям $T_{общ}(N)$ – наблюдается близкий к линейному спад среднего времени задержки с ростом числа узлов.

Таблица 1

Показатели улучшения производительности системы «мультикластер»

Параметры	Число узлов, N								
	2	3	4	5	6	9	12	15	20
$T_{общ}$ (N)	205	148	115	95	87	65	56	51	39
$T_{общ1}$ / $T_{общ}$ (N), где $T_{общ1}$ = 373	1,82	2,52	3,24	3,93	4,29	5,73	6,66	7,31	9,56

На рис. 3 представлен сравнительный график σ($T_{зд}$) для одной из ПБД КС АЗ (БД1) при работе в архитектурах моно- и мультикластер.

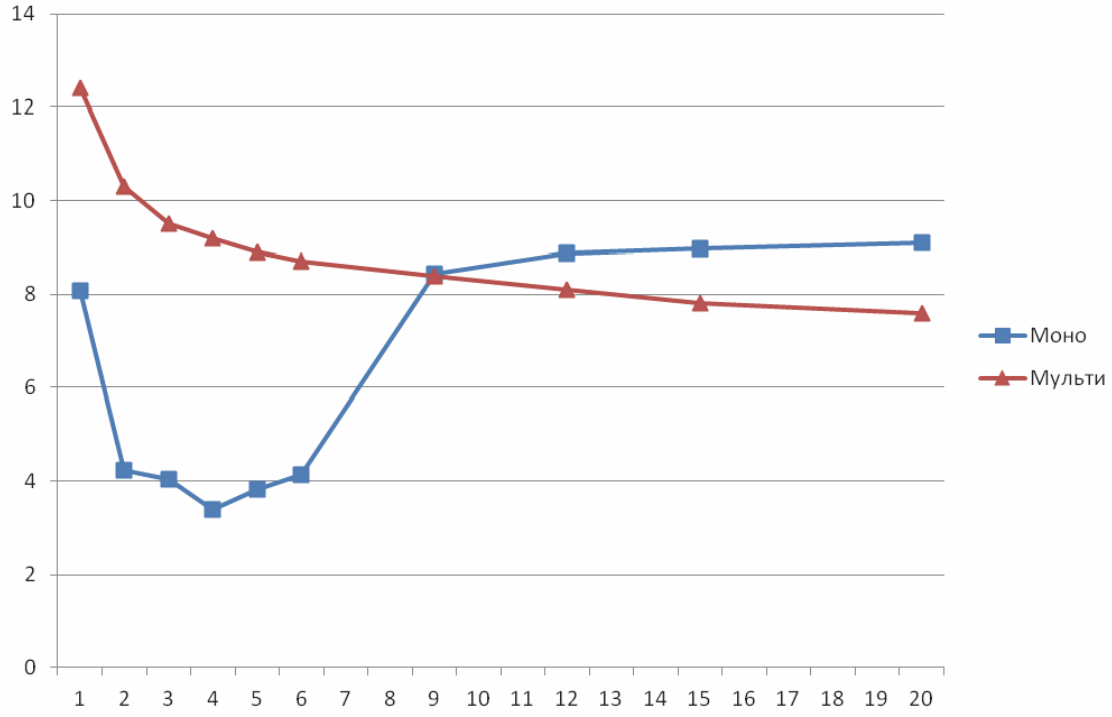


Рис. 3. Сравнительный график σ($T_{зд}$) для БД1

Из рисунка видно, что значения σ($T_{зд}$) для архитектуры «монокластер» минимальны на границе масштабируемости. Для обеих архитектур изменение σ с ростом N аналогично поведению $T_{общ}$. При больших N мультикластер предпочтителен.

Эксперимент 3. Сравнение проводилось на БД точечных объектов среднего размера, БД2. Для монокластера было выбрано граничное число узлов, 4 (рис. 2, *а*). Для мультикластера проводились два эксперимента: при N = 4 и N = 9. В табл. 2 приведены полученные значения M($T_{зд}$), σ($T_{зд}$) и $T_{общ}$ для случайно сформированного потока из 1000 запросов. Запросы не обязательно отвечают оговоренному ранее условию (более реальный сценарий).

Таблица 2

Значения $M(T_{зд})$, $\sigma(T_{зд})$, $T_{общ}$ для случайно сформированного потока запросов

Система	Среднее время задержки $M(T_{зд})$, с	Отклонение $\sigma(T_{зд})$, с	Время обработки пакета $T_{общ}$, с
Монокластер, 4 узла	359	41	27343
Мультикластер, 4 узла	96	29,7	9172
Мультикластер, 9 узлов	37	14,2	4154

Мультикластер имеет выигрыш над монокластером по всем рассмотренным характеристикам. Существенная разница между показателями $\sigma(T_{зд})$, приведенными в табл. 3, и $\sigma(T_{зд})$ из рис. 3 обусловлена:

– проведением *эксперимента 3* над более объемной ПБД КС АЗ (БД2);

– использованием существенно большего по размеру потока, случайно сформированного на более мощном множестве запросов.

Заключение

Эксперименты, проведенные на исследовательском прототипе системы, выявили наличие выраженного порога масштабируемости при работе в архитектуре монокластера. Это обусловливается потерями на накладные расходы при передаче данных между узлами и наличием барьерной синхронизации. Архитектура мультикластера показывает близкий к линейному рост производительности при увеличении числа узлов. Ухудшений из-за возможного возникновения «узкого места» в виде канала между одним управляющим узлом и множеством вычислителей, передающих результаты запросов, не наблюдалось.

Мультикластерная реализация предпочтительна при обработке непрерывных потоков запросов и эпизодичности запросов изменения. Об этом говорят полученные результаты для суммарного времени обработки пакета, среднего времени задержки запроса и среднеквадратичного отклонения.

Монокластер предпочтителен при обработке редко поступающих одиночных запросов на выборку больших областей либо при высокой плотности запросов изменения. Целесообразно применение моноархитектуры и при генерации новой ПБД КС АЗ или изменении защитного ключа в системе *операция*, при которой необходимо провести раскрытие и повторное маскирование всех скрываемых параметров. С такими задачами значительно быстрее справится монокластер.

Список литературы

1. Шаши Шекхар, Санжей Чаула. Основы пространственных баз данных. – М.: КУДИЦ-ОБРАЗ, 2004. – 336 с.

2. Райхлин В.А., Вершинин И.С., Гибадуллин Р.Ф. Конструктивное моделирование систем в приложении к защите данных картографии // Методы моделирования: тр. республ. науч. семинара АН РТ. – Казань: ФЭН, 2010. – Вып. 4. – С. 68–95.

3. С.В. Пыстогов. Моделирование процессов в файл-сервере СУБД полнообъектных картографических сцен с ассоциативной защитой // Моделирование систем: тр. республ. науч. семинара «Методы моделирования». − Казань: ФЭН, 2013. – Вып. 5. – С. 100–110.

4. Гибадуллин Р.Ф., Пыстогов С.В. Подходы к организации системы управления защищенными картографическими базами данных // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2013): материалы 13-й Всерос. конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2013. – С. 77–81.

А.Г. Савельев

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

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ ДЛЯ АНАЛИЗА БИНАРНЫХ СЦЕН

Рассматриваются неграфические вычисления на графических процессорах. Для исследования использовался разработанный программный комплекс. Получены сравнительные оценки времени идентификации кадра на CPU и GPU в зависимости от параметров распознаваемых объектов.

Введение

Рассматриваемая задача распознавания стилизованных бинарных объектов изображений является обобщением классической процедуры ассоциативного поиска на двумерный случай и формулируется следующим образом [1]. Для данного троичного эталона (троичной матрицы)

$$X^t = \left\| x^t_{pq} \right\|;\ x^t_{pq} \in \{0, 1, –\},\ q = 1\ldots m^t,\ p = 1\ldots n^t;\ t \in \{\overline{1,\lambda}\}$$

(λ – число типов объектов) найти координаты всех покрываемых им объектов двоичного изображения (булевой матрицы)

$$A = \left\| a_{ji} \right\|;\ a_{ji} \in \{0, 1\},\ i = 1\ldots K, j = 1\ldots L;\ K \geq m^t,\ L \geq n^t.$$

Запись x^t_{pq} = (–) означает безразличное (замаскированное) значение соответствующего элемента эталона. Множество замаскированных элементов отображается единичными элементами матрицы масок:

$$M^t = \left\| m^t_{pq} \right\|;\ m^t_{pq} = \begin{cases} 0,\ x^t_{pq} \in \{0,1\}, \\ 1,\ x^t_{pq} \in \{-\}. \end{cases}$$

«Быстрый» алгоритм для последовательной ЭВМ

«Быстрый» алгоритм был предложен для идентификации объектов структурированных двоичных изображений для последовательной ЭВМ с целью повышения производительности нечисловой обработки массивов данных. Не вдаваясь в подробности, важно отметить следующие черты этого алгоритма. Для *k*-строки любого двоичного объекта в памяти ЭВМ хранится своя матрица инциденций (рис. 1). Элемент этой матрицы $\alpha^k_{s,z}$ = 1, если *z*-содержимое *k*-строки анализируемого символа представляет одну из реализаций *k*-строки *s*-эталонного символа. Например, для $[k]_s$ = 01-010-1 имеем реализации *z* = 73, 75, 105, 107.

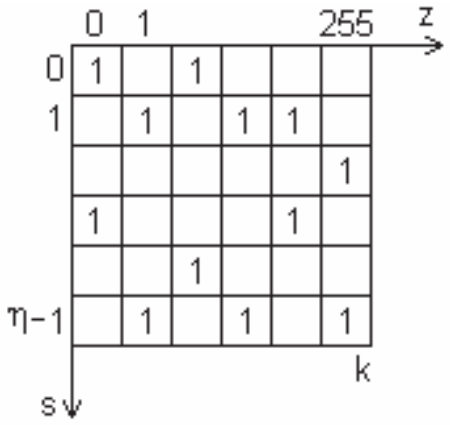


Рис. 1. Матрица *k*-инциденций

В отличие от «стандартного» алгоритма, где распознавание происходит с обращением к матрицам-маскам, здесь информация о дихотомизирующих битах хранится в матрицах инциденций.

Содержимое *k*-анализируемой строки выделяет *z*-столбец *k*-матрицы инциденций. Следующая строка анализируемого символа определяет свой столбец (*k*+1)-матрицы, а конъюнкция выделенных столбцов – номера эталонов, которым может соответствовать анализируемый символ. В шаге *k* = *z* конъюнктивный столбец будет содержать в точности одну единицу (это следует из условия взаимной непокрываемости эталонов), координата которой (*s*) укажет номер символа.

Таким образом, при идентификации объектов матрицы инциденций обладают большей информативностью, чем сами маски, что положительно выделяет «быстрый» алгоритм. Но основной его минус в том, что число и размер матриц существенно возрастает с увеличением числа типов и размеров распознаваемых объектов.

Для хранения матриц инциденций в реализации алгоритма для CPU используется двумерный целочисленный массив M[k][z], где k – это номер матрицы инциденций, а z – столбец матрицы. Таким образом, например, массив типа int может хранить информацию до 15 объектов, занимая в памяти $k * z *$ 4 байт. Распознавание происходит операцией побитового «И» над каждой строкой распознаваемого участка кадра. Если анализируемый участок является распознанным объектом, то результатом этого побитового умножения будет целое число, не равное нулю.

$$R_{ij} = \log_2 S,$$

Здесь S – результат побитового умножения; R – номер распознанного объекта, а i и j – координаты распознанного участка.

Алгоритм для комплекса Host-GPU

Отличие алгоритма для комплекса Host-GPU от алгоритма для последовательной ЭВМ заключается в разделении работ по распознаванию кадра и параллельной обработке участков кадра.

Матрица инциденций, как и прежде, формируется на центральном процессоре, после чего записывается в pinned-память GPU. Кадр делится на 4 части. Два потока последовательно запускают функцию распознавания во время его выполнения, данные асинхронно загружаются и выгружаются по шине PCIe x16.

Функция распознавания использует по одной нити на каждый элемент получающегося массива результатов, при этом нить извлекает все необходимые элементы из глобальной памяти и производит требуемые вычисления. Элемент S_{ij} распознавания получается следующим фрагментом псевдокода:

$$S_{ij} = 2^{\lambda} - 1,$$

$$\text{for k in 0..K} - 1,$$

$$S_{ij} \,\&= \text{M}[k][\text{Кадр}[i][j][k]],$$

где K – высота объекта; M – матрица инциденций; λ – количество объектов.

Нити формируются в двумерные блоки размером 16×16, которые образуют двумерный массив блоков.

После выполнения ядра результаты выгружаются в оперативную память компьютера, и CPU определяет номер распознанного объекта аналогично предыдущему алгоритму.

Результаты исследований

В первом экспериментальном исследовании распознавались объекты с размерами 8×15 и $\lambda = 10$. Идентификация кадра размером 2048×2048 (4194304 объектов) велась по известному набору масок. Всего было проведено 10 испытаний по каждому из алгоритмов. В результате среднее арифметическое время распознавания кадра программным модулем для комплекса Host-GPU равняется ~116,3 мс, а модулем для CPU – ~3750 мс. С помощью CUDA-отладчика Parallel NSight были получены статистические данные работы функции на GPU (таблица).

Статистические данные работы функции на GPU

Функция	Размерность сетки	Размерность блока	Время (μs)
Kernel	{64, 64, 1}	{16, 16, 1}	3,280.896
Kernel	{64, 64, 1}	{16, 16, 1}	3,268.376
Kernel	{64, 64, 1}	{16, 16, 1}	3,276.624
Kernel	{64, 64, 1}	{16, 16, 1}	3,276.760

Регистров на нить	Выделенные варпы на блок	Выделенные регистры на блок
7	8	2048
7	8	2048
7	8	2048
7	8	2048

GPU Issued IPC	GPU Executed IPC	Активность SM	Достигнутая занятость
1,966783349	1,414995914	0,999489116	0,95099632
1,816124252	1,178869326	0,999578923	0,940766708
1,964114387	1,411264947	0,999368926	0,951478569
1,814132215	1,17540564	0,999472017	0,939884818

Во втором исследовании были получены сравнительные оценки быстродействия и выявления зависимости прироста производительности от формы и количества распознаваемых объектов. Для каждого отдельного случая было произведено по 10 испытаний. Идентификация велась на кадре 2048×2048 (4194304 закодированных объектов). В эксперименте варьировалась ширина объектов, высота объектов и их количество.

Эксперимент показал, что изменение размеров объекта существенно влияет на производительность. Увеличение объектов заметно снижает прирост производительности, при этом ключевую роль в снижении играет

ширина объекта. Заметное снижение наблюдается, когда ширина становится больше 8. Это можно объяснить тем, что данные в памяти карточки не выравниваются, и это не позволяет нитям делать объединенные запросы к памяти. Та же самая ситуация наблюдается при увеличении количества объектов. Дело в том, что для большого количества объектов (>31) памяти для хранения, выделенной под тип int, недостаточно, поэтому приходится оперировать с данными типа long (вместо 4 байт – 8), что приводит к резкому снижению производительности.

Проведенные исследования показали, что современные многоядерные GPU – потенциально значимый ресурс повышения производительности систем, который позволяет достичь прироста скорости более чем на порядок.

Выводы

Наибольшая производительность достигается при обработке больших объемов данных.

Пересылка данных на видеокарту и обратно – узкое место в работе приложений на графических процессорах.

Следует избегать длинных циклов и большой глубины вложений условий.

Заключение

При грамотном использовании возможностей видеокарты можно получить довольно высокую производительность в задачах обработки изображений. Использование GPU в задачах, с которыми может быстро справиться CPU, нецелесообразно. Наличие в алгоритме большого количества сложных вычислений может стать причиной выбора графических процессоров в качестве вычислительной мощности.

Список литературы

1. Райхлин В.А. Анализ производительности процессорных матриц при распознавании двоичных образов //Автометрия. РАН. Сибирское отделение. – 1996. – № 5. – С. 97–103.

2. Patterson D., Microprocessors in 2020 //Scientific American, 150th Anniversary Edition. – 2009. – No. 3. – P. 62–67.

3. Сандерс Д., Кэндрот Э. Технология CUDA в примерах. Введение в программирование графических процессоров / пер. с англ. А.А. Слинкина – М.: ДМК Пресс, 2011. – 232 с.

М.А. Савин, Ю.А. Селянинов, К.С. Галягин

Пермский национальный исследовательский политехнический университет

ПОСТРОЕНИЕ СЕТОК БОЛЬШОЙ РАЗМЕРНОСТИ СРЕДСТВАМИ STAR-CCM+

Представлены основные замечания по поводу использования автоматического сеточного генератора пакета STAR-CCM+, а также оригинальная авторская методика, позволяющая построить качественную конечно-объемную сетку больших размеров в автоматическом режиме.

Эффективность применения вычислительного инженерного пакета STAR-CCM+ для построения компьютерных моделей технологических устройств и процессов определяется следующими аспектами.

Во-первых, представленный пакет является конечно-объемным, что само по себе дает гораздо более широкие возможности в выборе размерности задачи. Идеология конечных объемов подразумевает конечно-разностную аппроксимацию всех разрешающих соотношений в отличие от конечных элементов, где используются вариационные принципы, поэтому нет необходимости в хранении матриц жесткости при расчете.

Во-вторых, клиент-серверная архитектура STAR-CCM+ позволяет проводить расчеты на сколь угодно больших сетках (размерностью 1 млрд ячеек и более). STAR-CCM+ осуществляет распараллеливание по технологии MPI и при соответствующей настройке может захватить любое доступное число узлов.

Очевидно, что чем выше размерность задачи, тем труднее работать с ней в ручном режиме, а также сложнее контролировать требуемые параметры сеток.

Автоматический сеточный генератор STAR-CCM+ позволяет построить довольно качественные сетки на не слишком сложных областях. Построение сеток на сложной геометрии обычно вызывает чрезмерное и иногда неравномерное увеличение количества ячеек. Эта проблема не исчезает и при использовании контрольных объемов, позволяющих сгустить сетки в строго определенных местах исследуемой геометрии изделия. И даже при включении анизотропии размера ячеек в контрольном объеме число ячеек зачастую оказывается избыточным.

Довольно трудной задачей является построение сеток на поверхностях, смыкающихся под острым углом. В этом случае технология автоматического исправления при довольно крупном размере ячеек сильно искажает расчетную геометрию, что недопустимо.

Для минимизации либо полного ухода от этих и других проблем была разработана методика автоматического построения сеток на основе java-макросов. Данная методика создана в рамках комплекса работ по компьютерному моделированию поведения волоконно-оптического гироскопа (ВОГ) в условиях внешних тепловых и механических воздействий. При разработке расчетной модели ВОГ возникла необходимость детального описания регулярно микронеоднородной среды датчика чувствительности гироскопа в виде периодически повторяющихся сечений кварцевого световода, уложенного на карбоновом каркасе оптического блока. Диаметр оптического волокна в рассматриваемой задаче составляет 162 мкм. Стоит отметить, что волокно типа Panda моделируется с соблюдением особенностей внутреннего строения, а именно: осуществляется построение сетки на обоих защитно-упрочняющих покрытиях, кварцевой оболочке и нагрузочных стержнях (рис. 1).

При этом использовалась технология Directed Mesh STAR-CCM+, позволяющая создать концентрически ориентированные ячейки, а также требуемое количество ячеек по длине окружности. Генерация объемной сетки с помощью вышеупомянутой технологии позволяет задать необходимое число ячеек по толщине модели. Такое качество сетки при столь малом количестве ячеек автоматический генератор не обеспечит. На модель (см. рис. 1) затрачено порядка 2000 ячеек. Использование автоматического генератора не позволяет снизить размерность ниже 4500 ячеек без существенной потери качества.

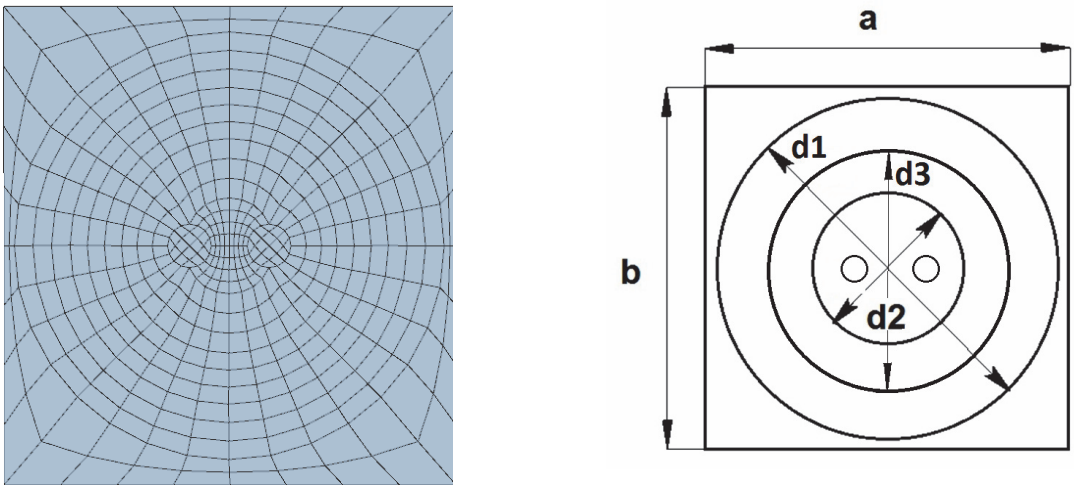


Рис. 1. Дискретизация оптического волокна

Поскольку создается осесимметричная модель с углом в 0,2 градуса, необходимо создать набор радиальных элементов, а затем тиражировать их по осевому направлению.

В этом случае используется специально разработанный макрос. С его помощью сетка собирается по заданному шаблону сколь угодно больших размеров. Стыковка и объединение соответствующих границ происходит автоматически при выполнении макроса. Таким образом, возможно получить сетку размерностью порядка 8 млн ячеек, насчитывающую 3060 элементов (см. рис. 1).

При таком подходе возможно построение различных расчетных сеток из любого набора базовых элементов (рис. 2).

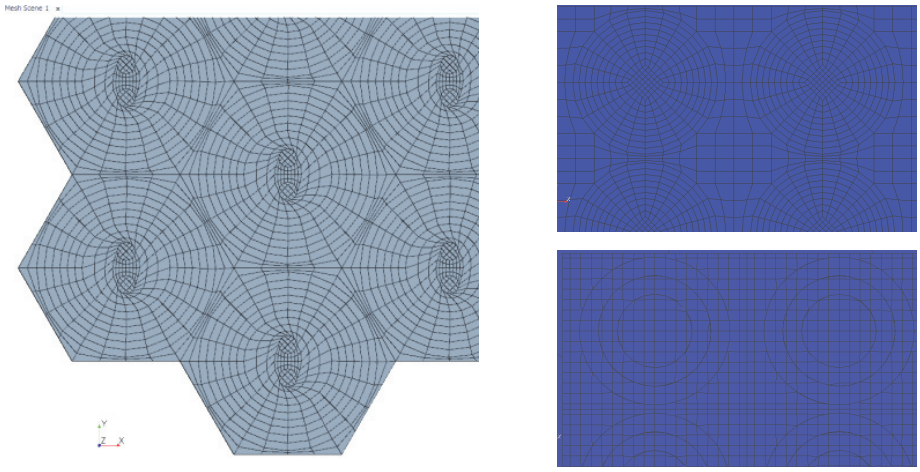


Рис. 2. Варианты сеток

В последних версиях пакета STAR-CCM+ доступно параллельное сеткопостроение, а также распараллелены некоторые операции, связанные с объединением регионов, что значительно уменьшает временные затраты на подготовку расчетной модели. Тем не менее основной проблемой при работе с большими сетками остается сам расчет в смысле вычислительной трудоемкости. При проведении расчетов необходимо оценить минимально достаточное количество ресурсов, требуемых для проведения расчета.

Для рассматриваемой задачи моделирования фиктивных показаний ВОГ размерностью порядка 8 млн ячеек проведена серия калибровочных расчетов, наглядно демонстрирующих необходимое и достаточное число логических процессов и вычислительных узлов (рис. 3).

В данных вычислительных экспериментах было задействовано два расчетных узла с максимальным числом ядер, равным 16 (32 логических ядра с использованием гипертрейдинга). Очевидно, что для текущей задачи оптимальным является 30 логических ядер, хотя можно ограничиться 12–15 без ощутимой потери производительности.

По рекомендациям специалистов компании CD-adapco, чьим продуктом является STAR-CCM+, следует рассчитывать число узлов и ядер так, чтобы на каждое логическое ядро приходилось не менее 200000 ячеек (что

согласуется с результатами наших экспериментов). В противном случае пользователь проигрывает в производительности.

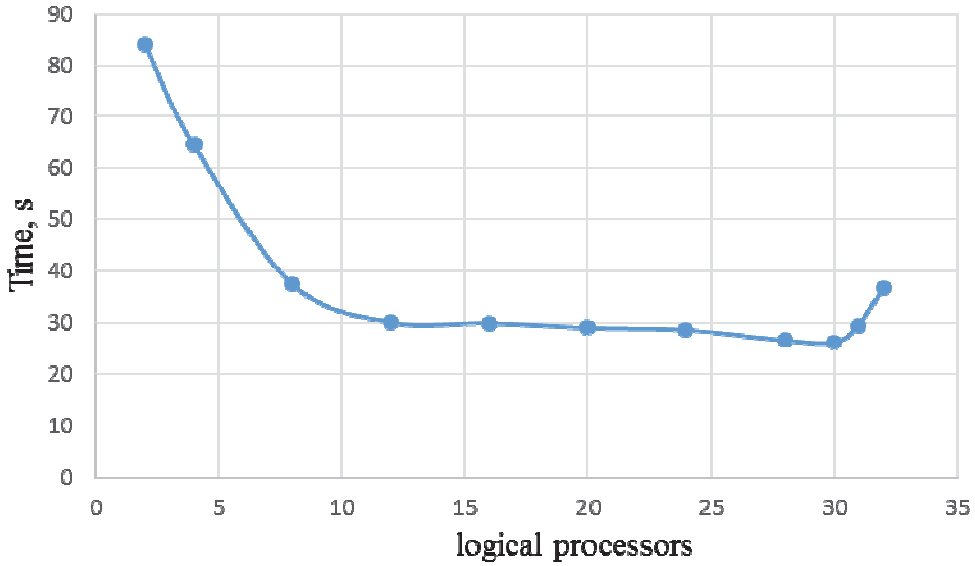


Рис. 3. Динамика затрат вычислительных ресурсов

При работе на кластерах стоит учитывать также время обмена данными между узлами. Этот параметр чрезвычайно важен при использовании витых пар для соединения узлов. Зачастую время обмена может составлять до 77 % времени счета узла при использовании 10Gb Ethernet кабеля и максимально может быть снижено до 38 % на витой паре при использовании FDR IB кабелей. Разумеется, эти цифры еще больше снижаются при подключении узлов по оптоволоконным линиям связи.

Список литературы

1. Документация и тьюториалы пакета STAR-CCM+ [Электронный ресурс]. – URL: http: www.saec.ru, www.cd-adapco.com.

2. Борох П. DELL. Максимизируем производительность STAR-CCM+ на технике DELL [Электронный ресурс]. – URL: http://download.star-russian-conference.ru/star2014/ SRC2014_DELL_Borokh.pdf.

Е.С. Саломатова, Д.Н. Трушников, А.И. Цаплин, В.Я. Беленький

Пермский национальный исследовательский политехнический университет

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ИССЛЕДОВАНИЯ ИСПАРЕНИЯ ПРИ ЭЛЕКТРОННО-ЛУЧЕВОЙ СВАРКЕ

Описывается оригинальная методика косвенного измерения давления паров и температуры в канале проплавления при электронно-лучевой сварке. Эта методика основана на экспериментальном определении концентрации химических элементов в парах над зоной сварки и сопоставлении этих результатов с результатами моделирования. 2D-модель основана на совместном решении уравнений диффузии с учетом тепло- и массопереноса в расплаве. Расчеты производились на вычислительном кластере в пакете прикладного программного обеспечения Comsol Multiphysics с применением технологии параллельных вычислений.

Введение

При воздействии мощного концентрированного электронного пучка на металл формируется глубокий и узкий парогазовый канал [1, 2]. В парогазовом канале проходит интенсивное испарение материала, что может приводить к снижению концентрации некоторых легирующих легкоплавких элементов в расплавленном металле. Поэтому процессы испарения при ЭЛС представляют интерес при разработке теоретических моделей процесса формирования сварного шва [3–4].

Целью данной работы является построение нелинейной теоретической модели, которая описывает неравновесные процессы в жидкой фазе металла при ЭЛС с учетом процессов испарения химических элементов со стенок канала, конденсации элементов сплава на стенки парогазового канала и диффузии элементов в расплаве.

Постановка задачи. Определяющее уравнение и краевые условия

Для построения модели были приняты следующие предположения: форма парогазового канала аппроксимировалась цилиндром; поверхность парогазового канала считалась изотермической, равной температуре окружающей среды; перенос тепла и массы вещества вдоль оси канала считался несущественным; коэффициент диффузии в расплаве принимался зависимым от температуры; истощение расплава легирующими элементами происходило в тонком слое у поверхности парогазового канала; из-за высокого давления в парогазовом канале скорость паров металла на выходе достигала скорости звука.

Таким образом, задача сводилась к двухмерной, а расчетная область содержала только жидкую фазу. Для решения задачи был использован численный метод, позволяющий учесть процессы испарения, конденсации и диффузии элементов в расплаве. Расчетная схема представлена на рис. 1.

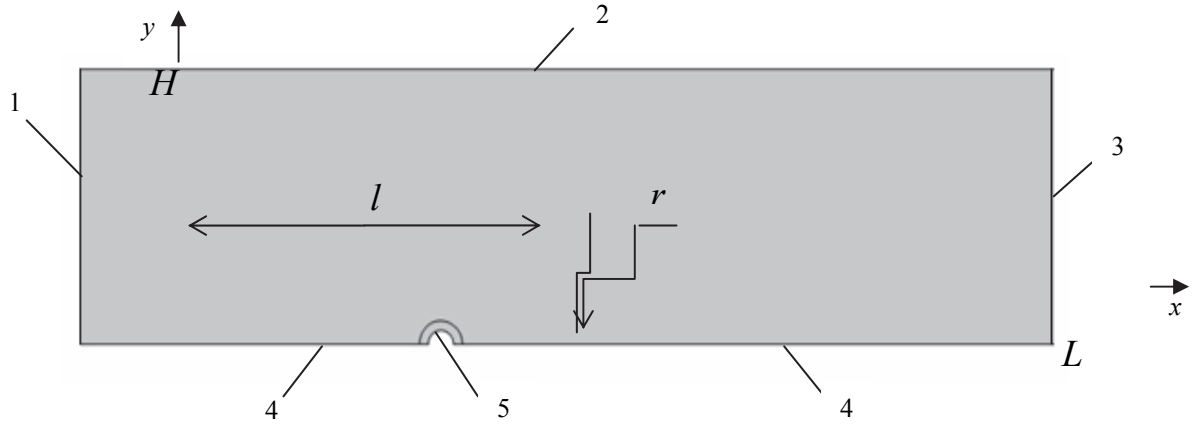


Рис. 1. Расчетная схема: *1* – входная граница;
2 – боковая граница; *3* – выходная граница;
4 – ось симметрии; *5* – парогазовый канал

Система уравнений, описывающая процессы испарения при ЭЛС, включает уравнение переноса импульса (уравнение Навье – Стокса)

$$\left(\vec{U}\cdot\nabla\right)\vec{U} = -\frac{\nabla p}{\rho} + \nu\Delta\vec{U}\,, \tag{1}$$

уравнение переноса энергии

$$\rho c_p \vec{U}\cdot\nabla T = \nabla\cdot\lambda\nabla^2 S_i, \tag{2}$$

уравнение переноса примеси

$$\vec{U}\cdot\nabla S_i = D_i\cdot\nabla^2 S\,, \tag{3}$$

где $\vec{U}$ – поле скоростей расплава; p – давление в расплаве; ρ – плотность расплава; ν – коэффициент кинематической вязкости; $c_\text{р}$ – теплоемкость при постоянном давлении; λ – коэффициент теплопроводности; T – температура на стенках парогазового канала, К;

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2};\ \nabla = \frac{\partial T}{\partial x} + \frac{\partial T}{\partial y};$$

S_i – концентрация i-го компонента в расплаве; D_i – коэффициент диффузии i-го элемента расплава.

В соответствии с работой [5] коэффициент диффузии принимался зависящим от температуры:

$$D_i = D0_i \cdot \exp\left(\frac{Q_i}{8{,}13 \cdot T}\right), \quad (4)$$

где $D0_i$ – амплитудное значение диффузии i-го элемента расплава; Q_i – энергия активации i-го компонента сплава.

Плотность потока массы i-го легирующего элемента на стенке парогазового канала определяется суммой плотностей потоков, обусловленных испарением (Jev_i) и конденсацией (Jc_i): $J_i = Jev_i + Jc_i$.

Результаты численного исследования

Численное исследование процесса испарения химических элементов со стенок канала, конденсации элементов сплава на стенки парогазового канала и диффузии элементов в расплаве проводилось для расчетной области при $H = 1{,}5 \cdot 10^{-3}$ м; $L = 3 \cdot 10^{-3}$ м; $r = 2{,}5 \cdot 10^{-4}$ м; $l = 1{,}6 \cdot 10^{-3}$ м. Исходные данные представлены ниже.

Данные, используемые для расчета при ЭЛС стали 12Х18Н10Т:

Характеристики	Значение
Плотность (кг/м^3)	7810
Теплопроводность (Вт/(м·К))	30,6
Теплоемкость при постоянном давлении (кДж/(кг·К))	0,830
Динамическая вязкость (Па·с)	$1 \cdot 10^{-3}$
Рассчитанный коэффициент конденсации	0,97
Начальная температура на стенках канала проплавления $T_п$ (К)	1800

Для решения задачи тепло- и массопереноса, включающей уравнения переноса импульса, энергии, массы, использовался пакет *Comsol Multiphysics*. Одним из входных параметров модели являлась температура стенок парогазового канала. Для каждого значения температуры путем решения испарения, конденсации, диффузии, тепло- и массопереноса определялась концентрация легирующих элементов в расплаве, давление и концентрация химических элементов в парах. Применение параметрического решателя *Comsol Multiphysics* для кластера, когда решение для каждого значения параметра температуры проводится на отдельном узле, позволило значительно сократить время расчетов.

Для определения температуры парогазовой фазы в канале проплавления результаты расчетов сопоставлялись с данными, полученными при экспериментальных исследованиях (рис. 2).

Значения температуры парогазовой фазы в канале определялись по пересечению расчетных и экспериментальных кривых концентрации элементов и составили: 2320 К (для Fe), 2340 К (для Cr) и 2410 К (для Mn).

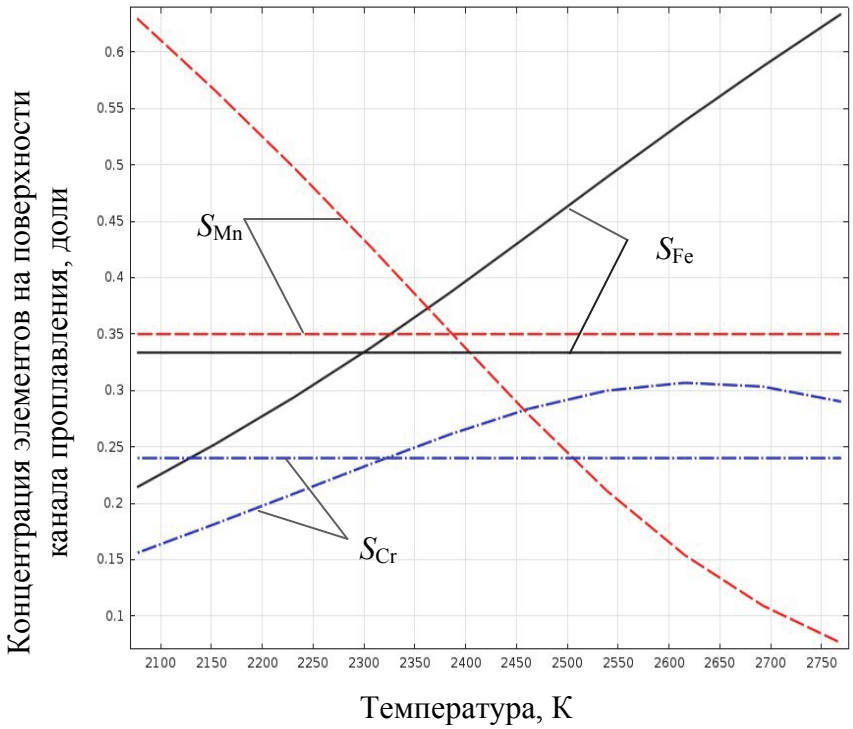


Рис. 2. Зависимость концентрации железа, марганца и хрома на поверхности канала от температуры для стали 12Х18Н10Т: горизонтальными линиями обозначены экспериментальные значения концентраций; кривыми – расчетные

Средняя арифметическая температура канала по этим элементам составляет 2360 К. Давление паров в канале при этой температуре около 2 кПа.

Заключение

На основании проведенных исследований можно сделать следующие выводы:

1. Диффузионные процессы в свариваемом материале при ЭЛС в значительной мере влияют на концентрацию химических элементов в поверхностном слое расплава парогазового канала и, как следствие, в парах над зоной сварки. Результаты численного моделирования с учетом обеднения стенок канала легирующими элементами приводят к меньшим, по сравнению с равновесной моделью, температурам (~2360 К). Указанное различие температур обусловливает значительно меньшие расчетные величины давлений паров в парогазовом канале (2 кПа – при учете диффузионных процессов и 11 кПа – без учета диффузии).

2. Разработанная модель позволяет рассчитать давления, температуру и концентрации химических элементов в парогазовой смеси над зоной сварки при ЭЛС. Модель также позволяет прогнозировать изменение химического состава сварного шва при ЭЛС.

Список литературы

1. Ерофеев В.А., Логвинов Р.В., Нестеренков В.М. Особенности использования эквивалентного источника теплоты при учете деформаций и напряжений в процессе электронно-лучевой сварки // Сварка и диагностика. – 2010. – № 4. – С. 22–26.

2. Язовских В.М. Модель глубокого проплавления при электронно-лучевой сварке // Сварочное производство. – 2005. – № 8. – С. 16–20.

3. Kaplan A.F.H., Norman P., Eriksson I. Analysis of the keyhole and weld pool dynamics by imaging evaluation and photodiode monitoring // Proceedings of LAMP 2009 – The 5th International Congress on Laser Advanced Materials Processing. – 2009. – P. 1–6.

4. Numerical simulation of molten pool dynamics in high power disk laser welding / W.-I. Cho, S.-J. Na, C. Thomy, F. Vollertsen // Journal of Materials Processing Technology. – 2012. – Vol. 212. – P. 262–275.

5. Ono Y., Matsumoto S. Diffusion of Chromium, Manganese, and Nickel in Molten Iron // Materials Transactions. – April 1974. – № 11. – P. 38–39.

Я.Д. Сергеев, Д.Е. Квасов

Калабрийский университет, г. Ренде, Италия
Нижегородский государственный университет им. Н.И. Лобачевского

БЕЗЫЗБЫТОЧНАЯ СТРАТЕГИЯ РАЗБИЕНИЯ В ЗАДАЧАХ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ С ЛИПШИЦЕВЫМИ ПРОИЗВОДНЫМИ

Предлагается подход к решению задач оптимизации многоэкстремальных функций с липшицевыми градиентами на основе безызбыточной стратегии разбиения области поиска и множественных оценок константы Липшица для градиента. Приводятся результаты численного сравнения нового метода с известными алгоритмами глобальной оптимизации на классах тестовых функции GKLS-генератора.

Введение

Рассматриваются задачи принятия оптимальных решений, которые характеризуются многоэкстремальными дифференцируемыми целевыми функциями с липшицевыми градиентами. Предполагается, что каждое вычисление целевой функции и ее градиента («испытание» функции) в точке допустимой области является трудоемкой операцией. Отмеченные особенности отличают системы, вычисление критериев качества которых связано

со сложными и длительными численными расчетами (численным решением сложных систем дифференциальных уравнений, выполнением компьютерного имитационного моделирования и т.п.).

Для решения данных задач предлагается использовать одноточечные геометрические методы глобальной оптимизации (как в последовательном, так и в параллельном вариантах) на основе безызбыточной диагональной стратегии разбиения области поиска и множественных оценок констант Липшица для градиента целевой функции.

Постановка задачи

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

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

$$\| f'(x^1) - f'(x^2) \| \le K \|x^1 - x^2\|,\ x^1, x^2 \in D, \tag{2}$$

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 рассматриваются различные алгоритмы решения задачи (1)–(3) [4, 7, 9, 18, 26, 27]. Они могут быть разделены на несколько групп в зависимости от способа получения оценки константы Липшица K из (2). К первой группе относятся алгоритмы, в которых применяется априорная оценка K [5, 10]. Вторая группа включает алгоритмы, адаптивно оценивающие константу Липшица в ходе поиска (адаптивная глобальная оценка, [1, 2, 11, 13, 25, 27], или адаптивные локальные оценки, [7, 13, 19, 22, 23, 25, 27]). Наконец, к третьей группе принадлежат алгоритмы, в которых оценка константы Липшица выбирается из некоторого множества возможных значений [16, 17]. 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 D на множество подобластей D_i, а теоретическое исследование алгоритмов разбиения может быть проведено в рамках единых схем [4, 7]. Например, применяются сложные стратегии разбиения, основанные на симплексах [4, 20, 21]. Успешно используются разбиения с проведением испытаний функции в вершинах, соответствующих главной диагонали каждого гиперинтервала (диагональные

схемы разбиения, [6, 7, 8, 24]. Широко применяются также экономные одноточечные стратегии разбиения, например разбиение области D на гиперинтервалы с вычислением функции в их центральных точках [4, 5, 15].

Безызбыточная стратегия разбиения

В работе [7] описана безызбыточная диагональная стратегия разбиения, преодолевающая недостатки традиционных диагональных схем, рассмотрены ее основные свойства и обсуждены детали ее программной реализации. В частности, отмечено, что преимущества использования данной схемы в диагональных алгоритмах по сравнению с традиционными становятся более заметными при увеличении размерности задачи. Тем самым открываются интересные перспективы в создании новых быстрых алгоритмов глобальной оптимизации. Во-первых, одномерные методы глобальной оптимизации могут быть существенно более эффективно (по сравнению с традиционными диагональными подходами) обобщены на многомерный случай при помощи безызбыточной диагональной схемы. Во-вторых, методы глобальной оптимизации на основе данной стратегии разбиения гиперинтервалов могут быть успешно распараллелены по схеме [9, 14, 27], что дает дополнительные возможности для ускорения поиска.

Первоначально безызбыточная стратегия была разработана для использования в рамках диагонального подхода, с испытанием функции в двух точках каждого гиперинтервала. Однако, как было показано в [7], она может быть полезна и при работе с одноточечными методами. Процесс разбиения при помощи этой стратегии проиллюстрирован на рисунке. На первой итерации некоего одноточечного метода целевая функция $f(x)$ и ее градиент $f'(x)$ вычисляются в точке b (или же в точке a). Начальный гиперинтервал делится на три равные части, так что каждый из трех новых подынтервалов содержит только одну вершину, в которой проводится испытание (второе испытание проводится в точке *2*, рисунок, *а*). Предположим, что подынтервал, выделенный светло-серым цветом на рисунке, *а*, выбирается для разбиения в ходе третьей итерации, результат которой показан на рисунке, *б*. Испытание функции производится в точке *3*. Результат четвертой итерации представлен на рисунке, *в*. Видно, что точка, в которой следует провести испытание на этой итерации, совпадает с точкой, в которой уже выполнено испытание на предыдущей итерации. Следовательно, метод производит разбиение «бесплатно», поскольку требуемые значения функций могут быть просто считаны из поисковой базы данных [7, 8]. Наконец, на рисунке, *г* показана ситуация, сложившаяся после проведения 11 итераций. Видно, что метод сгенерировал 21 гиперинтервал, вычислив при

этом $f(x)$ и $f'(x)$ всего лишь в семи точках. Таким образом, благодаря одноточечной безызбыточной стратегии разбиения исходная область поиска исследуется методом достаточно детально и с минимальными вычислительными затратами.

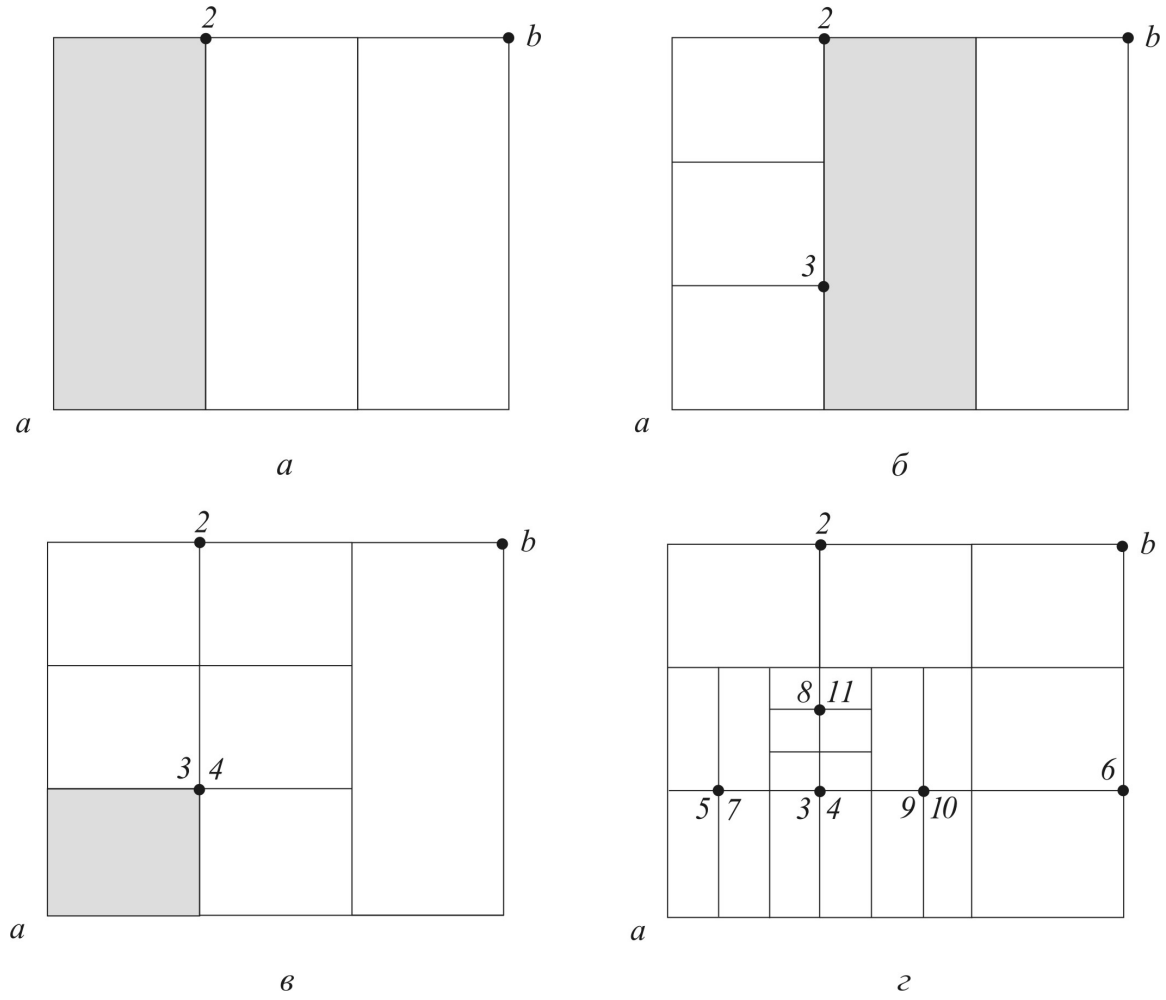


Рис. Одноточечная безызбыточная стратегия разбиения

Геометрический метод с множественными оценками константы Липшица

Геометрический подход к решению задач глобальной оптимизации [4, 7, 27] предполагает использование вспомогательных функций для аппроксимации $f(x)$ из (1). В случае задач с липшицевыми производными этот подход стал активно развиваться с начала 90-х гг. прошлого века, когда независимо появились работы [1, 10]. Использование геометрических методов с множественными оценками константы Липшица для градиента стало возможным благодаря работе [16], в которой идеи метода *DIRECT* [15] для недифференцируемых задач были успешно распространены на случай задач с липшицевыми производными.

Как показано в работе [17], обобщение одномерного метода [16] на многомерный класс задач (1)–(3) при помощи одноточечной безызбыточной схемы разбиения позволяет существенно ускорить процедуру поиска глобального минимума (относительно проведенных трудоемких испытаний целевой функции).

В таблице приведены результаты сравнения метода из [17] (обозначим его *MultK*) на основе безызбыточной стратегии разбиения и множественных оценок константы Липшица K из (2) с популярными методами *DIRECT* и *DIRECTl* [15, 7] (оба метода разработаны для решения задач липшицевой глобальной оптимизации с использованием множественных оценок константы Липшица для целевой функции, без вычисления градиента функции).

Сравнение методов DIRECT, DIRECT*l* и Mult*K* на четырех GKLS-классах

n	ε	Число испытаний			Коэффициент ускорения метода *MultK* по отношению к	
		DIRECT	*DIRECTl*	*MultK*	*DIRECT*	*DIRECTl*
2	10^{-4}	3201	3414	1075	2,98	3,18
3	10^{-6}	>1 000 000	29 233	2352	>425,17	12,43
4	10^{-6}	>1 000 000	287 857	20 866	>47,92	13,80
5	10^{-7}	>1 000 000	>1 000 000	88 459	>11,30	>11,30

Сравнение проведено на четырех дифференцируемых тестовых классах *GKLS*-генератора [12] размерности n = 2, 3, 4 и 5. Каждый класс содержит 100 тестовых функций, которые генерируются (вместе с частными производными) путем переопределения выпуклой квадратичной функции (параболоида) в точках, соответствующих точкам локальных минимумов конструируемой функции, при помощи функций полиномиального типа третьей степени. Кроме размерности задачи необходимые пользовательские параметры *GKLS*-генератора заданы следующим образом: (*i*) – число локальных минимумов m = 10; (*ii*) – значение глобального минимума f^* = –1; (*iii*) – радиус области притяжения точки глобального минимума ρ^* = 0,20 (для *GKLS*-классов размерности n = 3, 4 и 5) и ρ^* = 0,10 (при n = 2); (*iv*) – расстояние между точкой глобального минимума и вершиной параболоида d^* = 0,90 (для *GKLS*-классов размерности n = 2, 3 и 4) и d^* = 0,66 (при n = 2). Остальные необходимые параметры (такие, как, например, расположение точек локальных минимумов, размеры их областей притяжения, значения функции в точках локальных минимумов) задаются случайным образом (при задании одного и того же набора пользовательских параметров генератор производит одни и те же функ-

ции, обеспечивая тем самым повторяемость численных экспериментов на тестовых классах *GKLS*-генератора).

Каждая из 400 тестовых задач считается решенной (при ограничении в 1 000 000 возможных испытаний целевой функции), если сравниваемый метод генерирует точку испытания функции в ε-окрестности известной точки глобального минимума (относительная точность поиска ε указана во втором столбце таблицы). Как видно из таблицы, в которой для каждого *GKLS*-класса указано максимальное число испытаний для решения всех 100 задач класса, метод *MultK* решает все задачи за значительно меньшее число испытаний функции по сравнению с двумя другими алгоритмами (коэффициенты ускорения указаны в последних двух столбцах таблицы).

Список литературы

1. Гергель В.П. Алгоритм глобального поиска с использованием производных // Динамика систем: межвуз. тематич. сб. науч. тр. / под ред. Ю.И. Неймарка. – Горький: Изд-во ГГУ, 1992. – С. 161–178.

2. Гергель В.П. Об одном способе учета значений производных при минимизации многоэкстремальных функций // Журнал вычислительной математики и математической физики. – 1996. – Т. 36, № 6. – С. 51–67.

3. Городецкий С.Ю. Несколько подходов к обобщению метода DIRECT на задачи с функциональными ограничениями // Вестник Нижегород. ун-та им. Н.И. Лобачевского. – 2013. – № 6. – С. 189–215.

4. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. – Н. Новгород: Изд-во ННГУ, 2007. – 257 с.

5. Евтушенко Ю.Г., Посыпкин М.А. Применение метода неравномерных покрытий для глобальной оптимизации частично целочисленных нелинейных задач // Журнал вычислительной математики и математической физики. – 2011. – Т. 51, № 8. – С. 1376–1389.

6. Квасов Д.Е., Сергеев Я.Д. Многомерный алгоритм глобальной оптимизации на основе адаптивных диагональных кривых // Журнал вычислительной математики и математической физики. – 2003. – Т. 43, № 1. – С. 42–59.

7. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008. – 352 с.

8. 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.

9. Параллельные вычисления в задачах глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов. – М.: Изд-во Моск. ун-та, 2013. – 280 с.

10. Breiman L., Cutler A. A deterministic algorithm for global optimization // Math. Program. – 1993. – Vol. 58, № 1–3. – P. 179–199.

11. Fowkes J.M., Gould N.I.M., Farmer C.L. A branch and bound algorithm for the global optimization of Hessian Lipschitz continuous functions // J. Global Optim. – 2013. – Vol. 56, № 4. – P. 1791–1815.

12. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization / M. Gaviano, D.E. Kvasov, D. Lera, Ya.D. Sergeyev // ACM Trans. Math. Softw. – 2003. – Vol. 29, № 4. – P. 469–480.

13. Gergel V.P. A global optimization algorithm for multivariate function with Lipschitzian first derivatives // J. Global Optim. – 1997. – Vol. 10, № 3. – P. 257–281.

14. Gergel V.P., Sergeyev Ya.D. Sequential and parallel algorithms for global minimizing functions with Lipschitzian derivatives // Comput. Math. Appl. – 1999. – Vol. 37, № 4–5. – P. 163–179.

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. A univariate global search working with a set of Lipschitz constants for the first derivative // Optim. Lett. – 2009. – Vol. 3, № 2. – P. 303–318.

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. Kvasov D.E., Sergeyev Ya.D. Deterministic approaches for solving practical black-box global optimization problems // Adv. Engineering Softw. – 2014. – В печати.

19. 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, no. 1. – P. 508–529.

20. Paulavičius R., Žilinskas J. Simplicial Global Optimization. – N.Y.: Springer, 2013. – 145 p.

21. Globally-biased DISIMPL algorithm for expensive global optimization / R. Paulavičius, Ya.D. Sergeyev, D.E. Kvasov, J. Žilinskas // J. Global Optim. – 2014. – Vol. 59, № 2–3. – P. 545–567.

22. Sergeyev Ya.D. Global one-dimensional optimization using smooth auxiliary functions // Math. Program. – 1998. – Vol. 81, № 1. – P. 127–146.

23. Sergeyev Ya.D. Multidimensional global optimization using the first derivatives // Comput. Math. Math. Phys. – 1999. – Vol. 39, № 5. – P. 711–720.

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 A deterministic global optimization using smooth diagonal auxiliary functions // Communications in Nonlinear Science and Numerical Simulation. – 2014. – В печати. DOI: 10.1016/j.cnsns.2014.08.026

26. Sergeyev Ya.D., Strongin R.G., Lera D. Introduction to Global Optimization Exploiting Space-Filling Curves. – N.Y.: Springer, 2013. – 133 p.

27. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms. – Dordrecht: Kluwer Academic Publishers, 2000. – 704 p.

А.А. Сиднев, В.П. Гергель

Нижегородский государственный университет им. Н.И. Лобачевского

ПРИМЕНЕНИЕ МАКРОМОДУЛЬНОГО ПОДХОДА РАЗРАБОТКИ ПРОГРАММ

Рассматривается макромодульный подход разработки программ, который позволяет снизить сложность перехода на новые библиотеки и выбора оптимальной библиотеки. Приводится краткое описание макромодульного подхода, методики его применения и основные полученные результаты. Накладные расходы на использование макромодульного подхода на тестовых вычислительных системах не превосходят 1 мс. Автоматический выбор наиболее эффективной реализации с использованием планировщика осуществляется более чем в 80 % случаев, а потери времени от ошибочного выбора не превосходят 6 %.

Введение

Для снижения сложности разработки программного обеспечения (ПО) общепринят модульный подход. Полученные модули могут быть использованы повторно, будучи оформленными в виде отдельных библиотек. В то же время постоянное развитие вычислительных систем [1], операционных систем и системного программного обеспечения приводит к широкому разнообразию вычислительных библиотек.

Один из нерешенных вопросов в модульном подходе разработки программ – отсутствие стандартов на интерфейсы модулей. Разработчик библиотеки сам определяет удобные ему структуры хранения данных и интерфейсы функций, которые их обрабатывают. В результате каждая библиотека получается уникальной. Наличие большого количества библиотек приводит к необходимости выбора из них одной или нескольких наиболее

подходящих. Поддержка нескольких библиотек усложняет структуру разрабатываемого проекта, а значит, трудозатраты на его разработку увеличиваются. Задача выбора библиотеки часто является многокритериальной, в связи с чем приходится идти на компромисс. Кроме того, в процессе разработки могут возникнуть задачи, которые нельзя решить, используя текущую библиотеку. При переходе к использованию новой библиотеки разработчику придется столкнуться с необходимостью выполнить модификацию разработанных структур данных и функций под те, которые используются в библиотеке. Итак, основной недостаток классической модульной разработки программного обеспечения – это сильная зависимость от конкретной реализации библиотеки. Этот недостаток порождает три актуальные проблемы, которые тесно связаны и часто возникают одновременно:

1) выбор наиболее подходящей библиотеки под текущие задачи проекта;

2) поддержка нескольких библиотек;

3) переход на новую библиотеку.

Одно из лучших решений проблемы перехода – это разработка стандартного интерфейса для библиотек, решающих задачи одного класса. Ярким примером является стандарт *MPI* [2]. У такого решения есть важный недостаток, который ограничивает его применимость, – разработка стандарта требует существенных усилий большой группы людей и занимает длительное время. Сложность перевода приложения на новую библиотеку в первую очередь определяется качеством его проектирования и тем, была ли заложена при этом возможность перехода. Наиболее удачные решения, используемые при проектировании приложений, оформились в виде шаблонов проектирования [3]. Поддержка нескольких библиотек реализуется за счет использования средств автоматизации сборки [4]. В работе [5] рассматривается шаблонный метод трансформации программ, который заключается в формализации структуры и поведения исходной и целевой библиотек с помощью специального языка. На основе этих описаний выполняется трансформация программы под новую библиотеку. Кроме того, широкое развитие получили средства автоматизации программного реинжиниринга, которые позволяют выполнять автоматическую трансформацию программ: языки *RASCAL* [6], *TXL* [7], *Stratego* [8], инструменты *DMS* [9].

Применение предлагаемых подходов требует специальных навыков, а по части рассмотренных направлений ведутся дальнейшие исследования. В данной работе рассматривается решение, объединяющее в себе часть рассмотренных подходов, что позволяет устранить их недостатки и снизить остроту перечисленных выше проблем. Данная работа является развитием работы [10].

Макромодульная разработка программ

Центральным элементом макромодульного подхода является наличие стандартного интерфейса для реализаций библиотек. Это обеспечивает как бинарную совместимость библиотек, так и единообразие программного кода.

Обеспечить реализации существующих библиотек, решающих задачи одного класса, единым интерфейсом (даже при наличии стандарта) – тяжелая задача. Для решения этой проблемы автором предлагается разрабатывать программы-переходники (адаптеры) для каждой библиотеки, которые обеспечат «стыковку» стандартных интерфейсов и интерфейсов, используемых в библиотеке.

Для подготовки вызова модуля программной библиотеки в соответствии со стандартным интерфейсом необходимо выполнить описание модели вычислений. Чтобы избежать зависимости описания модели вычислений от конкретного языка программирования, оно выполняется на специальном макроязыке.

Для выбора наиболее эффективной реализации для текущей программно-аппаратной платформы среда исполнения макромодульных программ должна содержать планировщик, определяющий лучшие реализации модулей. Выбор может осуществляться статически во время сборки программы или динамически во время ее выполнения. Последнее позволяет ориентироваться при разработке на более широкий круг программно-аппаратных платформ, так как для каждой из них выбирается лучшая из доступных реализаций.

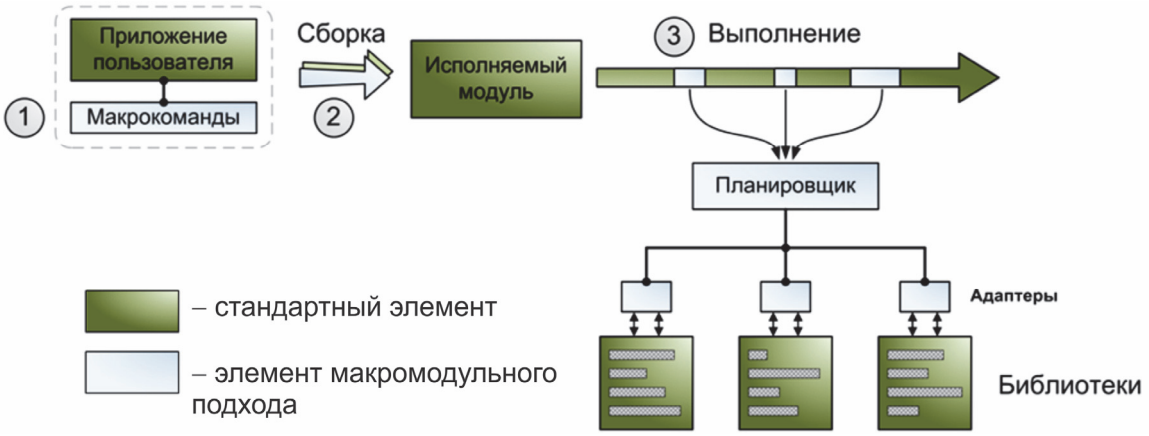


Рис. 1. Общая схема макромодульного подхода: цифрами указано соответствие элементов макромодульного подхода и этапов разработки и выполнения макропрограммы

Макромодульный подход ориентирован на практическую разработку программного обеспечения, поэтому он содержит как методологию по-

строения программного обеспечения, так и программную среду, которая включает следующие элементы (рис. 1):

1. Макроязык, который позволяет описывать в приложении пользователя использующиеся алгоритмы и структуры данных.

2. Макропроцессор, который выполняет обработку исходных кодов программы.

3. Планировщик, выполняющий выбор библиотеки.

4. Архитектуру среды времени исполнения, которая определяет схему взаимодействия компонент времени исполнения.

5. Архитектуру адаптеров и требования к их реализации.

Применение макромодульного подхода

Макромодульный подход предлагает готовые компоненты построения программного обеспечения: язык макроописаний, планировщик, средства сборки и выполнения приложений. Однако для разработки макропрограмм требуется предпринять ряд действий, специфичных для разных уровней пользователей (рис. 2):

1) прикладной программист;
2) системный администратор;
3) инженер по сопровождению библиотек.

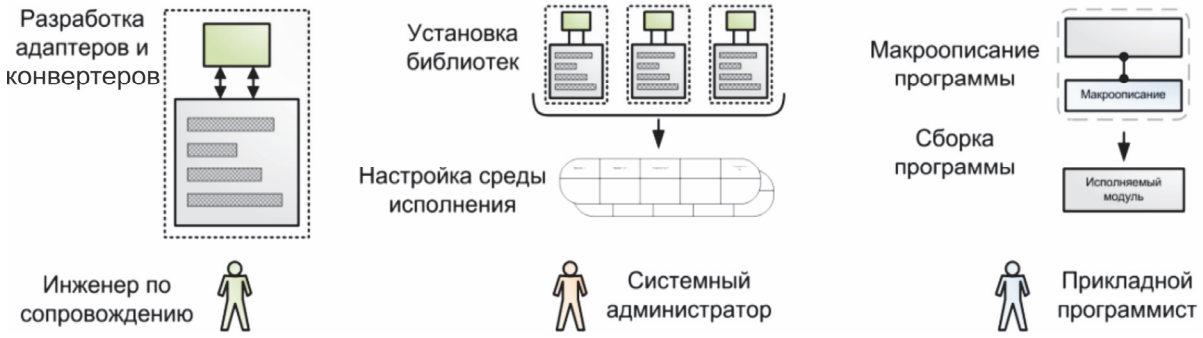


Рис. 2. Детализация действий пользователей макромодульного подхода

Инженер по сопровождению библиотеки выполняет реализацию адаптеров в соответствии с требованиями к их реализации, освобождая основных разработчиков библиотеки от данной работы. В том случае, если в библиотеке используются специфичные структуры данных, то для них потребуется реализация конвертеров. Разработанные адаптеры и конвертеры поставляются вместе с библиотекой. На практике эту роль может выполнять разработчик библиотеки.

Системный администратор выполняет установку библиотек и настройку среды исполнения макропрограмм на конкретной вычислительной системе.

Программист выполняет макроописание прикладной программы на макроязыке, а также сборку и запуск макропрограммы. При необходимости он может выбрать библиотеку и реализацию, которую необходимо использовать, иначе она определяется автоматически во время работы макропрограммы. Каждый независимый вычислительный участок программы описывается отдельно, формируя макрокоманду. Пример макрокоманды представлен на рис. 3.

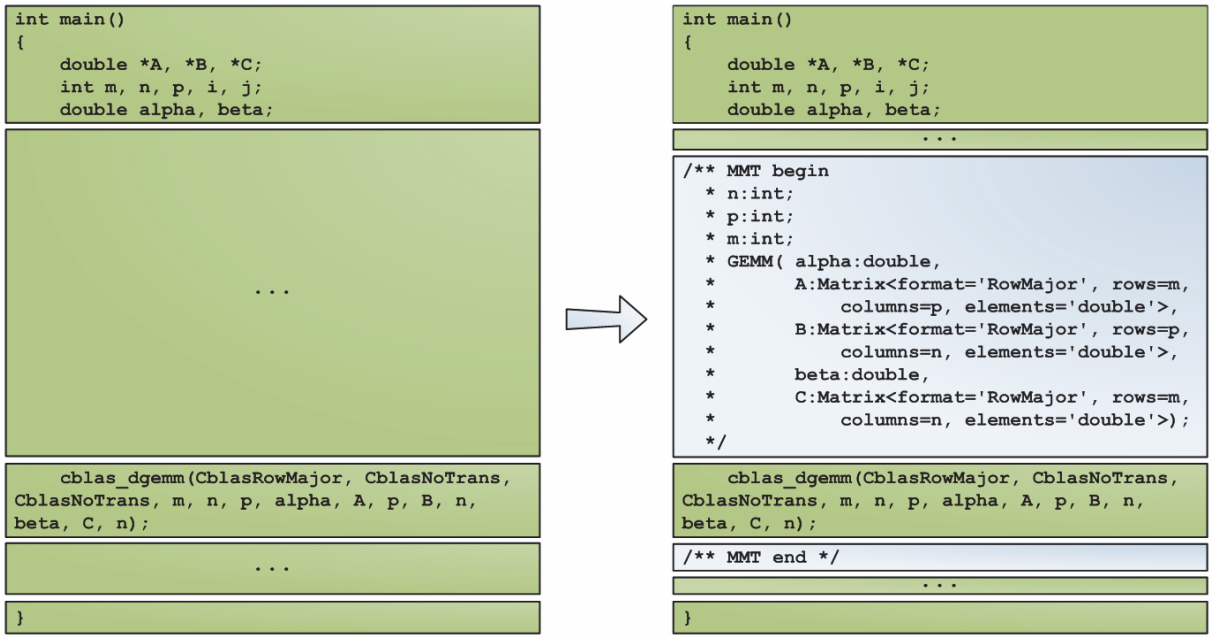


Рис. 3. Макрокоманда для умножения матриц (GEMM)

Каждая макрокоманда применяется к блоку кода программы и задается в виде аннотации. Аннотация оформляется как комментарий языка программирования. Макрокоманда разделена на два блока. В первом блоке содержится описание вычислительной модели на макроязыке. Блок начинается со сточки комментария /** *MMT begin* и заканчивается */. Далее следует программный код, которому соответствует макрокоманда. В конце этого участка кода должен содержаться комментарий /** *MMT end* */, который является признаком конца макрокоманды.

Результаты

Основными программными компонентами макромодульного подхода разработки программ являются:

– макропроцессор, выполняющий обработку программных кодов, которые содержат макрокоманды;

– модуль интеграции в среду разработки *Microsoft Visual Studio* 2012, который обеспечивает удобную сборку макропрограмм;

– средства исполнения макромодульных программ (планировщик, адаптеры и конвертеры).

Планировщик выполняет выбор наиболее эффективной реализации алгоритма среди множества доступных в системе. Результаты экспериментов на 84 вычислительных системах, которые были построены на процессорах разных поколений (от *Pentium 4* до современных на базе ядра *Haswell*), разных производителей (*Intel*, *AMD*), разного назначения (мобильные, настольные, серверные) и, как следствие, существенно отличающейся производительности, демонстрируют высокую эффективность работы планировщика. Выбор наиболее эффективной реализации алгоритма осуществляется более чем в 80 % случаев, а потери времени от ошибочного выбора не превосходят 6 %.

Накладные расходы на использование макромодульного подхода на тестовых вычислительных системах не превосходят 1 мс. Это позволяет без существенной потери эффективности решать задачи средней и большой размерности. При этом эффективного выполнения задач малой размерности можно достигнуть за счет программной оптимизации средств исполнения макромодульных программ и устранения зависимости от внешних сервисов.

Для апробации рассмотренного подхода разработаны примеры с использованием функций *BLAS* и *FFTW3*. Реализованы адаптеры для библиотек *MKL*, *OpenBLAS*, *cuBLAS* и *FFTW*. Полученные примеры могут использовать требуемую библиотеку без модификаций исходного кода. Разработан унифицированный интерфейс для переупорядочивателей при решении СЛАУ. Выполнена адаптация под этот интерфейс существующих реализаций с использование макромодульного подхода: *METIS* [11], O-*MATRUZ* [12], *MORSY* [13].

Для оценки эффективности использования макромодульного подхода выполнено исследование трудозатрат при разработке и модификации ПО на примере задачи умножения матриц. Результаты будут представлены во время доклада на конференции.

Заключение

На данный момент во многих предметных областях существует множество библиотек с реализацией различных алгоритмов. Отсутствие стандартов на интерфейсы этих библиотек порождает ряд проблем, связанных с поддержкой нескольких библиотек, миграцией на новые библиотеки и выбором наиболее подходящей под задачи проекта. Предложенный в работе макромодульный подход разработки программ позволяет снизить остро-

ту перечисленных проблем и сократить трудозатраты на разработку программного обеспечения.

Выполнена реализация всех программных компонент, необходимых для сборки и исполнения макромодульных программ. Накладные расходы на использование макромодульного подхода на тестовых вычислительных системах не превосходят 1 мс. Автоматический выбор наиболее эффективной реализации с использованием планировщика осуществляется более чем в 80 % случаев, а потери времени от ошибочного выбора не превосходят 6 %.

Работа поддержана грантом (соглашение от 27 августа 2013 г. № 02.B.49.21.0003 между МОН РФ и ННГУ).

Список литературы

1. TOP 500 [Электронный ресурс]. – URL: http://www.top500.org/lists (дата обращения: 30.09.2014).

2. MPI Forum [Электронный ресурс]. – URL: http://www.mpi-forum.org (дата обращения: 30.09.2014).

3. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПб: Питер, 2007. – 368 с.

4. Махоткин А. GNU Autotools – инфраструктура для сборки программ: Automake [Электронный ресурс]. – URL: http://squadette.ru/autotools-ru/article1.html (дата обращения: 30.09.2014).

5. Ицыксон В., Зозуля А. Автоматизированная трансформация программ при миграции на новые библиотеки // Программная инженерия. – 2012. – № 6. – С. 8–14.

6. Klint P., Storm T., Vinju J. RASCAL: A Domain Specific Language for Source Code Analysis and Manipulation // Proc. of Ninth IEEE International Working Conference on Source Code Analysis and Manipulation. – 2009. – P. 168–177.

7. Cordy J.R. The TXL source transformation language // Science of Computer Programming. – 2006. – № 61(3). – P. 190–210.

8. Program Transformation with Scoped Dynamic Rewrite Rules / M. Bravenboer, A. Dam, K. Olmos, V. Eelco. Technical Report UU-CS-2005-005 / Department of Information and Computing Sciences. – 2005.

9. DMS Toolkit [Электронный ресурс]. – URL: http://www.semdesigns.com/products/DMS/DMSToolkit.html (дата обращения: 30.09.2014).

10. Гергель В.П., Сиднев А.А. Методы и программные средства макромодульной разработки программ // Вестник Нижегород. ун-та им. Н.И. Лобачевского. – 2012. – № 5(2). – С. 294–300.

11. Metis K.G. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.0. Technical report / University of Minnecota, Department of Computer Science and Engeneering. – 2011.

12. Старостин Н.В., Филимонов А.В. Разработка и реализация параллельного многоуровневого алгоритма равномерного разбиения нераспределенного графа // Сб. тр. НРС. – 2013. – С. 243–248.

13. Pirova A., Meyerov I. MORSy – a new tool for sparse matrix reordering // An International Conference on Engineering and Applied Sciences Optimization. – Kos Island, Greece, 4–6 June 2014. – P. 1952–1964.

А.М. Сипатов[1], В.Я. Модорский[2], А.В. Козлова[2], Л.Н. Бутымова[2]

[1] ОАО «Авиадвигатель», г. Пермь

[2] Пермский национальный исследовательский политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ГАЗОДИНАМИЧЕСКИХ ПРОЦЕССОВ В МОДЕЛЬНОМ ВЕНТИЛЯТОРЕ С ПРИМЕНЕНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ СИСТЕМ

Для оценки возможностей пакета *ANSYS CFX* по расчету акустических характеристик вентилятора исследованы нестационарные поля давления, полученные во входном канале перед вентилятором. В ходе вычислительных экспериментов получены мгновенные распределения осевой скорости в сечениях расчетной модели; зависимость амплитуды давления на частоте следования лопаток от осевого расстояния до вентилятора.

Тональный шум в вентиляторной ступени генерируется нестационарным взаимодействием вращающихся вязких следов, которые возникают при обтекании лопаток вентилятора с неподвижными лопатками спрямляющего аппарата. Для реализации мероприятий по уменьшению шума необходим комплекс систем инженерного анализа, применимых для решения задачи о генерации тонального шума при взаимодействии ротора и статора. Среди этих систем можно выделить две группы: программы моделирования генерации тонального шума и системы инженерного анализа для расчета его распространения в каналах турбомашин.

При решении задачи о генерации шума в качестве математической модели реализована трехмерная нелинейная модель. Данная модель основана на системе уравнений Эйлера или Навье – Стокса, так как процессы являются нелинейными и трехмерными. В качестве системы инженерного анализа для решения такой задачи использована система *ANSYS CFX* [1].

Для отработки методики в данной области выбрана модельная задача о шуме туннельного вентилятора ВН-2, поэтому объект исследования представляет собой небольшой вентилятор, установленный в цилиндрической трубе. За вентилятором крестом расположены четыре силовые стойки прямоугольного сечения. Диаметр периферии вентилятора равен 120 мм, диаметр втулки вентилятора равен 64 мм, высота лопаток и стоек – 28 мм, диаметр трубы – 150 мм. Геометрическая модель исследуемого объекта показана на рис. 1. Результаты экспериментальных исследований вентилятора представлены в работе [2].

Численные эксперименты проведены в Пермском национальном исследовательском политехническом университете с использованием вычислительных мощностей Центра высокопроизводительных вычислительных систем. Отличительными особенностями высокопроизводительного вычислительного комплекса (ВВК) являются процессоры *Barcelona-3*; ускорители *Celoxica*; обеспечена возможность одновременной работы разноплатформенных операционных систем *Linux* и *Windows*.

Основные технические характеристики ВВК: 64 вычислительных узла; 128 четырехъядерных процессоров *Barcelona-3* (всего 512 ядер); пиковая производительность 4,096 Тфлоп; производительность в тестовом пакете *Linpack* 78 %; объем системы хранения информации 12 Тбайт; объем оперативной памяти 2048 Гбайт (32 Гбайт/узел).

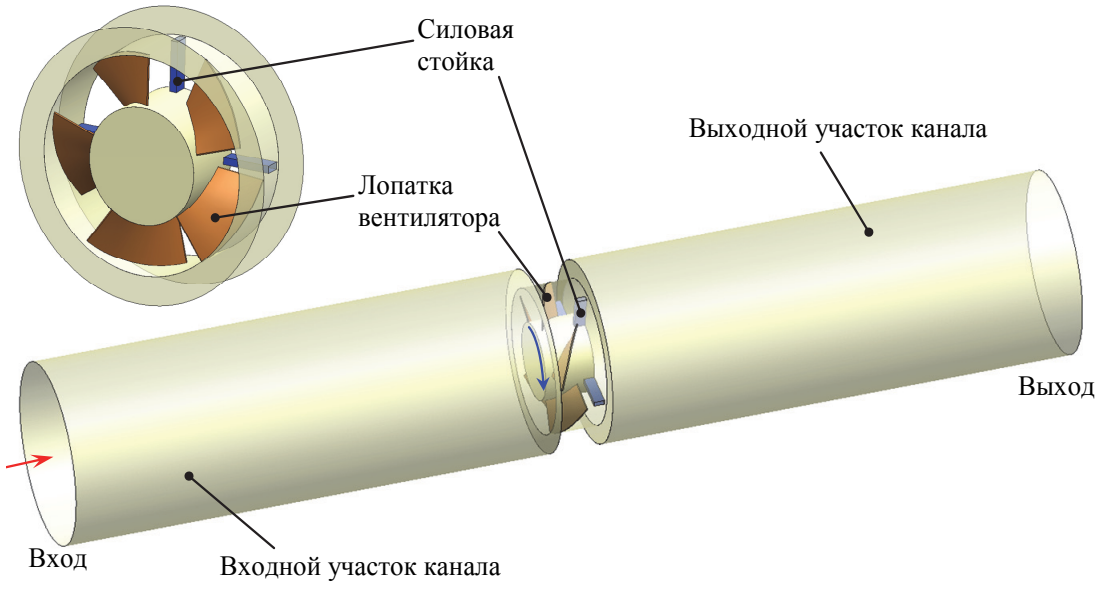


Рис. 1. Геометрическая модель вентилятора

Создана трехмерная модель, далее она импортирована в сеточный генератор *ANSYS ICEM CFD*. При выборе типа элементов сеточной модели необходимо отметить следующее: применение гексаэдрических сеточных

моделей более выгодно с точки зрения более эффективного расходования вычислительных ресурсов.

Для расчетов построены две сеточные модели с разным количеством областей и, соответственно, разным количеством интерфейсов. В первой модели расчетная область поделена на три подобласти. Первая подобласть – подобласть входа – неподвижная и имеет входной участок трубы и небольшой (длиной 2 мм) отрезок канала перед вентилятором. Вторая подобласть – подобласть вентилятора – включает в себя вентилятор и вращается. Третья подобласть – подобласть выхода – неподвижная и содержит силовые стойки и выходной участок трубы. Таким образом, первая сеточная модель содержит два ротор-статор-интерфейса с обеих сторон подобласти вентилятора.

Вторая сеточная модель содержит только две подобласти. Во второй сеточной модели подобласть вентилятора и входная подобласть объединены в одну подобласть, которая вращается. Подобласть выхода оставлена без изменений. Таким образом, для второй сеточной модели ротор-статор-интерфейс между двумя взаимно вращающимися подобластями был только один. Размер сеточных моделей приблизительно одинаков. Для первой модели общее количество узлов и элементов сетки составило 14 402 323 и 14 175 083 соответственно. Вторая модель несколько больше – полное количество узлов 17 064 676, а элементов 16 825 523.

Рассмотрение разных вариантов организации сеточной модели проводилось с целью оценить влияние ротор-статор-интерфейса на получаемые результаты.

Для вращающихся областей задавалась частота вращения 2670 об/мин (44,5 Гц), что соответствует частоте следования лопаток 222,5 Гц. На входе в расчетную область задавался расход 0,0599 кг/с и полная температура 300 К. На выходе – атмосферное статическое давление 101 325 Па. Эти условия соответствуют средней осевой скорости течения в канале примерно 2,5–3 м/с.

На поверхности лопаток, стоек, втулке и периферии канала вентилятора, где важны эффекты вязкости при генерации тонального шума, задавались условия прилипания, на стенках входного и выходного каналов, где существенно только распространение звуковых волн, задавались условия скольжения.

Результаты стационарных расчетов служили начальными данными для нестационарного расчета ротор-статор-взаимодействия (условия на интерфейсе *TransientRotorStator*).

Всего было проведено три нестационарных газодинамических расчета. Расчет № 1 был выполнен для сеточной модели 1 с четырехбайтной моделью представления чисел с плавающей точкой (*single precision*). Расчет № 2 также был выполнен для модели 1, но с применением восьмибайтной

модели представления чисел с плавающей точкой (*double precision*). Расчет № 3 был выполнен для модели 2 с четырехбайтной моделью представления чисел с плавающей точкой (*single precision*).

В ходе вычислительных экспериментов были получены мгновенные распределения осевой скорости и давления в продольном сечении расчетной области, а также в радиальном сечении при радиусе R = 46 мм (половина высоты лопатки вентилятора).

При сравнении распределений осевой скорости на поверхности R = 46 мм (рис. 2, *а*, *в*), можно увидеть, что распределения осевой скорости, полученные для моделей 1 и 2 перед вентилятором, практически не отличаются.

Из-за наличия завихренности потока в выходной части канала возникает большая зона обратных токов. Она располагается в центре канала, за секцией вентилятора с силовыми стойками.

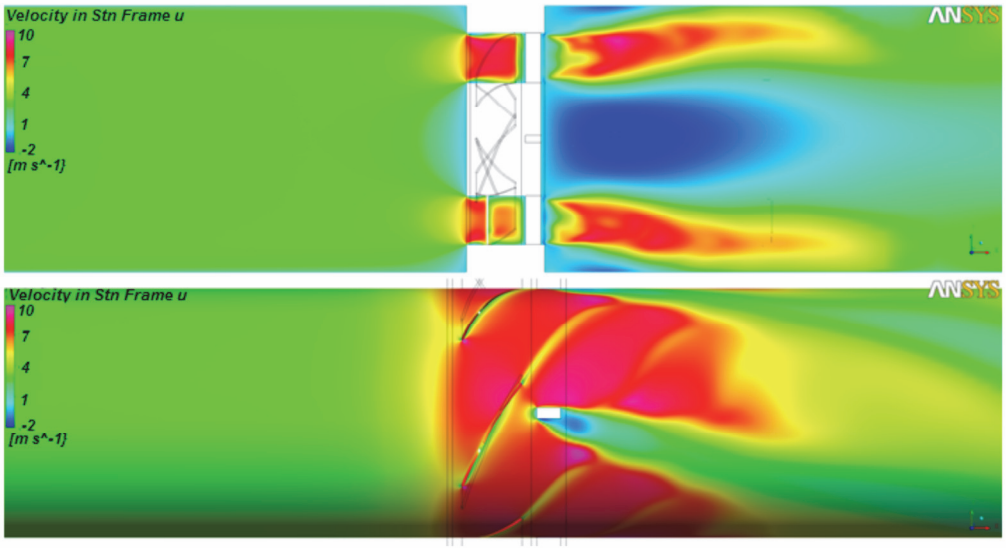
На распределениях осевой скорости, показанных на рис. 2, отчетливо видны вязкие следы, возникающие за лопатками вентилятора. Показано их взаимодействие с неподвижными стойками.

Результаты свидетельствуют о достаточно хорошей работе граничных условий для моделирования интерфейса между вращающейся и неподвижными подобластями расчетной области. При рассмотрении газодинамических полей какого-либо значимого эффекта от применения модели двойной точности не установлено. Картины течения *а* и *б* очень похожи.

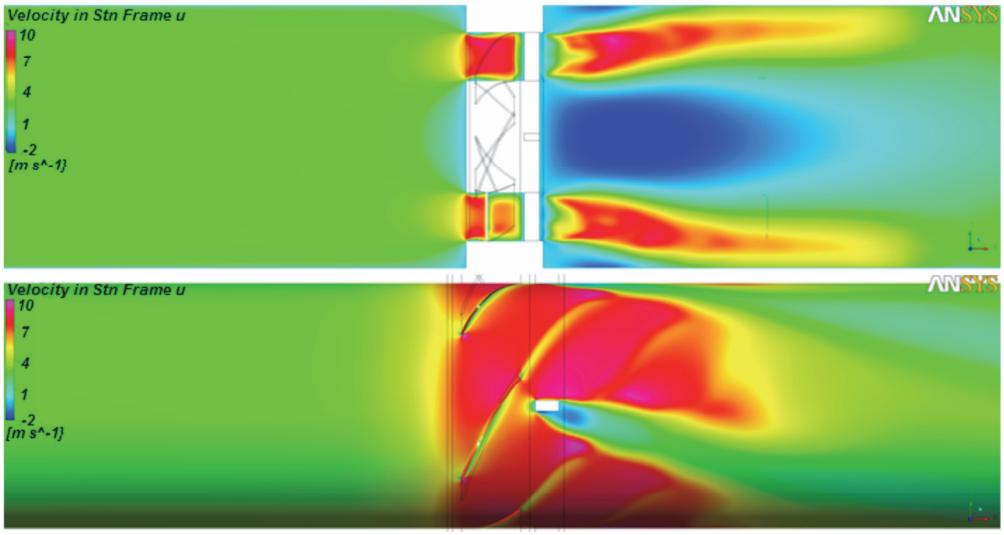
В качестве экспериментальных данных было доступно распределение амплитуды нестационарного давления на частоте следования лопаток вентилятора вдоль входного канала на радиусе R = 60 мм от оси вращения.

Для сравнения с экспериментальными данными амплитуды давления на частоте следования лопаток вентилятора, полученные в расчетах в радиальном сечении R = 60 мм, были осредненные в окружном направлении. Найденные в результате осреднения значения амплитуд для всех рассматриваемых вариантов расчетов представлены на рис. 3. Как можно видеть из графиков, хорошее совпадение с экспериментом сохраняется только на достаточно близком расстоянии от вентилятора, приблизительно 30–40 мм, что соответствует 1,5–2-осевым хордам рассматриваемого вентилятора. Вдали от вентилятора наблюдается затухание амплитуды ниже экспериментально измеренных уровней.

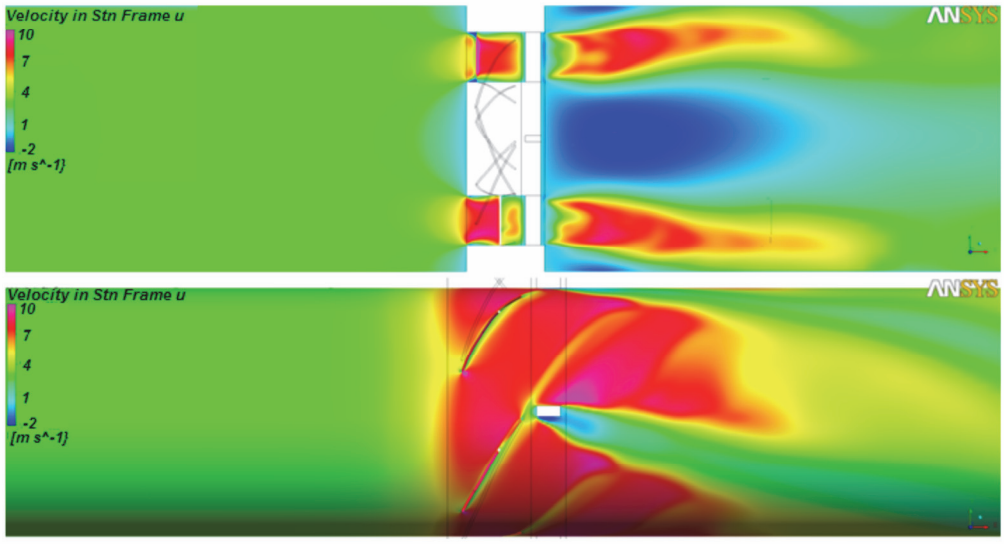
Результаты, полученные для всех вариантов расчета, очень близки между собой. Они также согласуются с аналитической теорией распространения шума в каналах.



а



б



в

Рис. 2. Мгновенные распределения осевой скорости:
а – модель 1; *б* – модель 1 (двойная точность); *в* – модель 2

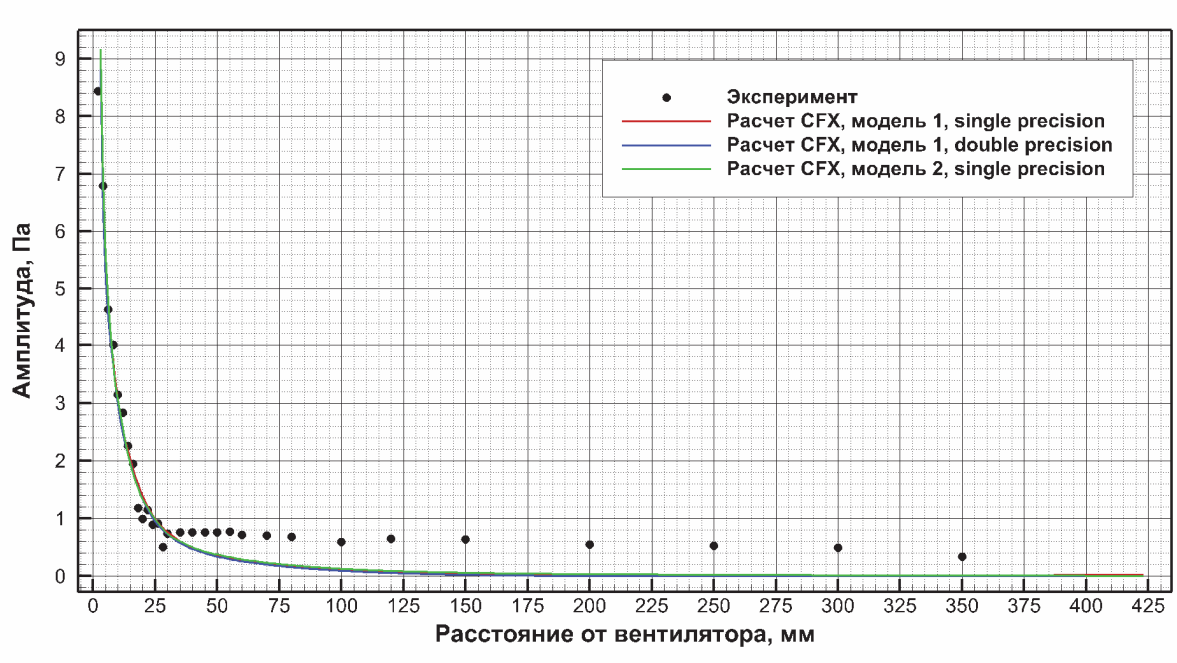


Рис. 3. Зависимость амплитуды давления на частоте следования лопаток от осевого расстояния до вентилятора

В ходе расчетов было получено ускорение вычислительного эксперимента (рис. 4), получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений, определяемое по формуле

$$S_p(n) = \frac{T_1(n)}{T_p(n)},$$

где n используется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи; $T_1(n)$ – время решения задачи на скалярной вычислительной машине; $T_p(n)$ – время выполнения скалярного алгоритма.

Ускорение определялось для нестационарного расчета на одинаковой сетке с количеством элементов 17 млн.

В результате проведенных численных исследований можно сформулировать несколько выводов и рекомендаций по проведению расчетов генерации тонального шума ротор-статор-взаимодействия в пакете *ANSYS CFX*:

1. Применение пакета *ANSYS CFX* для решения задач генерации тонального шума ротор-статор-взаимодействия возможно, но требует применения массивных параллельных вычислительных систем.

2. При создании сеточных моделей следует стремиться уменьшить количество сеточных интерфейсов между подвижными подобластями расчетной сетки. Наиболее предпочтителен вариант, когда в модели присутствует только один сеточный интерфейс.

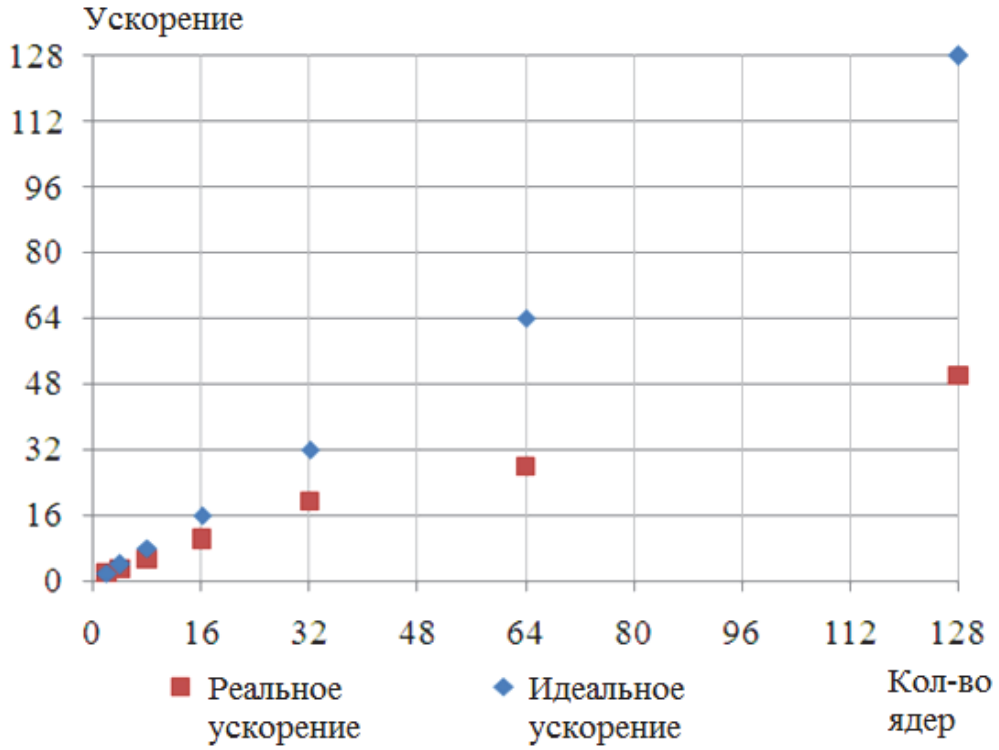


Рис. 4. Ускорение, получаемое при использовании параллельного алгоритма

3. Необходимо применение специальных неотражающих граничных условий на входной и выходной границах расчетной области. Также возможен использованный в данной работе подход с применением буферных зон около входных и выходных границ расчетной области, в которых происходит разгущение сетки.

4. Расположение поверхности для выделения модального состава для последующего решения задачи о распространения звука в каналах следует выбирать на расстоянии 0,5–1,5 осевых хорд от вентилятора, где уже нет сильных градиентов течения, связанных с действием вентилятора, но сетка еще достаточно густа для правильного описания процесса распространения генерирующихся при ротор-статор-взаимодействии волн.

5. Применение восьмибайтной модели чисел с плавающей точкой позволяет добиться более точных результатов и уменьшить отражения от сеточных интерфейсов.

Список литературы

1. Программный комплекс ANSYS CFD [Электронный ресурс] / ANSYS, Inc., 2012. – URL: http://www.ansys.com (дата обращения 30.08.2012).

2. Timouchev S., Nedashkovsky A., Pavic G. Axial fan 3D acoustic-vortex method CFD-CAA study and experimental validation // Proceedings of 3rd International symposium on Fan Noise 2007 (19–21 September, 2007), Lyon, France.

Н.В. Старостин[1], М.А. Панкратова[1], Д.В. Седаков[2]

[1] Нижегородский государственный университет им. Н.И. Лобачевского
[2] Научно-исследовательский институт
измерительных систем им. Ю.Е. Седакова, г. Нижний Новгород

ТЕХНОЛОГИЯ ПЛАНИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ В ПАРАЛЛЕЛЬНОЙ СРЕДЕ

Численное моделирование физических процессов в параллельной среде предполагает распределение аппроксимационной сетки по процессорам. Результат распределения во многом определяет количество межпроцессорных коммуникаций, коллизий во время пересылок информации, а следовательно, и время выполнения параллельной программы. В работе приводится постановка задачи распределения расчетной сетки по узлам вычислительной сети, формулируется критерий, предлагаются пути решения задачи.

Введение

Многие методики численного моделирования физических процессов основываются на аппроксимационных сетках, которые моделируются с помощью графов. На практике сеточные структуры имеют большие порядки, что приводит к необходимости их распределения по узлам параллельной вычислительной сети. Эффективность работы программы в параллельной среде во многом зависит от интенсивности межпроцессорных коммуникаций, которую можно сократить за счет удачного распределения расчетной сетки по узлам вычислительной сети. В работе приводится постановка задачи распределения расчетной сетки по узлам вычислительной сети, формулируются критерии, предлагаются пути решения задачи.

Постановка задачи

Взвешенный граф $G(V,E,w,u)$ моделирует расчетную сетку, $V=\{v_1,...,v_n\}$ – множество узлов сетки, $E\subseteq V^{(2)}$ – связи между узлами. Веса вершин $w(v_i)\in N, i=\overline{1,n},$ где N – затраты на расчет; веса ребер $u(v_i,v_j)\in N,\ i,j=\overline{1,n}.$

Матрица $\mathrm{T}=(t_{ij})_{k\times k}$ моделирует вычислительную сеть. Элемент матрицы $t_{ij}\in N$ соответствует оценке затраты на передачу данных между парой вычислителей.

Решением задачи является вектор $\mathrm{x} = \{x_1,...,x_n\}, x_i \in \{1,...,k\},\ i = \overline{1,n}$, где *i*-й элемент соответствует узлу расчетной сетки, а значение x_i определяет узел вычислительной сети, на который он распределен. Для ускорения процесса параллельного расчета на распределенной аппроксимационной сетке важно сбалансировать затраты процессоров на расчет собственных данных, чтобы время работы каждого процессора было максимально одинаковым. Пусть суммарный вес узлов сетки

$$W = \sum_{i=1}^{n} w(v_i);$$

загрузка (оценка относительного времени работы) вычислительного узла *j*,

$$W_j = \sum_{\{i:xi=j,\ i=\overline{1,n}\}} w(v_i),\ j = \overline{1,k};$$

идеальная загрузка вычислительных узлов

$$\tilde{W} = \frac{W}{k}.$$

Зададим $\varepsilon > 0$ – коэффициент отклонения от идеальной загрузки. Тогда неравенства

$$\left| \frac{W_j}{\tilde{W}} - 1 \right| \le \varepsilon,\ j = \overline{1,k} \qquad (1)$$

задают ограничения по балансировке нагрузки вычислительных узлов.

Зададим функцию

$$\beta(x, v_i, v_j) = u(v_i, v_j) \cdot t_{x_i x_j},$$

которая определяет оценку затрат на коммуникационный обмен между парой узлов вычислительной сетки.

Проблема моделирования коммуникационных обменов в программе, работающей в параллельной вычислительной сети, связана с тем, что очень сложно, а зачастую практически невозможно спрогнозировать точное время каждой коммуникации; возникновение коллизий на уровне сетевых устройств; снижение пропускной способности каналов в связи с повышением интенсивности обменов; маршруты, по которым происходит передача пакетов. Всё это значительно снижает адекватность применения модели в терминах планирования коммуникационных обменов. С практической точки зрения удобнее оперировать оценками затрат на всю совокупность коммуникационных обменов [1]. Нижняя оценка предполагает

случай, когда все обмены происходят одновременно и друг с другом не конфликтуют. В данном случае имеет место минимаксный критерий

$$F_1(x) = \max_{(v_i, v_j) \in V} \beta\left(x, v_i, v_j\right) \to \min. \quad (2)$$

Верхняя оценка предполагает, что все обмены происходят последовательно. В данном случае имеет место аддитивный критерий

$$F_2(x) = \sum_{(v_i, v_j) \in V} \beta\left(x, v_i, v_j\right) \to \min. \quad (3)$$

Проблема организации исполнения параллельной программы

Как правило, при организации исполнения параллельной программы заранее известна только структура расчетной сетки и общее число вычислительных узлов, которое будет выделено для расчета. Структура выделенного участка вычислительной сети определяется только в момент начала расчета, когда планировщик выделяет очередной задаче требуемое число вычислителей. С другой стороны, в силу того, что размер исходных данных очень велик и «не умещается» в памяти одного процессора, аппроксимационная сетка должна быть декомпозирована еще до начала расчета. В результате в начале расчета исходные данные оказываются уже распределенными по выделенным процессорам, т.е. каждый процессор владеет только частью исходных данных. Таким образом, на этапе декомпозиции данных информация о топологии выделенного участка вычислительной сети неизвестна. Поэтому в задачах (1), (2) и (1), (3) элементы матрицы принимают значения $t_{ij} = \text{const},\ i \neq j,\ t_{ii} = 0,\ i, j = \overline{1,k}$. В этой постановке задача (1), (2) соответствует задаче минимизации ширины ленты графа [2], а задача (1), (3) – классической задаче *k*-разбиения графа [3]. Обозначенные задачи реализованы с помощью многоуровневых подходов в рамках библиотеки MATRUZ [4].

Назначение подграфов разбиения на вершины графа топологии вычислительной сети с минимизацией затрат на коммуникации

После запуска параллельной программы становится доступной информация о топологии выделенного участка вычислительной сети. Предварительная декомпозиция и упорядочение исходных данных позволяет снизить затраты на коммуникации. Однако данные затраты можно сократить дополнительно за счет перенумерации вычислительных узлов с учетом известной топологии используемого фрагмента сети. Сформулируем задачу в терминах квадратичной задачи о назначениях [5].

Введем матрицу $\mathrm{B} = \left(b_{ij}\right)_{k\times k}$, задающую связи между частями декомпозированной сетки. Здесь

$$b_{ij} = \sum_{\{l:x_l=i,\ l=\overline{1,n}\}} \sum_{\{m:x_m=j,\ m=\overline{1,n}\}} u'\left(v_l, v_m\right), i, j = \overline{1,k},$$

где

$$u'\left(v_i, v_j\right) = \begin{cases} u\left(v_i, v_j\right), & \text{если } \left\{v_i, v_j\right\} \in E, \\ 0, & \text{в противном случае.} \end{cases}$$

Решением задачи является перестановка $p = \left\{p(1), ..., p(k)\right\}$, которая ставит соответствие *i*-й части аппроксимационной сетки процессор $p(i), i = \overline{1,k}$. Целью задачи является минимизация суммарных издержек на пересылку данных:

$$\min_{p\in\Pi} \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} t_{p(i)p(j)}. \qquad (4)$$

В общем случае задача (4) является *NP*-трудной [5].

Эвристические методы

С точки зрения технологии практического получения решений задачи (4) можно отметить конструктивные [6] и итерационные [7] эвристические алгоритмы. Конструктивные алгоритмы генерируют решения, но не содержат механизмов их «улучшения». В свою очередь, итерационные методы нацелены на исследование окрестности начального решения, но их проблема состоит в том, что поиск заканчивается при получении локального оптимума. На решении этих проблем сосредоточены такие методы, как имитации отжига [8], генетические алгоритмы [9] и др. В работе предлагается гибридный подход, который позволяет исследовать окрестность некоторого начального решения генетическим методом и содержит механизмы перехода от одного начального решения к другому.

Генетический метод решения задачи

Предлагается алгоритм, который на каждой итерации генетическим поиском исследует окрестность, в которую попадают все решения, отдаленные от текущего на расстояние[1], не превышающее заданное. Алгоритм,

[1] Подразумевается расстояние Хэмминга – число несовпадающих позиций в строках, соответствующих решениям

основанный на подобной стратегии, будем называть улучшающим генетическим алгоритмом (*improving genetic algorithm, IGA*). В листинге (см. листинг 1) представлен псевдокод алгоритма.

Контракт *IGA* (см. листинг 1, строка 1) заключается в том, что функция поиска решения получает на вход некоторое начальное решение x и, как результат, возвращает найденное решение. Решение реализуется сущностью *solution* – по сути, программной конструкцией, которая непосредственно соответствует решению задачи. В процессе работы *IGA* (строка 14) контролирует лучшие решения с точки зрения функции приспособленности μ. *IGA* генерирует начальную популяцию p (строка 3), при этом размер популяции неизменен в процессе поиска и является параметром работы *IGA*. Каждый элемент популяции описывается бинарной строкой определенной длины, которая кодирует технологию перехода от начального решения x в новое решение.

```
1    solution IGA(solution x)
2     {
3     binaryString[] p =
4    generate();
5     while (!iga_stop())
6     {
7     evaluate(p, x);
8     while (!ga_stop())
9     {
10    breeding();
11    crossover();
12    mutation();
13    select();
14    }
15    x = record(p);
16    }
17    return x;
      }
```

Листинг 1. Структура IGA.

Внешний цикл работы *IGA* (строка 4) не прекращается до тех пор, пока не наступят условия останова. Нами в основу функции iga_stop() положена тривиальная логика, ограничивающая число итераций алгоритма.

В *IGA* функция оценки evaluate(p, x) (строка 6) заключается в расчете оценки каждой бинарной строки текущей популяции p относительно текуще-

го решения x. Для каждой строки s выполняется акт перехода от начального решения x в новое решение x^s согласно технологии, закодированной в строке s. Формально технологию перехода можно представить в виде отображения $\psi : X \times S \to X$. Для решения x^s рассчитывается значение приспособленности μ. Важно отметить, что при смене начального решения необходимо пересчитывать значения приспособленности для всей популяции.

Внутренний цикл (строки 7–13) осуществляет традиционный генетический поиск, на его протяжении начальное решение x не изменяется. Функция останова ga_stop() в нашей реализации просто ограничивает число итераций. Процедура breeding() (см. листинг 1, строки 9) выполняет отбор родительской пары. В наших экспериментах мы использовали стратегию панмиксии: право сформировать родительскую пару имеют все особи популяции в равной вероятностью. Процедура crossover() (строка 10) рекомбинацией родительских строк получает две новые строки потомков. Процедура mutation() (строка 11) генерирует случайные изменения в строках потомков. Все новые решения оцениваются, и в процедуре select() (строка 12) происходит замена худших решений из текущей популяции на новые решения, полученные в рамках текущей итерации.

Рекорд, полученный в результате работы генетического поиска (внутренний цикл), можно использовать в качестве начального решения (строка 14). Лучшее найденное решение возвращается как результат работы алгоритма (строка 16).

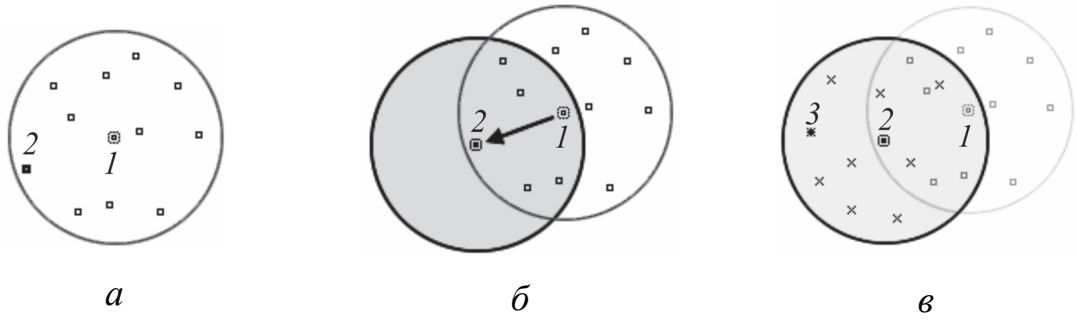


Рис. Процесс поиска *IGA*: *а* – внутренний цикл *IGA* осуществляет генетический поиск в окрестности решения *1*; *б* – внешний цикл *IGA* выполняет переход в лучшее решение *2*; *в* – внешний цикл *IGA* пересчитывает решения и осуществляет генетический поиск в окрестности решения *2* и находит новый рекорд *3*

На рисунке демонстрируется схема исследования пространства поиска *IGA*. Когда *IGA* стартует с некоторого начального решения (рисунок, *а*), то внутренний цикл *IGA* осуществляет генетический поиск в заранее определенной окрестности (задается длиной кодирующей строки) начального

решения. Результат работы внутреннего цикла *IGA* – это новый рекорд, который принимается новым начальным решением (рисунок, *б*), при этом решения текущей популяции отображаются в новую окрестность. Внутренний цикл *IGA* исследует окрестность нового решения и находит новый рекорд (рисунок, *в*), который является кандидатом на новое начальное решение. Таким образом, *IGA* перебирает одну окрестность за другой, пока не наступят условия останова.

Вычислительный эксперимент

Для оценки качества работы алгоритма были выбраны известные тестовые примеры для задачи *QAP* [10]. В ходе вычислительного эксперимента было проведено 134 теста, из которых на 90 примерах удалось достигнуть известного оптимального или рекордного решения.

Список литературы

1. Pellegrini F. Scotch and LibScotch 6.0 User's guide / University of Bordeaux 1 & LaBRI. – Bordeaux, 2012. – 162 p.

2. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений: пер. с англ. – М.: Мир, 1984. – 333 с.

3. Батищев Д.И., Старостин Н.В. *K*-разбиение графов // Вестник ННГУ. Математическое моделирование и оптимальное управление. – Н. Новгород, 2000. – С. 25–37.

4. Старостин Н.В., Филимонов А.В. Разработка и реализация параллельного многоуровневого алгоритма равномерного разбиения нераспределенного графа // Материалы XIII Всерос. конф. (Н. Новгород, 14–16 ноября 2013 г.). – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2013. – С. 243–247.

5. Burhard R.E., Cela E., Pardalos P.M., Pitsoulis L.S. The quadratic assignment problem // Handbook of Combinatorial Optimization / ed. D.-Z. Znu, P.M. Pardalos. – Kluwer, 1998. – Vol. 3.

6. Chen S., Eshaghian M.M. A fast recursive mapping algorithm / Department of computer and information science New Jersey Institute of technology. – Newarc. – NJ 07102.

7. Gevezes T.P., Pitsoulis L.S. A new greedy algorithm for the quadratic assignment problem // Optimization Letters. – 2013. – Feb. – Vol. 7, iss. 2. – P. 207–220.

8. Bollinger S.W., Midkiff S.F. Heuristic Technique for Processor and Link Assignment in Multicomputers // IEEE Trans. On Computers. – 1991. – March. – Vol. 40, № 3. – P. 325–333.

9. Батищев Д.И., Неймарк Е.А., Старостин Н.В. Применение генетических алгоритмов к решению задач дискретной оптимизации: учеб. пособие. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2006. – 136 с.

10. Burkard R.E., Karisch S.E., Rendl F. QAPLIB – A Quadratic Assignment Problem Library // Journal of Global Optimization. – 1997. – June. – Vol. 10, iss. 4. – P. 391–403.

Н.В. Старостин, А.В. Филимонов

Нижегородский государственный университет им. Н.И. Лобачевского

ДЕКОМПОЗИЦИЯ ГРАФОВЫХ СТРУКТУР В ТОПОЛОГИЧЕСКОМ СИНТЕЗЕ ИНТЕГРАЛЬНЫХ СХЕМ

Рассматривается задача декомпозиции графовых структур, которая возникает на этапах физического проектирования сверхбольших интегральных схем. Предлагается способ решения задачи декомпозиции графа с помощью разработанного авторами распределенного параллельного итерационного многоуровневого алгоритма декомпозиции графа и его реализации в виде библиотеки программного кода *MATRUZ*.

Введение

Процесс синтеза топологии интегральных схем (ИС) представляет собой сложную многостадийную технологию, архитектура которой сильно зависит от технологического процесса производства ИС, ее структуры и функциональных, параметрических и других требований заказчика. В общем случае физическое проектирование включает следующие этапы: компоновку схемы, планирование кристалла, размещение элементов, трассировку цепей, синтез и оптимизацию синхросигналов. Основная сложность в топологическом синтезе связана с решением *NP*-трудных математических проблем. Ярким примером подобной математической проблемы является задача декомпозиции, в терминах которой формулируется проблема компоновки схемы – схему требуется распределить на подсхемы с минимальным числом трасс на внешней шине. Нередко задача декомпозиции эксплуатируется на этапе глобального размещения [1] с целью выбора оптимальной компоновки элементов по блокам, рядам или областям монтажного пространства ИС для их последующего детального размещения.

Постановка задачи

Математическим объектом, который естественно описывает принципиальную схему для задач компоновки, является гиперграф $H(V, E, w)$. Вершины гиперграфа V соответствуют элементам схемы, а натуральный вес вершины, определяемый отображением $w: V \to N$, характеризует число занимаемых элементом посадочных мест. Гиперребра E моделируют электрические цепи. В задаче декомпозиции требуется разрезать гиперграф на k подграфов $(V_1, ..., V_k)$ определенных размеров. Главная цель декомпозиции направлена на минимизацию внешних связей – совокупность гиперребер, которые целиком не попали ни в один из подграфов разбиения:

$$V_i \cap V_j = \varnothing, \;\; i, j = \overline{1,k}, i \neq j; \bigcup_{i=1}^{n} V_i = V, \tag{1}$$

$$W_i^- \leq \sum_{v_j \in V_i} w(v_j) \leq W_i^+, \quad i = \overline{1,k}, \tag{2}$$

$$Q(V_1, ..., V_k) = \sum_{e_i \in E} q(e_i) \to \min,$$

где

$$q(e_i) = \begin{cases} 0, \text{ если } \forall v_l \in e_i, v_i \in V_a, a \in \{1, ..., k\}, \\ 1, \text{ в противном случае.} \end{cases} \tag{3}$$

Гиперграфы – сложные с точки зрения хранения и обработки на вычислительной технике объекты, поэтому имеет смысл перейти от гиперграфовой формализации задачи компоновки к формализации, основанной на классических графах. Достигнуть этого можно разными стратегиями, например заменяя каждое гиперребро на полносвязную компоненту. В качестве базовой стратегии нами выбрана схема перехода к графу Кёнига: каждое гиперребро трансформируется в новую вершину графа нулевого веса, а инцидентные ему вершины связываются ребрами с новой вершиной.

Метод решения

Поскольку на практике размерности СБИС велики, задачи декомпозиции графа также имеют размерности, исчисляемые сотнями миллионов, а иногда миллиардами. Решение задач декомпозиции таких размерностей на современных персональных компьютерах невозможно, но возможно на так называемых суперкомпьютерах. Последние отличаются от классических вычислительных машин наличием распределенной памяти и большого количества обрабатывающих устройств (процессоров). Алгоритмы, реализуемые для такого рода компьютеров, должны обладать структурой, учитывающей эти особенности.

Предлагаемый авторами алгоритм представляет собой итерационный многоуровневый параллельный алгоритм декомпозиции неориентированных графов с распределенным представлением в памяти. Такой алгоритм эксплуатирует многоуровневую парадигму, предполагающую поэтапное построение решения исходной задачи [2]. Многоуровневый алгоритм строит ряд приближений исходной задачи, называемых редуктантами и отличающихся уменьшающимися размерностями. Решения задач-редуктантов используются как фундамент для построения исходной задачи. Структура многоуровневого алгоритма декомпозиции представлена на рисунке. Алгоритм состоит из трех основных стадий: стадии загрубления, в ходе которой строятся редуктанты, стадии рекурсивного поиска начального разбиения или оптимизации, осуществляющей работу с редуктантом наименьшей размерности, и фазы восстановления, в ходе которой путем ряда последовательных проекций решения задачи меньшей размерности на задачу большей размерности и оптимизации решения осуществляется построение решения исходной задачи. В ходе фазы загрубления и восстановления, а также оптимизации работают оригинальные, предложенные авторами параллельные алгоритмы на распределенных данных. Кроме того, реализован параллельный алгоритм рекурсивной бисекции с балансировкой решения.

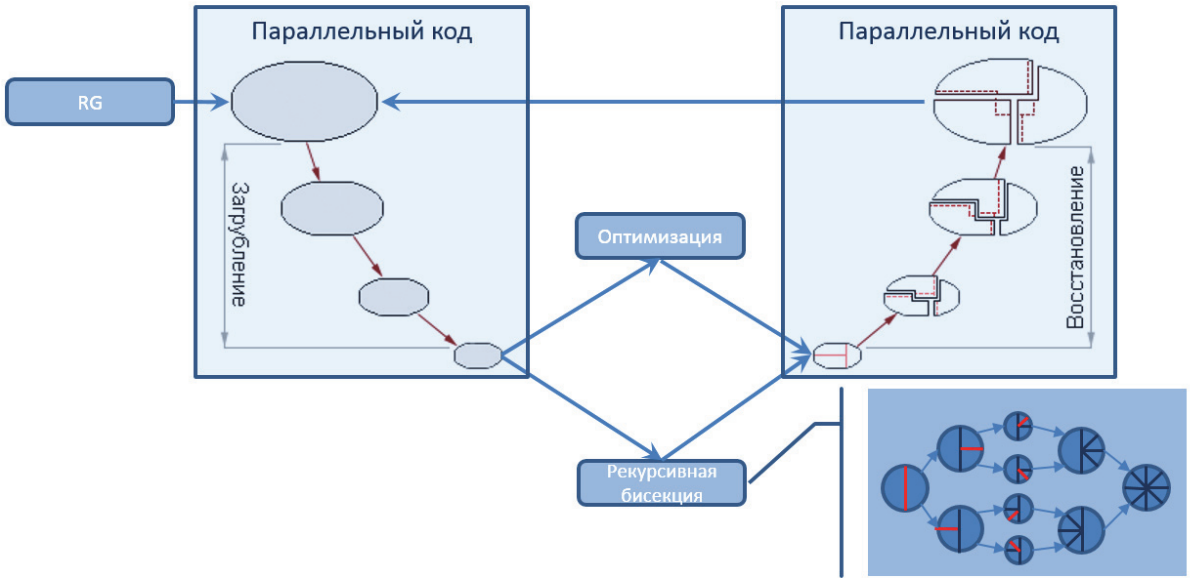


Рис. Структура многоуровневого алгоритма

Хотелось бы отдельно отметить, что количество подграфов разбиения k никоим образом не соотносится с количеством используемых при решении задачи вычислительных узлов или процессоров. Часто реализуемые алгоритмы обработки графов предполагают равенство этих величин,

что в данном случае было бы существенным ограничением, накладываемым на решаемую задачу.

Программная реализация

Предлагаемый алгоритм был реализован в виде программной библиотеки, написанной на языке *C++*, и интерфейса на языке *C*. В качестве средства коммуникации используется библиотека *MPI*.

Можно выделить следующие особенности библиотеки:

– способность работать на системах с распределенной памятью;

– процедуры компиляции под *Linux* (протестировано на компиляторах *GCC* 4.8, *Intel* 10-го и 11-го семейств, реализации *MPI – MPICH*, *MVAPICH*) и *Windows* (компиляторы *Microsoft VisualStudio* 2010 и 2012, *Intel* 10-го и 11-го семейств, реализация *MPI – MPICH*);

– процедуры загрузки и сохранения графов и решений задачи декомпозиции в различных форматах;

– возможность компиляции с поддержкой 32- и 64-битных типов данных;

– возможность проверки входных данных на корректность и корректировки.

Список литературы

1. Балашов В.В., Старостин Н.В., Филимонов А.В. Эволюционно-генетический алгоритм линейного размещения цифровых элементов интегральных схем // Системы управления и информационные технологии. – 2009. – № 1(35). – С. 64–68.

2. Старостин Н.В., Филимонов А.В. Разработка и реализация параллельного многоуровневого алгоритма равномерного разбиения нераспределенного графа // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XIII Всерос. конф. (Н. Новгород, 14–16 ноября 2013 г.) / под ред. проф. В.П. Гергеля. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2013. – С. 243–247.

К.С. Стефанов

Научно-исследовательский вычислительный центр им. М.В. Ломоносова

ОБ ОДНОМ ПОДХОДЕ К МОНИТОРИНГУ ПРОИЗВОДИТЕЛЬНОСТИ ПОТОКА ЗАДАЧ НА СУПЕРКОМПЬЮТЕРЕ

Мониторинг производительности вычислительных комплексов в целом и отдельных программ – важная задача, решение которой позволяет оценить эффективность использования имеющихся ресурсов и предложить пути увеличения эффективности выполняемых программ. Имеющиеся системы позволяют оценивать производительность либо вычислительного комплекса в целом, либо отдельных задач. В статье рассмотрен подход к созданию системы мониторинга производительности, который позволит исследовать производительность всего потока задач, выполняющихся на вычислительном комплексе.

Введение

Системы мониторинга суперкомпьютерных комплексов – важная часть их программного обеспечения, призванная обеспечить максимально эффективное и бесперебойное функционирование, а в случае возникновения нештатных ситуаций – обеспечить сохранность оборудования и оповещение персонала. Кроме того, возникла еще одна важная область использования систем мониторинга – мониторинг производительности. Существующие системы мониторинга справляются с задачами обеспечения сохранности оборудования, хотя достижение нужного времени реакции для больших комплексов может быть непростой задачей. Что касается мониторинга производительности, существующие системы предназначены для исследования отдельных конкретных задач и не могут быть использованы для исследования производительности всего потока задач, работающих на суперкомпьютере, и оценки эффективности использования суперкомпьютера в целом.

Архитектура систем мониторинга

На рис. 1 изображена общая архитектура систем мониторинга. За получение информации отвечают агенты, запущенные на вычислительных узлах. При получении информации о состоянии других компонент вычислительной системы ту же роль могут играть агенты *SNMP* или компоненты, отвечающие за получение данных по каким-то другим протоколам. Собранные таким способом данные передаются в серверную часть системы мониторинга, где происходит их обработка, сохранение какой-то части

информации в базу данных, выделение событий, реагирование на события, и имеются части, предназначенные для представления данных пользователю, в частности, визуализации.

Существующие системы мониторинга [1–4] построены по описанной схеме. Их объединяет отсутствие обработки собираемых данных в точке их получения (на вычислительном узле) и жесткая конфигурация путей передачи данных. Пути передачи данных задаются при конфигурировании системы и не меняются в процессе работы. При этом для обеспечения надежности и производительности серверная часть может быть распределенной (иерархической) и/или дублированной.

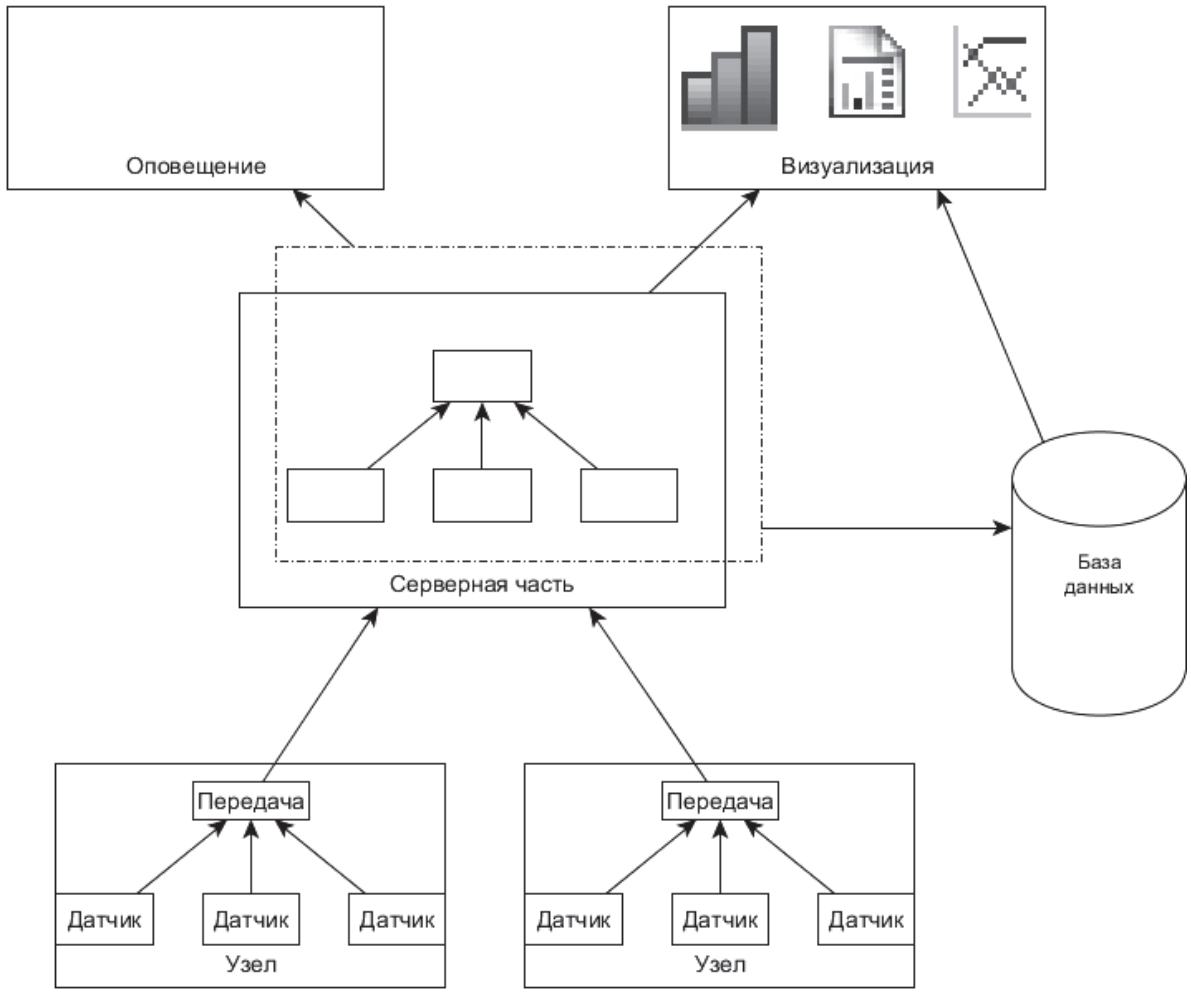


Рис. 1. Общая архитектура систем мониторинга

Такая архитектура хорошо работает для задач отслеживания аварийных ситуаций (перегрев, протечки, проблемы с электропитанием, пожар), доступности отдельных компонентов вычислительной системы и их пригодности к работе. Период сбора данных при этом составляет от десятков секунд до десятков минут, а также используется пассивный мониторинг (реагирование на сигналы о событиях, приходящие от внешних источников).

Задача мониторинга производительности отличается от других задач мониторинга. Сбор данных должен производиться чаще: период должен составлять единицы секунд, а потенциально – доли секунд. Как следствие, объем данных, который нужно обрабатывать, значительно больше. Кроме того, вычислительная задача – объект динамический, ее расположение и источники данных, относящиеся к конкретной задаче, определяются в момент запуска этой задачи.

Предложено много подходов к мониторингу производительности задач [5–8]. Они объединены тем, что предназначены для мониторинга производительности конкретных задач. С этой целью в момент запуска задачи производится запуск агентов на узлах, где будет выполняться задача, данные от которых собираются в базу данных, а после окончания работы задачи производится анализ.

Этот подход вполне оправдан при анализе отдельных задач. Однако если ставить целью анализ всего потока задач, исполняемых на вычислительной системе, то такой подход приносит множество лишних накладных расходов.

При анализе всего потока задач детальное рассмотрение каждой задачи невозможно, да и не представляет интереса. От каждой задачи будет использоваться небольшое число метрик производительности, которые будут подсчитаны один раз. Подробный анализ нужен только для отдельных задач. Поэтому данные мониторинга от большинства задач будут один раз записаны в базу данных и один раз из нее прочитаны при подсчете соответствующих метрик.

Однако такой режим использования приводит к сильному падению производительности системы хранения данных, используемой базой данных, либо к ее удорожанию. Действительно, для самых распространенных систем хранения данных на основе жестких дисков наиболее производительным режимом является режим линейной записи или линейного чтения. Однако одновременная запись данных мониторинга и чтение данных для подсчета метрик завершившихся задач приводит к переходу в режим случайного доступа к данным, что для систем на основе жестких дисков дает падение производительности на два порядка и более. Решением могут стать системы на основе твердотельных накопителей (*SSD*), однако они гораздо дороже систем на жестких дисках и, кроме того, деградируют при большом количестве перезаписей.

Если задачей является анализ производительности вычислительной системы в целом, то запуск отдельной инфраструктуры для получения данных конкретной задачи приводит к тому, что фактически создаются две

параллельные системы сбора данных: одна для оценки производительности вычислительной системы в целом, работающая постоянно, и вторая, запускаемая для конкретных задач.

Предлагаемый подход

Для решения вышеуказанных проблем в рамках единой системы мониторинга мы предлагаем систему, основанную на следующих принципах:

а) возможность направлять копии потоков данных по разным путям;

б) обеспечение динамической переконфигурации режимов работы системы мониторинга (путей передачи данных, параметров сбора данных, путей обработки данных);

в) обеспечение возможности вычисления метрик производительности для отдельных задач во время сбора данных, без записи их на диск с последующим считыванием;

г) перенос части обработки данных мониторинга на вычислительные узлы.

Направление копии данных по разным путям позволяет одновременно вычислять метрики как для системы в целом, так и для задачи, выполняющейся на конкретном вычислительном узле. Для этого же служит переконфигурация путей передачи данных в процессе работы: при запуске задачи создается новый путь, по которому идут данные конкретной задачи.

Для обеспечения вычисления метрик производительности задач во время сбора данных нужно уметь получить выборку данных для конкретной задачи в одной точке, где и следует произвести вычисление метрики. Создание такой точки при запуске задачи и сбор туда данных для задачи, и только их, при помощи динамически сконфигурированных путей, дает возможность вычислений в процессе работы. Отметим, что для полноценного мониторинга производительности нужны данные не только от вычислительных узлов, на которых работает задача, но и общих элементов вычислительной системы: систем хранения данных, сетевых коммутаторов и т.п. [7].

Наконец, вынос части обработки на вычислительные узлы позволит существенно увеличить производительность системы мониторинга и ее возможности по обработке информации. Обычно избегают какой-либо обработки информации на вычислительных узлах, мотивируя это необходимостью уменьшения влияния на выполняемую на узле задачу. Однако использование технологий типа *Simultaneous multithreading* (*SMT*) или *Hyper-threading* в процессорах *Intel* позволяет загрузить недоиспользованные основной задачей ресурсы процессора другой деятельностью с минимальным влиянием на основную задачу. В то же время виртуальные ядра, доступные

при включении *SMT*, обычно не используются для выполнения вычислительных задач. Также стоит отметить, что даже если мы ограничимся незначительной долей ресурсов вычислительных узлов крупной вычислительной системы, чтобы сохранить незначительное влияние на поведение основной вычислительной задачи, мы можем получить суммарную производительность, для получения которой на отдельном оборудовании может понадобиться значительное количество серверов.

Далее перейдем непосредственно к описанию разрабатываемой системы. Разрабатываемая система является модульной. Вся обработка информации о состоянии вычислительной системы производится в узлах системы мониторинга, передача информации между узлами идет по связям между ними. Связь может быть как между узлами, работающими на одном сервере (вычислительном узле), так и между узлами, работающим на разных серверах. При этом связи могут изменяться динамически в процессе работы системы по командам узлов системы мониторинга.

На всех серверах и вычислительных узлах работают агенты системы мониторинга. Агент системы мониторинга обеспечивает работу узлов, которые и осуществляют получение и обработку информации и реагирование на наступающие события. Различие между агентами, работающими на вычислительных узлах и серверах, выделенных под серверную часть системы мониторинга, заключается только в их конфигурации, с точки зрения функциональности самого агента различий нет.

Вся информация между узлами передается в унифицированном формате. Семантика данных должна быть известна только узлу, отвечающему за обработку конкретного типа данных. Все другие узлы могут обеспечить передачу данных.

Узел системы мониторинга, который предназначен для получения информации о состоянии вычислительной системы (сервера, на котором он работает, или другого оборудования), мы называем датчиком. Основная задача датчика – периодически опрашивать внешние по отношению к системе мониторинга источники данных и передавать их значения для дальнейшей обработки.

Узлы, которые не осуществляют получение или обработку информации мониторинга, мы называем управляющими узлами. Их задача – выдавать команды на изменение конфигурации других узлов системы мониторинга, например задавать сетевые адреса для передачи информации.

Остальные узлы осуществляют обработку информации мониторинга.

Еще раз подчеркнем, что программный интерфейс и возможности, предоставляемые агентом мониторинга, не зависят от типа узла, деление по типам скорее логическое.

Таким образом, мы можем выстраивать произвольные конфигурации путей передачи информации мониторинга. Пример такой конфигурации показан на рис. 2.

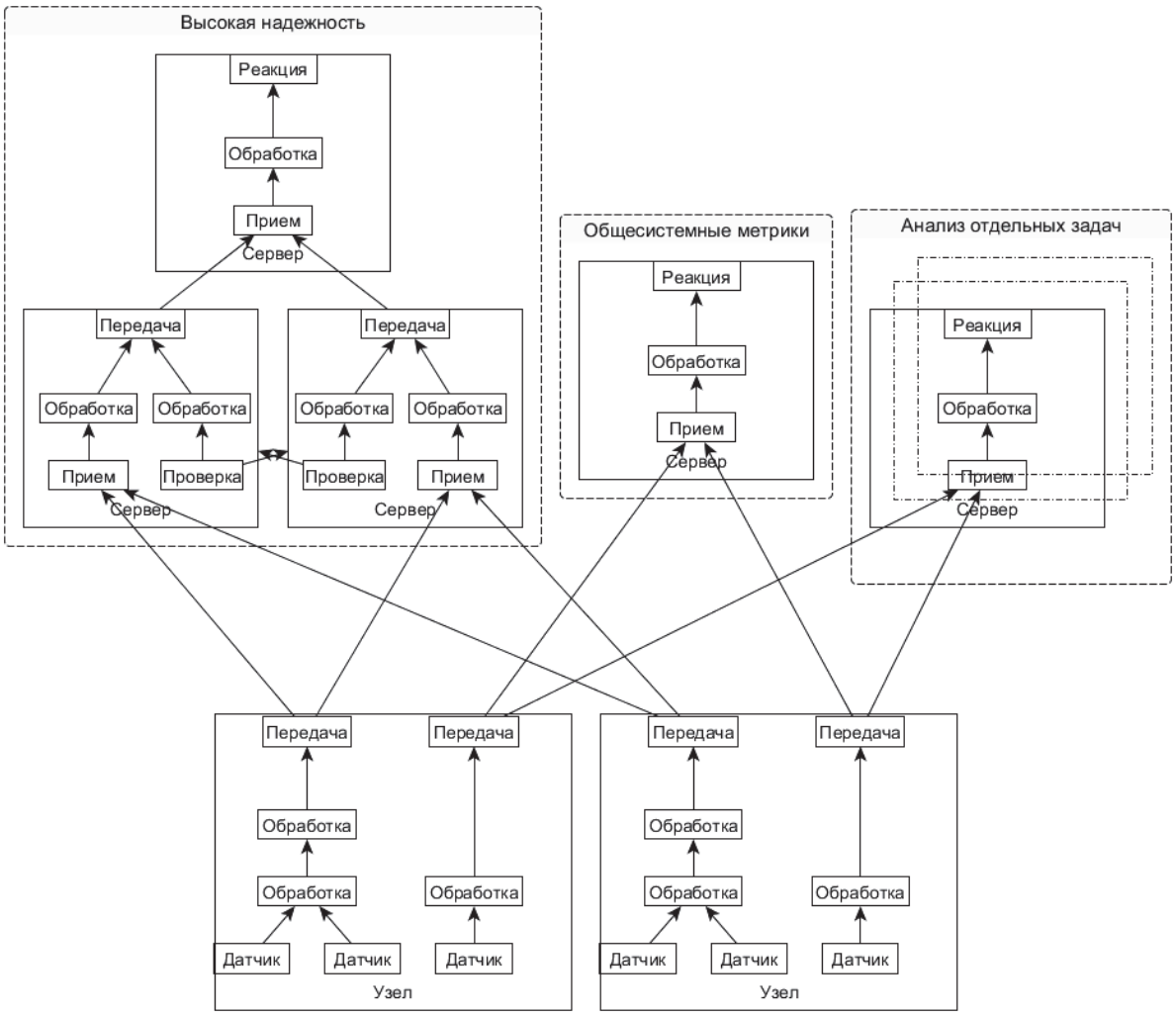


Рис. 2. Возможная конфигурация системы мониторинга

Для мониторинга производительности отдельных задач при запуске конкретной задачи конфигурация агентов мониторинга на вычислительных узлах, выделенных этой задаче, меняется таким образом, чтобы информация о состоянии этих вычислительных узлов передавалась на расположенный в серверной части системы мониторинга узел системы мониторинга, специально выделенный под обработку данных именно этой задачи. Этот узел системы мониторинга в серверной части также создается перед запуском вычислительной задачи и уничтожается после ее завершения и завершения обработки информации, связанной с этой задачей.

Часть промежуточной обработки информации выносится в агенты мониторинга, работающие на вычислительных узлах. Некоторые данные поставляются операционной системе в виде счетчиков, значение которых по-

стоянно увеличивается (например, количество байт, прошедших через сетевой интерфейс). Абсолютные значения этих датчиков не представляют интереса, реально используется скорость их изменения. Эта скорость вполне может вычисляться прямо в месте получения информации. Другим примером обработки, который можно вынести на вычислительные узлы, является сравнение значений датчиков этих вычислительных узлов с заданным порогом.

Также можно создать распределенную систему отслеживания доступности вычислительных узлов. Каждый узел может быть сконфигурирован отслеживать доступность одного или нескольких соседних узлов, таким образом распределяя нагрузку на сеть по всей системе, и сигнализировать в серверную часть только при недоступности другого узла. Централизованное отслеживание доступности с малым временем реакции является довольно ресурсоемкой задачей, в то время как отслеживание доступности единиц узлов не требует значительных ресурсов.

В настоящее время ведутся работы по реализации системы с указанной архитектурой. На основе уже реализованного прототипа были проведены оценки производительности. Была достигнута скорость передачи и приема данных в 350 тыс. пакетов в секунду, что составило примерно 230 Мбит/с. В другом эксперименте проверялась возможность обрабатывать данные от многих источников, для чего осуществлялся подсчет средней загрузки процессора для примерно 5,5 тыс. вычислительных узлов. Скорость получения данных серверной частью составила примерно 47,5 тыс. значений в секунду, при этом загрузка процессора серверной частью составила менее 4 % одного ядра процессора *Intel Xeon X5670*, работающего на частоте 2,93 ГГц.

Работа выполняется при финансовой поддержке РФФИ, грант № 13-07-00775 А.

Список литературы

1. Nagios – The Industry Standard in IT Infrastructure Monitoring [Электронный ресурс]. – URL: http://www.nagios.org (дата обращения: 10.09.2014).

2. Zabbix [Электронный ресурс]. – URL: http://www.zabbix.com (дата обращения: 10.09.2014).

3. Cacti® – The Complete RRDTool-based Graphing Solution [Электронный ресурс]. – URL: http://www.cacti.net (дата обращения: 10.09.2014).

4. Massie M.L., Chun B.N., Culler D.E. The ganglia distributed monitoring system: design, implementation, and experience // Parallel Computing. – 2004. – Vol. 30, № 7. – С. 817–840.

5. Dynamic monitoring of high-performance distributed applications / D. Gunter [et al.] // Proceedings 11th IEEE International Symposium on High Performance Distributed Computing. IEEE Comput. Soc. – 2002. – P. 163–170.

6. HPCVIEW: A Tool for Top-down Analysis of Node Performance / J. Mellor-Crummey [et al.] // The Journal of Supercomputing. – 2002. – Vol. 23, № 1. – С. 81–104.

7. A Holistic Approach for Performance Measurement and Analysis for Petascale Applications / H. Jagode [et al.] // Computational Science – ICCS 2009 Lecture Notes in Computer Science. / ed. by G. Allen [et al.]. – Berlin; Heidelberg: Springer, 2009. – С. 686–695.

А.В. Сысоев, К.А. Баркалов, В.П. Гергель

Нижегородский государственный университет им. Н.И. Лобачевского

БЛОЧНАЯ МНОГОШАГОВАЯ СХЕМА ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ ЗАДАЧ МНОГОМЕРНОЙ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

Рассматриваются задачи многомерной многоэкстремальной глобальной оптимизации. Предложен новый подход к решению указанных задач, комбинирующий две схемы редукции размерности: многошаговую и основанную на кривых Пеано. Предложен параллельный алгоритм, реализующий данный подход. Представлена схема *MPI*-реализации параллельного алгоритма для систем с распределенной памятью. Приведены результаты численных экспериментов, показывающие ускорение параллельной реализации по сравнению с последовательным прототипом.

Введение

Методам решения задач многомерной многоэкстремальной глобальной оптимизации посвящена обширная литература (см. обзоры в [1–3]). Численное решение указанных задач характеризуется так называемым «проклятием размерности» – экспоненциальным ростом числа требуемых испытаний. При этом использование эффективных методов оптимизации, строящих не всюду плотные сетки (плотные в областях возможного расположения экстремумов и более редкие вне их), лишь частично снижает остроту проблемы. Естественным подходом для уменьшения времени, требуемого для получения решения, является использование параллельных реализаций методов оптимизации, ориентированных в первую очередь на системы с распределенной памятью, содержащие в современных условиях тысячи и десятки тысяч процессоров и соответственно сотни тысяч и миллионы вычислительных ядер.

В работе рассматривается класс параллельных характеристических алгоритмов, введенный в статье [4]. Там же показано, что по отношению к своим последовательным прототипам параллельные характеристические алгоритмы являются эффективными при небольшом числе (2–4) используемых процессоров. Причина низкой эффективности при большем числе процессоров состоит в значительном числе избыточных точек испытаний, порождаемых параллельными версиями. В данной работе предложена новая схема использования параллельных характеристических алгоритмов, позволяющая обойти указанную проблему и обеспечить значительное ускорение при решении многомерных многоэкстремальных задач. Подход представляет собой новую многоуровневую схему редукции размерности, комбинирующую кривые Пеано и классическую многошаговую схему.

Постановка задачи многомерной глобальной оптимизации

Рассмотрим многомерную задачу глобальной оптимизации:

$$\varphi(y^*) = \min\{\varphi(y) : y \in D,\ \ g_j(y) \le 0, 1 \le j \le m\},$$
$$D = \{y \in R^N : a_i \le y_i \le b_i,\ 1 \le i \le N\}. \tag{1}$$

Пусть $y(x)$, $x \in [0,1]$, есть развертка Пеано, однозначно отображающая отрезок $[0,1]$ на единичный N-мерный гиперкуб P, т.е.

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

Используя отображение $y(x)$, многомерную задачу можно свести к одномерной:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1],\ \ g_j(y(x)) \le 0,\ 1 \le j \le m\}. \tag{3}$$

Введя области

$$q_1 = [0,1],\ q_{j+1} = \{x \in q_j : g_j(y(x)) \le 0\},\ 1 \le j \le m$$

и положив

$$1 \le M = \max\{v(y(x)) : x \in [0,1]\} \le m+1,$$

получим следующее представление задачи (3):

$$g_M^* = g_M(x_M^*) = \min\{g_M(x) : x \in Q_M\} = \min\{g_M(y(x)) : x \in q_m\}.$$

При этом испытание в любой точке $x^k \in [0,1]$ позволяет определить пару

$$z^k = g_v\left(y\left(x^k\right)\right),\ v = v\left(y\left(x^k\right)\right).$$

Если $\varphi(y)$ удовлетворяет условию Липшица, то $\varphi\left(y(x)\right)$ удовлетворяет равномерному условию Гельдера:

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \le 4L\sqrt{N}(|x_1 - x_2|)^{(1/N)}. \quad (4)$$

Рассмотренная схема сведения многомерной многоэкстремальной задачи условной оптимизации к эквивалентной ей одномерной задаче позволяет применить для ее решения индексный метод [5, 6].

Индексный метод решения многомерных задач

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

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

$$0 = x_0 < x_1 < \ldots < x_k < x_{k+1} = 1, \quad (5)$$

и сопоставить их со значениями $z_i = g_v(x_i)$, $v = v(x_i)$, $1 \le i \le k$, вычисленными в этих точках; точки $x_0 = 0$ и $x_{k+1} = 1$ введены дополнительно (значения z_0 и z_{k+1} не определены) для удобства последующих обозначений.

Правило 2. Провести классификацию номеров i, $1 \le i \le k$, точек из ряда (4) по числу ограничений задачи, выполняющихся в этих точках, путем построения множеств

$$I_v = \{i : 1 \le i \le k,\ v = v(x_i)\},\ 1 \le i \le m+1,$$

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

Правило 3. Вычислить текущие нижние границы

$$\mu_v = \max\left\{\frac{|z_i - z_j|}{(x_i - x_j)^{1/N}},\ i, j \in I_v,\ i > j\right\} \quad (6)$$

для неизвестных констант Липшица L_v функций g_v, $1 \le v \le m + 1$. Если множество I_v содержит менее двух элементов или если μ_v из (5) оказывается равным нулю, то принять $\mu_v = 1$.

Правило 4. Для всех непустых множеств I_v, $1 \le v \le m + 1$, определить величины

$$z_v^* = \begin{cases} min\{z_i : i \in I_v\}, \ v = v^* \\ 0, \ v < v^*. \end{cases} \tag{7}$$

т.е. $z_v^* = 0$, если существуют точки x_i, $1 \le i \le k$, имеющие индекс, больший v.

Правило 5. Для каждого интервала (x_{i+1}, x_i), $1 \le i \le k + 1$, вычислить *характеристику* $R(i)$, где

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{r_v^2 \mu_v^2 \Delta_i} - 2\frac{(z_i + z_{i-1} - 2z_v^*)}{r_v \mu_v}, \ v = v(x_{i-1}) = v(x_i), \tag{8}$$

$$R(i) = 2\Delta_i - 4\frac{(z_i - z_v^*)}{r_v \mu_v}, \ v(x_{i-1}) < v(x_i) = v, \tag{9}$$

$$R(i) = 2\Delta_i - 4\frac{(z_{i-1} - z_v^*)}{r_v \mu_v}, \ \ v = v(x_{i-1}) > v(x_i), \tag{10}$$

$$\Delta_i = (x_i - x_{i-1})^{1/N}. \tag{11}$$

Величины $r_v > 1$, $1 \le \nu \le m + 1$, являются параметрами алгоритма. Подходящий выбор значений r_v позволяет использовать произведение r_v и μ_v как оценку константы Липшица L_v, $1 \le v \le m + 1$.

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

$$R(t) = \max\{R(i): 1 \le i \le k + 1\}. \tag{12}$$

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

$$x^{k+1} = \frac{x_t + x_{t-1}}{2}, \ v(x_{t-1}) \ne v(x_t). \tag{13}$$

В противном случае провести испытание в точке

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \text{sign}(z_t - z_{t-1})\frac{1}{2r_v}\left[\frac{|z_t - z_{t-1}|}{\mu_v}\right]^N, \ v = v(x_{t-1}) = v(x_t). \tag{14}$$

Описанные правила можно дополнить условием остановки, прекращающим испытания, если

$$x_t - x_{t-1} \le \varepsilon, \tag{15}$$

где $\varepsilon > 0$ есть заданная точность.

Многошаговая схема

Многошаговая схема редукции размерности основана на соотношении (см. [7])

$$\min\{\varphi(y) : y \in D\} = \min_{a_1 \le y_1 \le b_1} \min_{a_2 \le y_2 \le b_2} \ldots \min_{a_N \le y_N \le b_N} \varphi(y), \tag{16}$$

которое позволяет заменить решение многомерной задачи (1) решением семейства одномерных подзадач, рекурсивно связанных между собой.

Введем в рассмотрение множество функций

$$\varphi_N(y_1, \ldots, y_N) = \varphi(y_1, \ldots, y_N), \tag{17}$$

$$\varphi_i(y_1, \ldots, y_i) = \min_{a_{i+1} \le y_{i+1} \le b_{i+1}} \varphi_{i+1}(y_1, \ldots, y_i, y_{i+1}),\ 1 \le i \le N. \tag{18}$$

Тогда, в соответствии с соотношением (16), решение исходной задачи (1) сводится к решению одномерной задачи

$$\varphi_1(y_1^*) = \min\{\varphi_1(y_1) : y_1 \in [a_1, b_1]\}. \tag{19}$$

Однако при этом каждое вычисление значения одномерной функции $\varphi_1(y_1)$ в некоторой фиксированной точке предполагает решение одномерной задачи минимизации

$$\varphi_2(y_1, y_2^*) = \min\{\varphi_2(y_1, y_2) : y_2 \in [a_2, b_2]\}, \tag{20}$$

и так далее до вычисления φ_N согласно (17).

Подробные результаты последовательных и параллельных экспериментов с данной схемой редукции размерности представлены в работе [8].

Блочная многошаговая схема

Рассмотрим вектор y как вектор «агрегированных» макропеременных:

$$y = (y_1, y_2, \ldots, y_N) = (u_1, u_2, \ldots, u_M), \tag{21}$$

где i-я макропеременная представляет собой вектор размерности N_i из последовательно взятых компонент вектора y, т.е. $u_i = (y_{1_i}, y_{2_i}, \ldots, y_{N_i})$, причем $N_1 + N_2 + \ldots N_M = N$.

С использованием макропеременных основное соотношение многошаговой схемы (16) может быть переписано в виде

$$\min_{y\in D}\varphi(y)=\min_{u_1\in D_1}\min_{u_2\in D_2}\ldots\min_{u_M\in D_M}\varphi(y), \tag{22}$$

где подобласти D_i , $1 \le i \le M$, являются проекциями исходной области поиска D на подпространства, соответствующие макропеременным u_i, $1\le i\le M$.

Принципиальным отличием от исходной схемы является тот факт, что в блочной многошаговой схеме вложенные подзадачи

$$\varphi_i(u_1,\ldots,u_i)=\min_{u_{i+1}\in D_{i+1}}\varphi_{i+1}(u_1,\ldots,u_i,u_{i+1}),\ 1\le i\le M, \tag{23}$$

являются многомерными, и для их решения может быть применен способ редукции размерности на основе кривых Пеано.

Параллельная блочная многошаговая схема

Для описания процесса параллелизма многошаговой схемы (следуя [8]) введем вектор степеней распараллеливания

$$\pi=(\pi_1,\pi_2,\ldots,\pi_M), \tag{24}$$

где $\pi_i, 1\le i\le M,$ обозначает число параллельно решаемых подзадач (i + 1)-го уровня вложенности, возникающих в результате выполнения параллельных итераций на i-м уровне. Для макропеременной u_m число π_m означает количество параллельных испытаний в процессе минимизации функции $\varphi_M(u_1,\ldots,u_M)=\varphi(y_1,\ldots,y_N)$ по u_m при фиксированных значениях $u_1,\ldots,u_{m-1}$, т.е. количество параллельно вычисляемых значений целевой функции $\varphi(y)$.

Описанная схема реализована с помощью технологии *MPI*. При запуске решателя пользователь определяет схему отображения вектора y в вектор макропеременных u из (21), указывая разбиение N на $N_1 + N_2 + \ldots + N_M$, а также задает вектор π, определяя общее число *MPI*-процессов, которые должны быть созданы при запуске, и их распределение по уровням.

Распараллеливание на каждом уровне основано на модификации правил 6 и 7 индексного метода: в правиле 6 вместо выбора одного интервала с максимальной характеристикой выбирается p интервалов с наибольшими характеристиками, а в правиле 7 параллельно проводится p испытаний в точках внутри этих интервалов.

Вычислительные эксперименты

Представленные далее эксперименты выполнены на вычислительном кластере ННГУ (40 вычислительных узлов, по два двухъядерных процес-

сора *Intel Xeon 3.2 GHz* в каждом, *4Gb RAM*). Формулировки и названия четырехмерных тестовых задач взяты из работы [9].

Поскольку значение функции вычисляется быстро, то вектор распараллеливания π из (24) будем выбирать в виде (p–1,1), где p – число процессов в *MPI*-программе (последовательный запуск программы обозначен (0,0)). В табл. 1 – время решения в секундах, в табл. 2 – число вычислений значений оптимизируемой функции. Также приведены используемые параметры метода r и ε и найденное значение оптимума φ^*.

Результаты экспериментов показывают, что последовательный алгоритм с использованием одной кривой Пеано выполняет значительно меньше поисковых испытаний, чем алгоритм блочной многошаговой схемы. В результате в некоторых случаях параллельный алгоритм с использованием вектора распараллеливания (2,1) (и, соответственно, 3 *MPI*-процесса) работает примерно так же (или даже медленнее), чем последовательный. Однако уже использование вектора распараллеливания (4,1) дает значительное ускорение. Отметим также, что по времени работы и по числу выполненных поисковых испытаний при решении данных задач предложенный алгоритм не уступает другим известным методам [10].

Таблица 1

Результаты решения четырехмерных задач, время

Problem	r	ε	φ^*	(0,0)	(2,1)	(4,1)	(8,1)
Levy 1	2	0,002	0,0003	13,33	6,3	3,62	3,01
Neumaier 2	2	0,01	0,005	31,3	12,1	9,1	7,38
Neumaier 3	3.2	0,002	–15,998	27,5	21,2	15,54	11,63
Rastrigin	2.5	0,002	0,003	2,2	2,96	1,5	0,7
Rosenbrock	2.1	0,01	0,1	14,22	23,31	9,13	6,7
Shekel 5	2.5	0,002	–10,152	0,92	0,35	0,19	0,18
Shekel 7	2.5	0,002	–10,402	0,72	0,44	0,25	0,16
Shekel 10	2.5	0,002	–10,534	0,65	0,54	0,15	0,23

Таблица 2

Результаты решения четырехмерных задач, число испытаний

Problem	r	ε	φ^*	(0,0)	(2,1)	(4,1)	(8,1)
Levy 1	2	0,002	0,0003	1 378 380	2 145 661	2 021 384	2 349 743
Neumaier 2	2	0,01	0,005	3 059 379	4 870 091	6 976 209	6 252 963
Neumaier 3	3,2	0,002	–15,998	2 763 317	8 636 710	11 466 074	10 176 447
Rastrigin	2,5	0,002	0,003	235 766	1 275 904	1 286 144	999 423
Rosenbrock	2,1	0,01	0,1	1 491 435	9 309 296	7 206 413	9 499 122
Shekel 5	2,5	0,002	–10,152	95 302	122 831	105 876	127 946
Shekel 7	2,5	0,002	–10,402	75 820	146 677	135 331	131 481
Shekel 10	2,5	0,002	–10,534	64 604	181 132	71 182	154 927

Список литературы

1. Horst R., Tuy H. Global Optimization: Deterministics Approaches. – Berlin: Springer, 1996.

2. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Se-quential and parallel algorithms. – Dordrecht: Kluwer Academic Publishers, 2000.

3. Floudas C.A., Gounaris C.E. A review of recent advances in global optimization // Journal of Global Optimization. – 2009. – Vol. 45. – P. 3–38.

4. Grishagin V.A., Sergeyev Ya.D., Strongin R.G. Parallel Characteristical Algorithms for Solving Problems of Global Optimization // Journal of Global Optimization. – 1997. – Vol. 10. – P. 185–206.

5. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. – М.: Наука, 1978.

6. Стронгин Р.Г. Поиск глобального оптимума. – М.: Знание, 1990.

7. Carr C.R., Howe C.W. Quantitative Decision Procedures in Management and Eco-nomic. Deterministic Theory and Applications. – New York: McGraw-Hill, 1964.

8. Sergeyev Ya.D., Grishagin V.A. Parallel Asynchronous Global Search and the Nested Optimization Scheme // Journal of Computational Analysis and Applications. – 2001. – Vol. 3(2). – P. 123–245.

9. Ali M., Khompatraporn Ch., Zabinsky Z.B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems // Journal of Global Optimization. – 2005. – Vol. 31. – P. 635–672.

10. Paulavicius R., Zilinskas J., Grothey A. Parallel branch and bound for global opti-mization with combination of Lipschitz bounds // Optimization Methods & Software. – 2011. – Vol. 26(3). – P. 487–498.

М.А. Ташкинов

Пермский национальный исследовательский политехнический университет

ПРИМЕНЕНИЕ СИСТЕМЫ WOLFRAM MATHEMATICA ДЛЯ ВЫЧИСЛЕНИЯ МНОГОМЕРНЫХ ИНТЕГРАЛОВ В ЗАДАЧАХ СТОХАСТИЧЕСКОЙ МЕХАНИКИ

Рассмотрены методики вычисления многомерных сингулярных интегралов для определения статистических характеристик полей напряжений и деформаций в компонентах неоднородных материалов на основе решения стохастических краевых задач с помощью метода функций Грина.

Введение

При изучении современных функциональных материалов важной задачей является создание точных математических моделей, определяющих их поведение и свойства. Для расчета напряженно-деформированного состояния узлов и конструкций из них традиционно используются методы конечных элементов и их вариации. При этом, как правило, деталям конструкции присваиваются эффективные механические характеристики, которые могут быть получены либо из экспериментов, либо с помощью моделирования. В отдельный класс можно выделить материалы с неоднородной микроструктурой. Для них, в отличие от однородных материалов, существует отдельная задача изучения влияния внутренних морфологических параметров структуры на микроуровне, которые могут оказывать значительное влияние на гомогенизированные характеристики материала.

Один из способов математического описания подобных материалов состоит в использовании структурно-феноменологический подхода [1, 2], особенность которого в том, что однородные физико-механические свойства элементов структуры задаются с помощью общепринятых в механике феноменологических уравнений и критериев, а характеристики структурных полей деформирования и эффективные свойства вычисляются, например, из решений краевых задач, которые выражаются с помощью метода функций Грина [3, 4]. В данной работе исследовано применение такого подхода для изучения свойств матричных двухкомпонентных композитов. Оно связано с методами стохастической механики, основанными на теории случайных функций и позволяющими учитывать такой важный фактор реальной структуры композитов, как статистический разброс формы, размеров и физико-механических свойств элементов структуры.

Возможность влиять на физические, механические и геометрические свойства составных компонент неоднородных материалов позволяет создавать новые материалы с заранее заданным комплексом характеристик. При этом математическое моделирование используется не только для определения эффективных свойств, но и для изучения поведения каждой фазы в отдельности, что невозможно при экспериментальном исследовании образцов. Методы стохастической механики позволяют определять статистические характеристики полей структурных напряжений и деформаций в каждом компоненте композита на основе моментных функций и приближенных решений стохастических краевых задач сводящихся к суперпозиции многомерных интегралов.

Стохастическая краевая задача и статистические характеристики

Итак, характеристики структурных полей деформирования микронеоднородных сред могут быть определены из решения стохастических краевых задач, в которых уравнения и граничные условия содержат случайные величины. Поля напряжений, деформаций, перемещений, структурных модулей упругости, а также поле значений индикаторной функции могут быть выражены в виде суммы средней составляющей и пульсации. Например, поля перемещений принимают следующий вид:

$$u_m(\vec{r}) = \langle u_m(\vec{r}) \rangle + u'_m(\vec{r}), \qquad (1)$$

где $\vec{r}$ – радиус-вектор с компонентами; скобки $\langle\ \rangle$ обозначают осреднение (по представительному объему материала; апостроф ' – флуктуацию, или отклонение от среднего.

Решение стохастической краевой задачи записывается в виде рекуррентной последовательности:

$$\frac{\partial u_i'^{(\chi)}(\vec{r})}{\partial x_j} = \int_{V_1} G_{im,j}(\vec{r},\vec{r}_1)\left[C'_{mnkl}(\vec{r}_1) e_{kl} + C'_{mnkl}(\vec{r}_1) u_{k,l}'^{(\chi-1)}(\vec{r}_1) \right]_{,1n} \mathrm{d}V_1, \qquad (2)$$

где χ – приближение, в котором решается задача; $C_{ijkl}(\vec{r})$ – тензор структурных модулей упругости, $G_{im}(\vec{r},\vec{r}_1)$ – функция Грина.

Для двухкомпонентных композиционных материалов статистические характеристики в виде моментов первого (средних значений) и второго порядка (дисперсий) в матрице M и включениях I в общем виде выражаются через решение краевой задачи (2). Точность характеристик зависит от приближения решения. Показано [3, 4], что для получения численного значения искомых статистических характеристик необходимо вычислить значения интегралов, входящих в формулы для безусловных моментов $\langle \varepsilon'_{ij}(\vec{r}) \varepsilon'_{\alpha\beta}(\vec{r}) \rangle$, $\langle \lambda'(\vec{r}) \varepsilon'_{ij}(\vec{r}) \rangle$, $\langle \lambda'(\vec{r}) \varepsilon'_{ij}(\vec{r}) \varepsilon'_{\alpha\beta}(\vec{r}) \rangle$, где $\varepsilon'_{ij}(\vec{r}) = \frac{1}{2}\left(u'_{i,j}(\vec{r}) + u'_{j,i}(\vec{r}) \right)$. Так, например, момент $\langle \lambda'(\vec{r}) \varepsilon'_{ij}(\vec{r}) \rangle$ принимает следующий вид:

$$\langle \lambda'(\vec{r}) \varepsilon'_{ij}(\vec{r}) \rangle^{(1)} = \frac{1}{2} e_{kl} \overline{C}_{mnkl} \int_{V_1} \left(G_{im,j}(\vec{r},\vec{r}_1) + G_{jm,i}(\vec{r},\vec{r}_1) \right) \frac{\partial K_\lambda^{(2)}(\vec{r},\vec{r}_1)}{\partial x_{1n}} \mathrm{d}V_1, \qquad (3)$$

где $K_\lambda^{(2)}(\vec{r},\vec{r}_1)$ – структурная корреляционная функция (моментная функция второго порядка), получаемая для моделей или реальных образцов исследуемых материалов и характеризующая их геометрию на микроуровне.

В выражениях для прочих моментов также присутствуют моментные функции более высоких порядков.

Сложность численных расчетов определяется порядком приближения решения краевой задачи и методикой его получения. Существует две методики получения формул для искомых моментов с использованием решений краевой задачи. В первом случае безусловные моменты записываются с использованием решения задачи в том виде, в котором оно записано в выражении (2). При этом под интегралом остаются первые производные функции Грина и производные моментных функций. Такой методикой получают решения в первых производных функции Грина.

Во втором случае используется операция свертки Стилтьеса [3], при этом в подынтегральном выражении остаются вторые производные функции Грина и моментные функции без производной. В таком случае говорят о решении во вторых производных функции Грина.

Функция Грина и ее производные имеют особенность в точках, когда знаменатель функции обращается в ноль, поэтому при вычислении значений моментов $\left\langle \varepsilon'_{ij}(\vec{r})\varepsilon'_{\alpha\beta}(\vec{r})\right\rangle$, $\left\langle \lambda'(\vec{r})\varepsilon'_{ij}(\vec{r})\right\rangle$, $\left\langle \lambda'(\vec{r})\varepsilon'_{ij}(\vec{r})\varepsilon'_{\alpha\beta}(\vec{r})\right\rangle$ применять напрямую численные методы нельзя. При интегрировании выражений, содержащих первые производные функции Грина, используются специальные приемы устранения сингулярности. Для интегралов со второй производной функции Грина можно выделить формальную и сингулярную части, что избавит от необходимости интегрирования по сингулярной области.

При расчетах в первых производных функции Грина, в первом приближении решения задачи (2), выражения для безусловных моментов содержат три типа интегралов различной кратности, отличающихся друг от друга подынтегральным выражением, в которое входят производные функции Грина и аппроксимирующие выражения для моментных функций. Во втором приближении количество интегралов увеличивается до восьми.

При решении во вторых производных функции Грина первое приближение также требует вычисления трех интегралов, однако во втором приближении их число возрастает до 18 за счет слагаемых, представляющих собой перемножение формальной и сингулярных составляющих.

Интегрирование по объему всего тела в формулах, содержащих функцию Грина, можно заменить интегрированием по области статистической зависимости случайного поля структурных модулей упругости, т.е. области, где значения корреляционной функций отличны от нуля. Таким образом, верхний предел интегрирования равен радиусу статистической зависимости.

Интегралы, содержащие функции Грина и моментные функции, в сокращенной записи имеют следующий вид:

$$I_{1,2,11,21}^{(\vec{r},\vec{r}_1,\vec{r}_2,\vec{r}_{11},\vec{r}_{21})} = D_{\lambda}^{(5)} \int\limits_{V_1}\int\limits_{V_2}\int\limits_{V_{11}}\int\limits_{V_{21}} \left(G_{im,jn}^{(f)}(\vec{r},\vec{r}_1) + G_{jm,in}^{(f)}(\vec{r},\vec{r}_1)\right)\left(G_{\alpha\gamma,\beta\eta}^{(f)}(\vec{r},\vec{r}_2) + G_{\beta\gamma,\alpha\eta}^{(f)}(\vec{r},\vec{r}_2)\right)\times$$

$$\times G_{kf,ls}^{(f)}(\vec{r}_x,\vec{r}_{11}) G_{\varphi v,hw}^{(f)}(\vec{r}_x,\vec{r}_{21}) f_{\lambda}^{(5)}\left(\vec{r},\vec{r}_1,\vec{r}_2,\vec{r}_{11},\vec{r}_{21}\right) \mathrm{d}V_{21}\mathrm{d}V_{11}\mathrm{d}V_2\mathrm{d}V_1, \qquad (4)$$

где нижние индексы в скобках обозначают наличие в интеграле соответствующих множителей, содержащих функцию Грина, верхние индексы определяют порядок и аргумент моментной функции. Индекс x радиус-вектора однозначно определяется подынтегральными множителями и правилами свертки Дирака. Кратность интеграла определяется количеством независимых переменных $\vec{r}$. Отсутствие переменных $\vec{r}$ в скобках означает, что в подынтегральном выражении нет моментной функции.

Методы вычисления многомерных интегралов в Wolfram Mathematica

Для вычисления статистических характеристик и входящих в них интегралов использовалась вычислительная среда *Wolfram Mathematica*, которая предлагает ряд различных методов численного интегрирования. Каждый метод определяется стратегией и правилом интегрирования. Комбинируя стратегии и правила, можно задавать методы, наиболее подходящие для данного подынтегрального выражения. Разделяют адаптивные и неадаптивные стратегии интегрирования. Адаптивные стратегии пытаются найти проблемные области интегрирования и концентрируют вычислительные усилия (разбиение на подобласти) в них. Неадаптивные стратегии увеличивают число элементов разбиения одинаково во всей области интегрирования. Адаптивные стратегии интегрирования включают в себя следующие компоненты: правило интегрирования для вычисления значения интеграла и погрешности в области; метод, определяющий разбиение области на множество подобластей; критерий завершения процедуры интегрирования.

В свою очередь, адаптивные стратегии разделяются на глобальные и локальные. Глобальная адаптивная стратегия основана на том, что из всех подобластей исходной области интегрирования выбирается подобласть с наибольшей погрешностью и делится пополам. Затем вычисляются значение интеграла и погрешность для каждой половины и процедура повторяется для всего множества подобластей. После каждого такого шага пересчитываются глобальное значение интеграла и глобальная погрешность, которые представляют собой сумму значения интеграла и погрешности

в каждой подобласти. Локальная адаптивная стратегия имеет исходную и рекурсивную процедуру. Исходная процедура вычисляет погрешность в подобластях, полученных разбиением области интегрирования на первом шаге. Рекурсивная процедура вычисляет значение интеграла и погрешность в каждой подобласти, используя заданное правило интегрирования. Если погрешность в подобластях значительно больше, чем исходная погрешность, принимается решение о продолжении рекурсивной процедуры, но только для данной подобласти. Погрешность для каждой из начальных подобластей определяется как сумма погрешностей, получаемых при реализации рекурсивной процедуры в данной подобласти.

Для вычисления интегралов, входящих в выражения для безусловных моментов и содержащих первую производную функции Грина или формальную часть второй производной функции Грина, в данной работе исследовалась возможность применения комбинации глобальной или локальной адаптивной стратегии в совокупности с трапецеидальным или многомерным правилом интегрирования. Формальная область интегрирования при вычислении интегралов со второй производной функции Грина задавалась путем исключения из области интегрирования точек, в которых подынтегральное выражение имеет особенность.

Первые производные функции Грина также имеют область сингулярности. Для интегрирования выражений с первыми производными применяется преобразование координат Даффи, принцип которого состоит в том, что интеграл по квадрату, кубу или гиперкубу с сингулярностью в одном из углов заменяется интегралом с сингулярностью вдоль линии.

Результаты и анализ

В таблице представлены полученные статистические характеристики полей деформирования, рассчитанные на основе второго приближения решения стохастической краевой задачи упругости композитов с использованием методики в первых производных функции Грина при всестороннем растяжении для представительного объема пористого керамического материала с объемной долей 15, 20, 25 и 30 %. Упругие свойства матрицы заданы через модуль Юнга и коэффициент Пуассона следующим образом: $E_M = 2 \cdot 10^{11}$ Па, $\nu_M = 0{,}3$.

Вычисление интегралов, содержащих первые или вторые производные функций Грина и моментные функции высших порядков, достаточно трудоемко. Для ускорения процесса счета в среде *Wolfram Mathematica* были применены алгоритмы параллельных вычислений. Вычисления проводились на 60 четырехядерных процессорах *AMD Barcelona-3*. Это позво-

ляет улучшить производительность расчетов в 10–15 раз по сравнению с использованием обычного персонального компьютера со средней современной конфигурацией.

Характеристики полей деформирования в пористом материале при всестороннем растяжении для структур с различной объемной долей

Статистические характеристики	15 %	20 %	25 %	30 %
Средние деформации, $\times 10^{-6}$ $\langle\varepsilon_{11}\rangle_M = \langle\varepsilon_{22}\rangle_M = \langle\varepsilon_{33}\rangle_M$	0,986 64	0,981 26	0,975 33	0,968 33
$\langle\varepsilon_{12}\rangle_M = \langle\varepsilon_{13}\rangle_M = \langle\varepsilon_{23}\rangle_M$	0	0	0	0
Средние напряжения, МПа $\langle\sigma_{11}\rangle_M = \langle\sigma_{22}\rangle_M = \langle\sigma_{33}\rangle_M$	0,388 43	0,49063	0,324 26	0,294 71
$\langle\sigma_{12}\rangle_M = \langle\sigma_{13}\rangle_M = \langle\sigma_{23}\rangle_M$	0	0	0	0
Дисперсии деформаций, $\times 10^{-12}$, в матрице $\langle\varepsilon'_{11}\varepsilon'_{11}\rangle_M = \langle\varepsilon'_{22}\varepsilon'_{22}\rangle_M = \langle\varepsilon'_{33}\varepsilon'_{33}\rangle_M$	0,032 03	0,040 29	0,050 89	0,069 07
$\langle\varepsilon'_{12}\varepsilon'_{12}\rangle_M = \langle\varepsilon'_{13}\varepsilon'_{13}\rangle_M = \langle\varepsilon'_{23}\varepsilon'_{23}\rangle_M$	0,235 49	0,306 62	0,382 46	0,500 65
Дисперсии напряжений, МПа^2, в матрице $\langle\sigma'_{11}\sigma'_{11}\rangle_M = \langle\sigma'_{22}\sigma'_{22}\rangle_M = \langle\sigma'_{33}\sigma'_{33}\rangle_M$	0,007 30	0,014 26	0,022 67	0,069 07
$\langle\sigma'_{12}\sigma'_{12}\rangle_M = \langle\sigma'_{13}\sigma'_{13}\rangle_M = \langle\sigma'_{23}\sigma'_{23}\rangle_M$	0,002 74	0,002 63	0,002 29	0,001 93

При использовании глобальной адаптивной стратегии наблюдалась медленная либо осциллирующая сходимость интегралов, при этом увеличение количества разбиений интегрируемой области не приводило к улучшению сходимости. Локальные адаптивные стратегии позволяли добиться лучшей сходимости интеграла, однако при использовании трапецеидального правила интегрирования интегралы сходились медленнее, чем при многомерном правиле, результат при этом в пределах заданной погрешности был равным. Таким образом, для класса подынтегральных выражений, содержащих производные функции Грина с сингулярностью и моментные функции, был выбран метод, комбинирующий локальную адаптивную стратегию и многомерное правило интегрирования.

Таким образом, разработанные в системе *Wolfram Mathematica* новые автоматизированные алгоритмы позволяют выполнять полный цикл необходимых вычислений для предложенной методологии решения задач стохастической механики, от синтеза структур и построения моментных функций для них до расчета статистических характеристик полей напряжений и деформаций в различных фазах композитов. Примененные методы параллельных вычислений значительно сократили время вычислений.

Разработанные методы моделирования микроструктуры и решений краевых задач стохастической механики могут быть использованы для сравнительного анализа влияния различных структурных параметров на статистические характеристики полей напряжений с целью создания материалов с заранее определенным набором свойств и оценки вероятности их разрушения.

Список литературы

1. Вильдеман В.Э., Соколкин Ю.В., Ташкинов А.А. Механика неупругого деформирования и разрушения композиционных материалов. – М.: Наука, 1997. – 288 с.

2. Волков С.Д., Ставров В.П. Статистическая механика композитных материалов. – Минск: Изд-во Белорус. гос. ун-та, 1978. – 208 с.

3. Tashkinov M.A., Wildemann V.E., Mikhailova N.V. Method of successive approximations in stochastic elastic boundary value problem for structurally heterogenous materials // Computational Materials Science. – Vol. 52, iss. 1. – P. 101–106. DOI: 10.1016/j.commatsci.2011.04.025

4. Tashkinov M. Statistical characteristics of structural stochastic stress and strain fields in polydisperse heterogeneous solid media // Computational Materials Science. – 2014. – Vol. 94. – P. 44–50. DOI: 10.1016/j.commatsci.2014.01.050

Д.И. Федоровцев, П.В. Писарев, А.А. Брюханова

Пермский национальный исследовательский политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ТЕПЛООБМЕННЫХ ПРОЦЕССОВ В ТЕРМОЭЛЕКТРИЧЕСКОМ ПРЕОБРАЗОВАТЕЛЕ ЭНЕРГИИ С ИСПОЛЬЗОВАНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА ПНИПУ

В рамках исследований был предложен альтернативный вариант генератора электрической энергии для ЭХЗ трубопроводов ОАО «Газпром» с минимальной стоимостью обслуживания, при неизменных выходных характеристиках, на основе элемента Пельтье.

Основной целью работы является численное моделирование теплообменных процессов в предложенном генераторе энергии и его последующая модификация.

Эксплуатация современных подземных сооружений и конструкций связана с интенсивными динамическими нагрузками и неблагоприятными условиями их эксплуатации. Современных изоляционных материалов не-

достаточно ввиду их скорого изнашивания. В тех местах, которые подверглись деформации, начинается электрохимическая коррозия. С данной проблемой можно справиться путем разработки инновационных высокотехнологичных средств защиты на базе электрохимзащиты (ЭХЗ). Разработка комплекса мер существенно продлит сроки эксплуатации и снизит расходы на проведение обслуживания и ремонтных мероприятий.

Для достижения поставленной цели были решены следующие вопросы: конструирование корпуса камеры термопреобразования и элементов конструкции, разработка методики численного расчета [1, 2] и численное моделирование отвода тепла в термоэлектрическом преобразователе энергии в связанной постановке. Численный эксперимент проводился с использованием многопроцессорного вычислительного комплекса. В качестве аппаратного обеспечения для вычислительных экспериментов использовались ресурсы высокопроизводительного вычислительного комплекса Центра высокопроизводительных вычислительных систем, имеющего следующие технические характеристики:

- 88 вычислительных узлов;
- 128 четырехядерных процессоров *Barcelona-3* (всего 512 ядер);
- 48 восьмиядерных процессоров *Intel Xeon E5-2680* (всего 384 ядра);
- пиковая производительность 20 Тфлопс;
- производительность в тестовом пакете *Linpack* 78 %;
- объем системы хранения информации 12 Тбайт;
- объем оперативной памяти 2816 Гбайт (32 Гбайт/узел);
- *Windows HPC Server 2008* в качестве исполнительной среды.

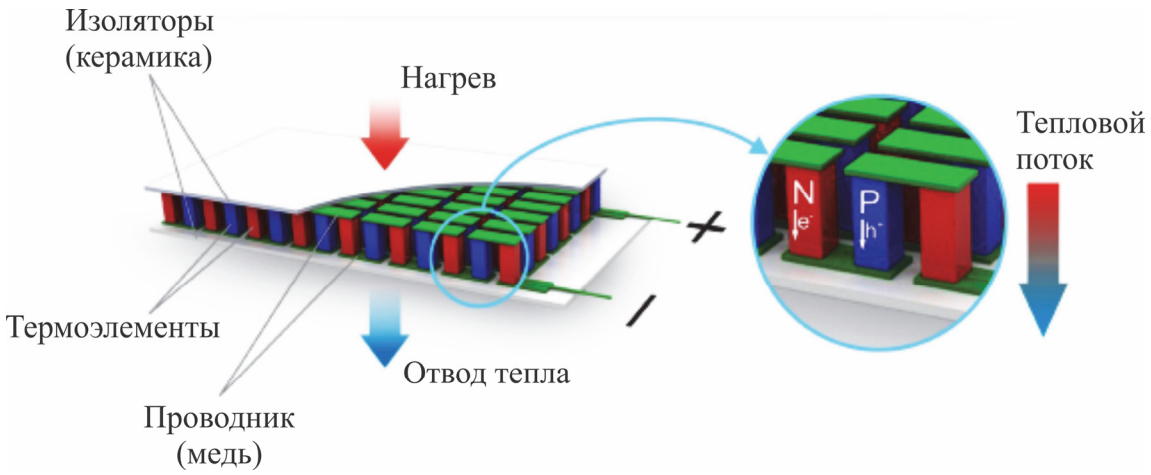


Рис. 1. Устройство и принцип работы термоэлектрического модуля

При обеспечении разности температур на сторонах модуля – одна сторона холодная (охлаждается), а другая горячая (нагревается) – в полупроводниках происходит *p–n*-переход, возникает разность потенциалов.

В момент замыкания термоэлементов на внешнюю нагрузку в цепи возникает постоянный электрический ток. Величина тока будет пропорциональна разности температур. Таким образом происходит преобразование тепла в электрическую энергию.

Корпус конструкции представляет полую призму, на передней стенке расположено отверстие для крепления горелки, на задней поверхности отверстие для выхода продуктов сгорания. Диаметр входного отверстия значительно больше диаметра выходного отверстия. Это условие является необходимым условием оптимального нагрева корпуса.

Ширина каждой грани и длина корпуса пропорциональна габаритам термоэлектрического генераторного модуля. При использовании среднетемпературного ТГМ серии *MARS* с габаритами 260×92×30 мм рационально использовать горячий корпус с десятью крепежными гранями. Длина корпуса 2000 мм, диаметр 880 мм, ширина каждой грани 280 мм. На гранях корпуса расположены резьбовые шпильки для крепления ТГМ и радиаторов охлаждения.

Данная конструкция обоснована простотой изготовления. Заготовкой служит стальной лист 2000×6000 мм, толщиной 5 мм. Одного листа достаточно для изготовления двух корпусов. В процессе изготовления применяются сварочные, гибочные, сверлильные работы. Шпильки из нержавеющей стали вворачиваются в отверстия, расположенные на гранях корпуса.

Рассматривается следующая задача. Имеется твердотельная полая конструкция с входным и выходным отверстиями. На гранях конструкции расположены термоэлектрические модули и радиаторы охлаждения. Во входное отверстие подаются продукты сгорания газа. Охлаждение радиаторов – пассивное, без нагнетания холодного воздуха.

Для проведения численного моделирования отвода тепла в термоэлектрическом преобразователе энергии использовались следующие исходные данные: материал полой конструкции – сталь 20; материал термоэлектрических модулей – висмут; материал радиаторов охлаждения – алюминий; температура окружающей среды – 293 К; температура продуктов сгорания – 1073 К; скорость входа продуктов сгорания – 2 м/с.

При моделировании принимались следующие допущения:

– рабочей температурой устройства является температура пластины-держателя в реперной точке;

– все материалы элементов конструкции однородны по составу и изотропны по свойствам;

– контакты между элементами идеальны;

– боковые поверхности теплоизолированы [3].

Уравнение нестационарной теплопроводности для теплового блока записывается следующим образом:

$$\frac{\partial}{\partial\tau}\left(\rho_i \cdot c_{pi} \cdot T_i\right) - \mathrm{div}\left(\lambda_i \cdot gradT\right) = S_{Ti}, \tag{1}$$

где τ – время, с; ρ_i – плотность, кг/м^3; c_{pi} – теплоемкость, Дж/(кг·К); λ_i – коэффициент теплопроводности, Вт/(м·К); T – температура, $T = T\left(\Omega_i, \tau\right)$, К; S_{Ti} – источниковый член, учитывающий источники и стоки теплоты, Вт/м^3.

Начальные условия:

– температура $T|_{t=0} = 293$ К;

– давление $P|_{t=0} = 0{,}1$ МПа;

– скорость $V|_{t=0} = 0$ м/с.

1. Вход:

$$\rho V\Big|_{\text{гр}} = \rho V_n\Big|_{\text{гр}} = v;\ \rho V_\tau\Big|_{\text{гр}} = 0;\ T\Big|_{\text{гр}} = \text{const}. \tag{2}$$

2. Выход:

$$P\Big|_{\text{гр}} = P\Big|_{t=0};\ \frac{\partial T}{\partial n}\Big|_{\text{гр}} = 0;$$

$$V\Big|_{\text{гр}} \le 0, \text{ то } \frac{\partial V}{\partial n}\Big|_{\text{гр}} = 0;\ V\Big|_{\text{гр}} > 0, \text{ то } V\Big|_{\text{гр}} = V_\tau\Big|_{\text{гр}}. \tag{3}$$

3. Стенка:

$$\bar{V}_n\Big|_{\text{гр}} = 0;\ \frac{\partial V_\tau}{\partial n}\Big|_{\text{гр}} = \frac{u_\tau}{\kappa}\ln\left(E\frac{\rho u_\tau y}{\mu}\right);\ \frac{\partial T}{\partial n}\Big|_{\text{гр}} = 0. \tag{4}$$

Движение газа описывается уравнением Навье – Стокса для сжимаемой жидкости:

$$\rho\left(\frac{\partial v_i}{\partial t} + v_k\frac{\partial v_i}{\partial x_k}\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_k}\left(\mu\left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3}\delta_{i,k}\frac{\partial v_l}{\partial x_l}\right)\right) + \frac{\partial}{\partial x_k}\left(\zeta\frac{\partial v_l}{\partial x_l}\delta_{i,k}\right),$$
$$\frac{\partial\rho}{\partial t} + \nabla\cdot\left(\rho\vec{v}\right) = 0, \tag{5}$$

где ρ – плотность; p – давление; μ – коэффициент динамической вязкости (сдвиговая вязкость); ζ – объемная вязкость; $\delta_{i,k}$ – дельта Кронекера.

Ввиду симметрии геометрии расчет проводился на отдельном сегменте (рис. 2), что позволило сократить время расчета и сэкономить вычислительные ресурсы.

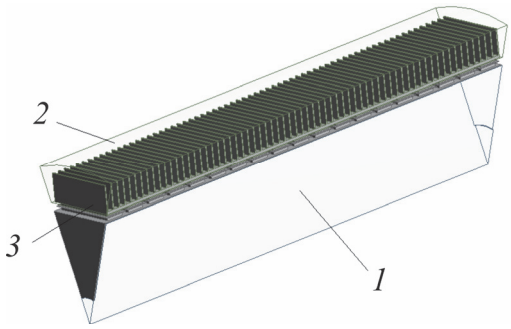


Рис. 2. Геометрическая модель: *1*, *2* – области
для газодинамического расчета; *3* – область для расчета
теплообмена в твердотельной конструкции

Задача решалась в связанной постановке, поэтому сегмент необходимо разбить на три области: область газодинамического расчета (истечение продуктов сгорания) и область расчета конвективного охлаждения радиаторов, расчет теплопроводности в термоэлектропреобразователе-радиаторе – твердотельная конструкция.

Для объединения решения газодинамического расчета и расчета теплообмена задан интерфейс на сопряженных сторонах расчетных областей.

В качестве элемента расчетной сетки был выбран тетраэдр. Минимальный размер элемента – 1 мм, максимальный – 10 мм.

В результате расчета были получены поля температуры, давления и скорости в различных плоскостях и сечениях. Распределение поля температуры в термоэлектрических модулях и радиаторе охлаждения представлено на рис. 3. Значения температуры возрастают в направлении течения продуктов сгорания. Максимальное значение составило 936 К, минимальное – 839 К.

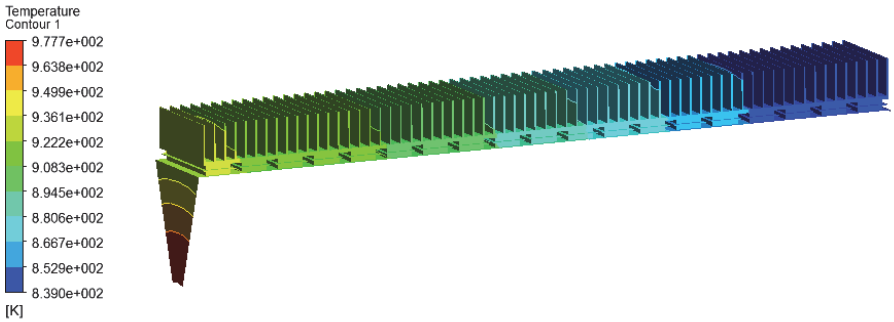


Рис. 3. Распределение поля температуры по поверхности
термоэлектрических модулей и радиатора охлаждения

Температура продуктов сгорания внутри полой конструкции (рис. 4) понижается по мере отдаления от центра к стенкам. Максимальное значение – 1084 К, минимальное – 840 К.

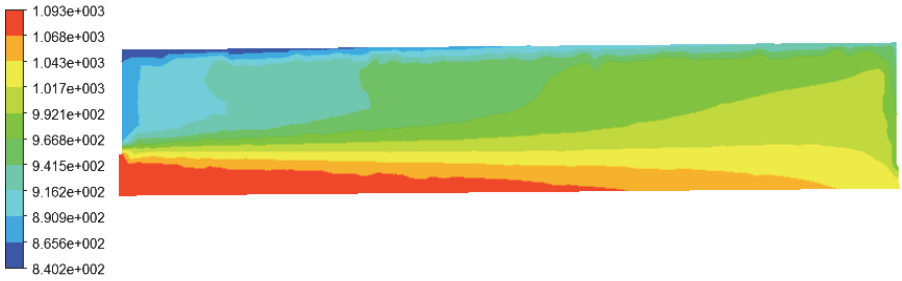


Рис. 4. Температура продуктов сгорания

Давление на стенках полой конструкции (рис. 5) распределено величиной 3,838 атм.

Рис. 5. Давление на стенках полой конструкции

Давление внутри полой конструкции (рис. 6) понижается по мере отдаления от центра к стенкам. Максимальное значение – 4,328 атм, минимальное – 3,692 атм.

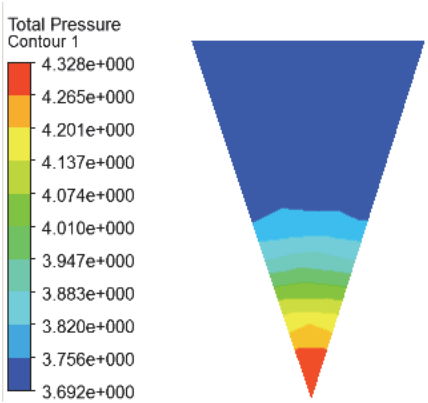


Рис. 6. Давление внутри полой конструкции

Распределение скоростей продуктов сгорания внутри полой конструкции представлено на рис. 7. Максимальное значение – 4,654 м/с, минимальное – 0.

Результаты полученных численных расчетов подтверждают физичность и адекватность модели. Температура на горячей стенке равна 887,2 К и близка к оптимальной (773 К), однако температура на охлаждающих ра-

диаторах слишком высока – 839 К (при оптимальной 400 К). При этом температура возрастает в направлении течения продуктов сгорания. Данные значения температур обусловливаются режимами подачи продуктов сгорания, интенсивностью системы охлаждения радиаторов, а также образованием избыточного давления на выходе из горячего корпуса.

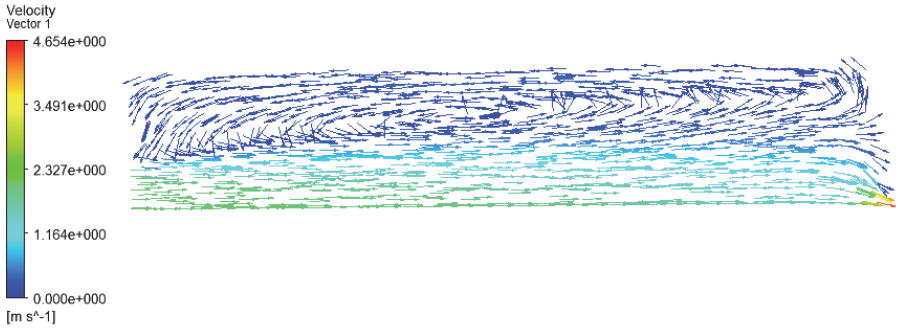


Рис. 7. Распределение скоростей

В целях получения значений температур близких к оптимальным в дальнейшем планируется провести численный эксперимент с уточненными режимами подачи продуктов сгорания.

Список литературы

1. Численное моделирование взаимодействия в динамической системе «газ – конструкция» / П.В. Писарев, Д.В. Зимин, В.Я. Модорский, Ю.В. Соколкин // Механика и процессы управления: материалы XXXXI Всерос. симпоз. / Рос. акад. наук. – М., 2011. – Т. 1. – С. 127.

2. Писарев П.В., Модорский В.Я., Зимин Д.В. Особенности разработки параллельных приложений по расчету газодинамических характеристик энергетических установок на ВВК ПГТУ// Аэрокосмическая техника, высокие технологии и инновации – 2009: XII Всерос. науч.-техн. конф. – Пермь: Изд-во Перм. гос. техн. ун-та, 2009. – С. 226.

3. Писарев П.В., Модорский В.Я., Соколкин Ю.В. Решение междисциплинарных задач на примере взаимодействия струи горячего газа с металлической преградой // Механика и процессы управления: материалы XXXXI Всерос. симпоз. / Рос. акад. наук. – М., 2011. – Т. 1. – С. 123.

Ю.В. Халевицкий, А.В. Коновалов

Институт машиноведения УрО РАН, г. Екатеринбург

ЭФФЕКТИВНЫЕ СТРУКТУРЫ ДАННЫХ ДЛЯ РЕШЕНИЯ УПРУГОПЛАСТИЧЕСКИХ ЗАДАЧ НА ГЕТЕРОГЕННЫХ КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЯХ

Описаны подходы к хранению данных и обмену ими в ходе решения упругопластических задач на гетерогенных кластерных системах методом конечных элементов. Предложены способы распределенного хранения вектора, а также матрицы жесткости на различных этапах решения задачи.

Введение

Упругопластические задачи возникают при моделировании процессов пластической деформации. Использование метода конечных элементов применительно к данному классу задач порождает систему линейных алгебраических уравнений (СЛАУ) большой размерности [1]. Матрица жесткости системы имеет ленточную структуру и значительно разрежена внутри ленты.

В силу большой ширины ленты и разреженности матрицы использование прямых методов решения СЛАУ требует значительных затрат памяти вычислителя и не позволяет решать задачи с большими разбиениями сетки. Применение итерационных методов позволяет не хранить нулевые элементы, находящиеся внутри ленты матрицы, что сокращает требования к памяти вычислительной системы.

Реализации методов решения СЛАУ конструируются из базовых подпрограмм линейной алгебры для проведения вычислений с векторами и матрицами. Наиболее требовательными к вычислительным ресурсам являются подпрограммы, оперирующие матрицами. Большинство современных итерационных методов используют лишь одну подпрограмму, задействующую матрицу, умножение матрицы на вектор.

Высокая скорость работы подпрограммы достигается не за счет оптимизации математической операции, так как все вычислители выполняют одни и те же арифметические действия, а с помощью спланированной с учетом особенностей вычислителя техники хранения и перемещения информации в памяти, т.е. при помощи эффективных структур данных.

Этапы решения упругопластической задачи методом конечных элементов

Решение упругопластической задачи с помощью метода конечных элементов происходит итерационно. Изменения состояния деформируемой

среды моделируются последовательными шагами по времени. На каждом шаге по времени решение задачи происходит в три этапа:

1. На первом этапе происходит расчет локальных матриц жесткости и локальных векторов правой части для каждого из конечных элементов. Строится матрица жесткости A, и к ней применяются граничные условия. Матрица жесткости A получается суммированием расширенных локальных матриц жесткости элементов. Локальные матрицы жесткости имеют размерность, равную произведению количества переменных в узле на количество узлов. Это обусловливает блочную структуру матрицы жесткости. Матрица жесткости хранится в памяти вычислительной системы в сжатом и распределенном виде, который обсуждается ниже.

2. На втором этапе решается система линейных алгебраических уравнений $Ax = b$. Решение системы происходит относительно x. В случае, если используется итерационный метод решения СЛАУ, происходит последовательное вычисление произведения матрицы A на различные векторы, природа которых зависит от конкретного итерационного метода. Данные векторы хранятся в памяти вычислительной системы в сжатом и распределенном виде, который обсуждается ниже

3. На третьем этапе напряженно-деформированное состояние конечных элементов пересчитывается на основании полученных значений x.

Каждый шаг по времени подразумевает однократное выполнение первого этапа вычислений и многократное (10–15 раз) выполнение второго и третьего этапа нагружения для удовлетворения условия текучести. При этом матрица, полученная на первом этапе, может быть использована повторно.

Логическая организация вычислителей в процессе решения задачи

Программное обеспечение кластерных суперкомпьютеров позволяет выделять ресурсы в виде сущностей, называемых процессами *MPI*. Каждому процессу *MPI* ставится в соответствие некоторое количество ресурсов, расположенных в пределах одного узла: ядер процессора и вычислительных ускорителей. При этом пользователь определяет количество процессов, а также количество выделяемых для каждого процесса ресурсов. Одно и то же количество ресурсов можно распределить различными способами, например создавая небольшое количество процессов *MPI*, управляющих значительным количеством вычислительных ресурсов, или создавая отдельный процесс *MPI* для каждого из ускорителей.

Создание процессов *MPI*, управляющих несколькими ускорителями, позволяет органично использовать фрагменты вектора, необходимые для работы множества ускорителей. Такой подход требует решения значитель-

ного круга задач, связанных с организацией синхронизации и взаимодействия между потоками. Учитывая, что использование процессов с несколькими ускорителями позволяет избежать лишь операций копирования порций вектора в памяти узла, время выполнения которых значительно меньше времени операций передачи данных между ускорителями и узлами, создание таких процессов для решения упругопластических задач выглядит нецелесообразным, а при использовании прямого обмена данными между ускорителями – гарантированно контрпродуктивным.

Учитывая вышесказанное, рационально создавать один процесс *MPI* для пары из одного вычислительного ядра и ускорителя. При этом вычисления, связанные с векторной алгеброй, выполняются на ускорителе, а ядро центрального процессора задействуется при сборке матрицы жесткости, интегрировании и управлении обменом данными.

Естественным способом распределения информации о матрице между процессами является хранение «слоями». При этом каждый процесс *MPI* сопоставляется с множеством последовательных строк.

Хранение матрицы жесткости во время сборки

На разных этапах решения задачи возникают разные требования к хранению матрицы жесткости. На первом этапе «структура разреженности» (*sparsity pattern*) матрицы неизвестна, поэтому необходимо эффективно динамически выделять память для поступающей информации об элементах матрицы. В данный момент авторам известны два подхода к хранению матрицы жесткости на этапе сборки.

Первый подход, предложенный в работе [2], предлагает хранить матрицу в координатном формате, упакованном в двоичное дерево поиска. При этом для сравнения элементов матрицы используется составной предикат. Это позволяет использовать для хранения матрицы готовые структуры, такие как std::map из стандартной библиотеки *C++*. К достоинствам данного метода можно отнести простоту реализации, простоту построения матрицы в сжатом формате, таком, как *HYB* или *CSR*, при этом структура сжатой матрицы получается простым обходом дерева. Наиболее важным недостатком такого подхода является чрезмерный расход оперативной памяти. В случае использования красно-черного дерева поиска для каждого из элементов матрицы требуется хранить координаты и значение элемента, а также три дополнительных указателя и одно логическое значение. В характерном случае использования двойной точности для значений, а также 64-битных указателей и 32-битных целых для индексов накладные расходы первого подхода

превосходят накладные расходы формата *CSR* более чем в 2,5 раза; более того, количество требуемой памяти пропорционально количеству ненулевых элементов матрицы, при этом информация о номере строки никак не используется. Значительные накладные расходы делают использование данного метода на больших матрицах нецелесообразным. Вставка элемента имеет логарифмическую асимптотическую сложность относительно количества элементов в матрице.

Второй подход, опубликованный в работе [3], предлагает хранить матрицу жесткости следующим образом: каждая строка матрицы хранится в виде односвязного списка. Доступ к конкретной строке матрицы осуществляется выбором ссылки на нужный список из массива, а поиск необходимого элемента происходит в процессе перебора элементов списка. Накладные расходы данного метода сравнимы с расходами формата *CSR*, а в случае аккуратной реализации с прямым управлением памятью и заменой указателей 32-битными индексами полностью с ним совпадает. Анализ асимптотического времени работы данной структуры показывает, что оно не зависит от количества ненулевых элементов матрицы, а связано только с длиной строки, которая, как правило, мала.

Оба метода были изначально предложены для квадратных матриц жесткости, однако хорошо обобщаются на матрицы общего вида, представляющие слои.

При решении упругопластических задач второй подход выглядит более предпочтительным, однако экспериментальное обоснование данного решения должно стать темой дополнительного исследования.

Хранение и передача между узлами элементов вектора

В ходе решения систем линейных алгебраических уравнений итерационными методами на системе с распределенной памятью необходимо пересылать между процессами *MPI* элементы вектора. Для большинства гетерогенных систем организация обмена элементами вектора значительно осложняется необходимостью переносить данные с ускорителя в основную память узла. Каждая операция копирования данных в память узла, как и операция прямой передачи данных между ускорителями, требует вызова процедуры драйвера ускорителя, что сопряжено со значительными накладными расходами. При программировании гомогенных кластерных систем в подобных ситуациях применяется разреженная схема хранения вектора, при которой для каждого элемента хранится его индекс. Вычисления с вектором, сжатым по такой схеме, имеют более

высокую асимптотическую сложность, чем вычисления с полным вектором. Алгоритмы сжатия и восстановления вектора плохо подходят для реализации на ускорителях. Опубликованные решения этой проблемы авторам работы неизвестны.

Авторами предлагается следующий подход: для минимизации накладных расходов вектор передается не поэлементно, а последовательными диапазонами одинаковой длины и достаточно крупного размера, называемыми в дальнейшем порциям. Размер порции зависит от настроек конкретной кластерной системы. Поскольку каждый процесс *MPI* работает лишь с одним слоем матрицы, на этапе сборки матрицы жесткости можно вычислить множество порций вектора, необходимых для умножения матрицы на вектор, называемое далее множеством исходных порций, и множество порций вектора, получаемых в результате операции, называемое далее множеством результирующих порций.

Множества результирующих порций для различных слоев не пересекаются, однако множества исходных порций для разных узлов содержат много повторяющихся элементов. Перед началом вычислительной процедуры умножения матрицы на вектор необходимо получить все недостающие исходные порции. Сложение векторов, а также операции редукции, такие, как нахождение нормы и скалярного произведения векторов, проводятся только над результирующими порциям

Рационально не хранить на узле те порции вектора, которые не входят ни в одно из множеств. В этом случае для эффективной реализации вычислительной процедуры умножения матрицы на вектор необходимо введение дополнительного уровня косвенности. Это достигается использованием специальных таблиц ссылок на начала порций векторов, называемых в дальнейшем *массивами диспетчеризации*. При организации вычислений с помощью массива диспетчеризации необходимые порции вектора хранятся в большом двухмерном массиве. Выделяется дополнительный массив указателей, размер которого совпадает с количеством порций вектора. Для того чтобы получить указатель на порцию вектора с индексом i, необходимо обратиться по i-му указателю в массиве диспетчеризации. При этом зная индекс элемента в векторе, можно легко получить номер порции, которая его содержит, выполнив целочисленное деление.

Хранение матрицы жесткости во время решения СЛАУ

На втором этапе требуется хранить матрицу жесткости A таким образом, чтобы обеспечить малое время выполнения операции умножения матрицы на вектор. Существует значительное количество исследований, посвя-

щенных подобным структурам данных [4], однако результаты их адаптации для гетерогенных кластерных суперкомпьютеров до сих пор не публиковались. Вариант способа хранения матрицы, оптимизированный для матриц, возникающих в упругопластических задачах, приводится в данной работе.

Внутри каждого слоя матрицы элементы разбиваются на две части. Первая часть хранится в формате *ELL-R*, описание которого можно найти в работе [5]. Используется два двухмерных массива: первый массив содержит значения элементов матрицы, а второй – их номера в соответствующей строке.

Количество элементов в строке матрицы зависит от количества конечных элементов, соотнесенных с соответствующим данной строке узлом сетки. Как правило, при решении упругопластических задач для тел с простой геометрией строки матрицы незначительно отличаются по количеству ненулевых элементов. При моделировании сложной геометрии возможно возникновение нехарактерных строк, которые имеют большее количество элементов. Элементы из таких строк, которые не помещаются в первую часть матрицы, хранятся во второй части. Для их хранения используется координатный формат. Операция умножения разреженной матрицы на вектор для первой части выполняется с высокой производительностью, так как не требует никаких дополнительных синхронизаций между потоками ускорителя. Для выполнения этой операции над координатной частью требуется задействование атомарных операций присваивания, однако число нехарактерных строк в характерном случае мало́.

Результаты вычислительного эксперимента

Для проверки предложенного подхода к хранению матрицы на втором этапе были проведены вычислительные эксперименты. Вычисления проводились на гетерогенной части кластера «Уран» Института математики и механики им. Н.Н. Красовского, оснащенной графическими ускорителями *Nvidia Tesla M2050*. Время работы реализации, основанной на предложенной технике, сравнивалось с временем выполнения аналогичной современной оптимизированной библиотечной процедуры, входящей в пакет *CUSPARSE*. Обе процедуры выполняли умножение матрицы на вектор. При этом контрольная процедура выполняла умножение матрицы на вектор, используя единственный ускоритель. Разработанная процедура выполняла умножение матрицы, в четыре раза превышающей по количеству ненулевых элементов контрольную. Таким образом, обе процедуры выполняли одни и те же арифметические операции. Результаты представлены в таблице.

Результаты сравнения процедур

Количество ненулевых элементов в строке матрицы	114 688	143 360	172 032	200 704	229 376	258 048	286 720
Время работы предложенного подхода, мс	1,02	1,45	1,52	2,83	2,02	2,61	2,52
Время работы контрольной процедуры, мс	0,73	1,43	1,09	2,02	1,59	2,63	1,83

Анализ результатов

Результаты вычислительных экспериментов показывают некоторое снижение производительности в сравнении с гетерогенными системами с общей памятью. Это объясняется необходимостью вычислений для организации дополнительного уровня косвенности при хранении вектора. Потеря производительности компенсируется возможностью параллельной работы множества ускорителей.

Работа выполнена в рамках программы фундаментальных исследований Президиума РАН №15 (проект 12-П-1-1025) при поддержке РНФ (проект №14-19-01358).

Список литературы

1. Поздеев А.А., Трусов П.В., Няшин Ю.И. Большие упругопластические деформации. – М.: Наука, 1986.

2. Jansson N. Optimizing sparse matrix assembly in finite element solvers with one-sided communication // High Performance Computing for Computational Science. – 2013. – P. 128–139.

3. Толмачев А.В, Коновалов А.В., Партин А.С. Анализ эффективности различных схем хранения матрицы при решении упругопластической задачи итерационными методами // Механика микронеоднородных материалов и разрушение: материалы VII Рос. науч.-техн. конф., Екатеринбург, 23–27 апреля, 2012 / ИМАШ УрО РАН. – Екатеринбург, 2012.

4. Li R., Saad Y. GPU-accelerated Preconditioned Iterative Linear Solvers // J. Supercomput. – 2013. – Vol. 63, № 2. – P. 443–466.

5. Moro G. Overview of Sparse Matrix formats. – 2010, available at: http://www.giammy.com/phd/sparse-matrix-formats.pdf.

Т.А. Ханова, В.Д. Кустикова

Нижегородский государственный университет им. Н.И. Лобачевского

ОБ ОДНОМ ПОДХОДЕ К РЕШЕНИЮ ЗАДАЧИ КЛАССИФИКАЦИИ ТРАНСПОРТНЫХ СРЕДСТВ

Рассматривается задача классификации транспортных средств на изображении. Приводится схема решения задачи с использованием методов, основанных на построении признаковых описаний для ребер объекта. Описывается прототип разработанной системы. Выполняется анализ результатов вычислительных экспериментов.

Введение

Задача классификации транспортных средств на видеопотоке является актуальной и практически значимой. Решение данной задачи позволяет анализировать качественный состав транспортных потоков, что представляет интерес в процессе обнаружения нарушений некоторых правил дорожного движения, оценивания нагрузки на дорожное полотно, а также для сбора статистической информации.

Математическая постановка задачи

Задача классификации транспортных средств на изображении состоит в том, чтобы для транспортных средств на входном изображении определить классы, которым они принадлежат. В простейшем случае рассматривается разделение на классы: «Автобус», «Автомобиль», «Мотоцикл», «Велосипед» [1, 2]. Задача классификации усложняется, если рассматривать внутриклассовое разделение, например «Фургоны» [3], «Внедорожники» и т.п. [3, 4].

Формализуем постановку задачи. Пусть $C = \{C_0, ..., C_{n-1}, C_n\}$ – множество классов транспортных средств, где класс C_n – класс выбросов. ***Классом выбросов*** называется общий класс всех объектов, которые не удалось отнести ни к одному из известных системе классов [5]. Тогда входными данными задачи классификации являются:

1. Изображение I, представленное в виде двумерной матрицы интенсивностей пикселей. $I = (\mathrm{v}_{ij})_{i=\overline{0,h-1},\, j=\overline{0,w-1}}$, где w, h – ширина и высота изображения соответственно, а v_{ij} – вектор интенсивностей.

2. Набор выделенных областей изображения, содержащих транспортные средства. Не уменьшая общности, можно считать, что область определяется координатами окаймляющего прямоугольника. Обозначим

множество прямоугольников как $B = \{b_l, l = \overline{0,s}\}$, где $b_l = ((x^l, y^l), w^l, h^l)$, (x^l, y^l) – координаты левого верхнего угла прямоугольника в системе координат, связанной с левым верхним углом изображения, w^l, h^l – его ширина и высота соответственно.

Задача состоит в том, чтобы каждому окаймляющему прямоугольнику поставить в соответствие класс транспортных средств. Таким образом, необходимо построить отображение $\varphi: B \to C$. Отображение φ может быть представлено как суперпозиция пары отображений $\varphi = \theta \cdot \omega: B \to C$. Отображение $\theta: B \to F$ переводит изображение, накрываемое каждым окаймляющим прямоугольником, в вектор признаков, F – множество векторов признаков. В большинстве случаев F является подмножеством пространства векторов $\Re^m$. *Функция-классификатор* $\varphi = \theta \cdot \omega: B \to C$ для каждого вектора признаков принимает решение о принадлежности транспортного средства некоторому классу. Построение отображения ω – ***тренировка классификатора*** – включает в себя построение множества векторов признаков F для тренировочного набора изображений транспортных средств известных классов и подбор параметров отображения.

Схема решения

В качестве основы алгоритма была выбрана работа [6]. Общая схема решения задачи включает построение модели класса и непосредственно классификацию.

1. Построение модели класса.

1.1. На первом этапе происходит извлечение краевых точек с помощью оператора Кэнни. Низкое разрешение и изменение освещения делает контуры объектов более информативными, чем сам объект.

1.2. Далее для каждой точки строится модифицированный дескриптор *SIFT* [6, 10].

1.3. Точки, расположенные рядом и обладающие близкими дескрипторами, группируются с помощью техники «сдвиг среднего» (*mean-shift* [7]), и их *SIFT*-дескрипторы объединяются.

1.4. Далее строится усредненная модель представителей класса. Обучение классификатора происходит на тренировочном наборе данных. Для каждого нового вектора признаков из тренировочного набора вычисляется расстояние χ^2 до остальных. И в зависимости от порогового значения текущий вектор становится новым признаком объекта или усредняется с существующим. Полученный в итоге набор векторов признаков составляет модель объекта.

2. Классификация. В процессе классификации строится модель изображения объекта по той же схеме. Для принятия решения используется правило Байеса [11].

Программная реализация

Программная реализация разработана на базе библиотеки *OpenCV* [8]. Система включает в себя два модуля:

1. Модуль построения описания объекта.
2. Модуль классификатора.

Модуль построения описания объекта включает выделение граничных точек, группировку полученного набора точек и формирование дескриптора изображения (рис. 1).

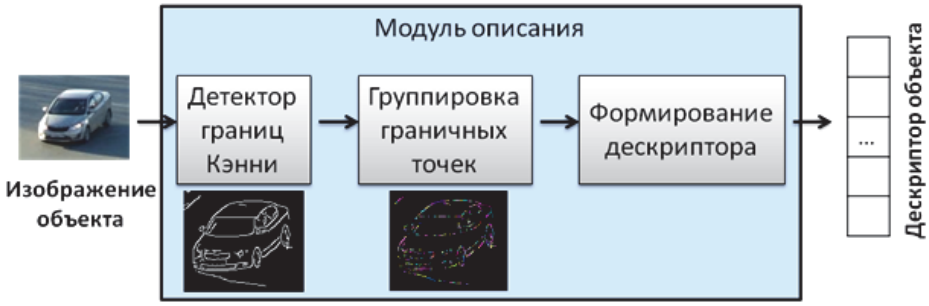


Рис. 1. Модуль построения описания объекта

Модуль классификатора состоит из следующих функциональных блоков (рис. 2), которые используются при обучении модели и классификации изображений:

1. Формирование множества дескрипторов, характерных для модели каждого класса.
2. Вычисление параметров модели класса объектов.
3. Вычисление вероятности принадлежности объекта к классу.

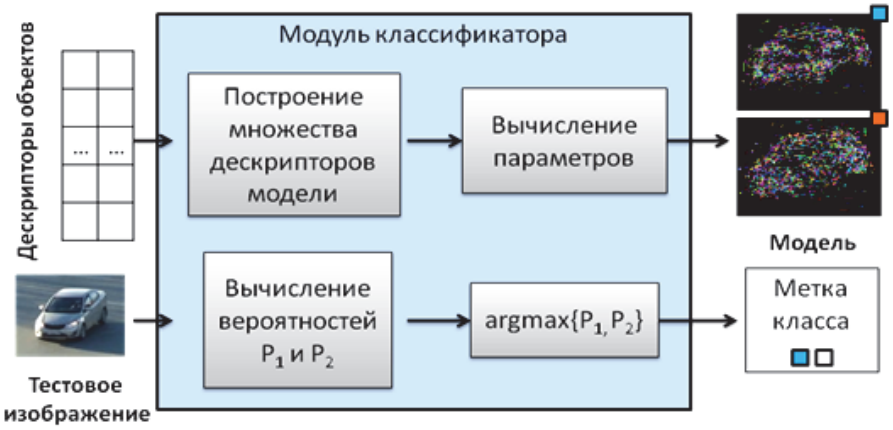


Рис. 2. Модуль классификатора

Обучение включает следующие этапы (рис. 3):

1. Все изображения тренировочной выборки переводятся в оттенки серого. На каждом изображении определяются граничные точки и в них строятся модифицированные *SIFT*-дескрипторы. Далее на множестве полученных точек запускается кластеризация методом сдвига среднего. Результатом кластеризации являются группы точек с близкими *SIFT*-дескрипторами.

2. В процессе извлечения дескрипторов из изображений тренировочной выборки формируются множества дескрипторов, характерные для каждого класса, путем слияния близких дескрипторов с разных изображений. В результате каждое множество дескрипторов ассоциировано с конкретным классом и содержит усредненные дескрипторы характерных частей объектов.

3. На основе построенного ранее множества дескрипторов обучается модель объекта. Значения дескрипторов рассматриваются как случайные величины, для которых с помощью метода максимального правдоподобия восстанавливаются параметры нормального распределения: матрица ковариаций и математическое ожидание.

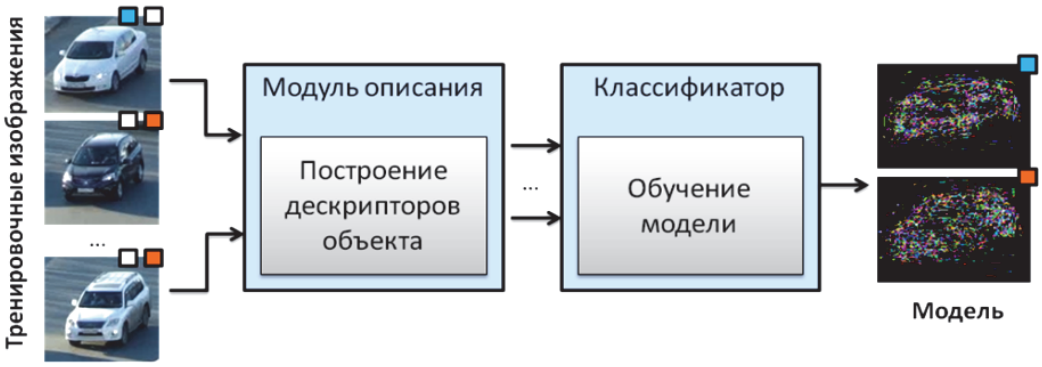


Рис. 3. Этап обучения модели

На этапе классификации модель класса уже построена. На вход подается изображение объекта, для которого необходимо принять решение о принадлежности классу. Классификация включает следующие действия (рис. 4):

1. Изображение переводится в оттенки серого. Затем из него извлекаются дескрипторы. Для этого используется тот же алгоритм, что и в процессе обучения модели.

2. Выполняется перебор по всем моделям.

2.1. Для каждого дескриптора тестового изображения вычисляется ближайший из дескрипторов рассматриваемой модели.

2.2. Определяется вероятность принадлежности объекта на тестовом изображении определенному классу по формуле Байеса.

3. Полученные для разных моделей вероятности сравниваются, модель с максимальной вероятностью является результатом классификации.

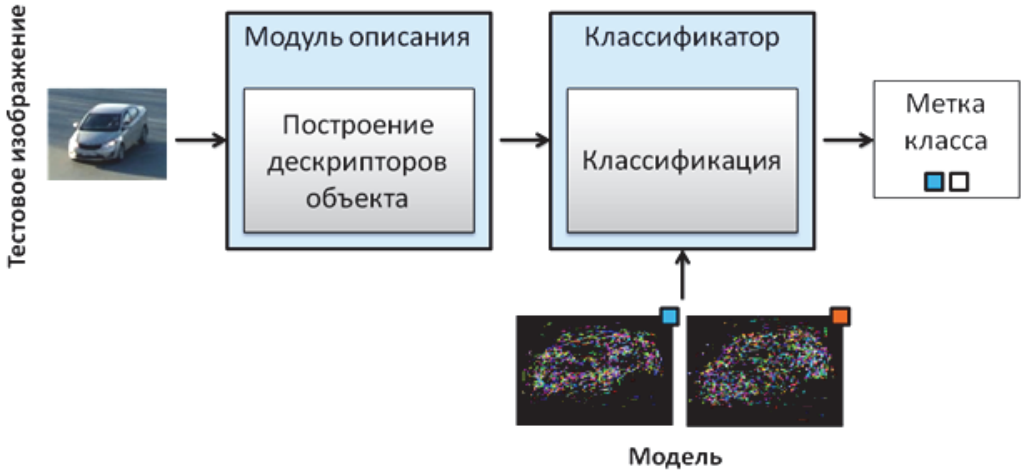


Рис. 4. Этап классификации

Критерии оценки качества поиска объектов

Для оценки качества классификации в работе используется *функционал качества* на тестовой выборке X_l, содержащей l объектов, вычисляемый по формуле

$$Q(X_l) = \frac{1}{l}\sum_{i=1}^{l} L(x_i),$$

где $L(x_i)$ – *функция потерь*, характеризующая ошибку классификации объекта $x_i \in X_l$ и вычисляемая по формуле

$$L(x_i) = \begin{cases} 0, & c(x_i) = c^*(x_i), \\ 1, & c(x_i) \neq c^*(x_i), \end{cases}$$

$c(x_i)$ – метка класса, присвоенная объекту классификатором; $c^*(x_i)$ – реальная метка класса. При реализации алгоритма классификации ставится задача минимизации функционала качества, по сути, доли неверно проклассифицированных объектов.

Тестовые данные

Качество классификации с использованием описанного ранее метода оценивалось на двух наборах данных. В каждой базе объекты имеют одинаковый ракурс.

Тестовая база 1. База изображений [9] из работы [6] (рис. 5, справа). Среднее разрешение изображений 90×60. Содержатся транспортные средства двух классов: «Седан» и «Минивэн». Тренировочная и тестовая выборки состоят из 100 и 400 изображений соответственно, соотношение представителей обоих классов 1:1.

Тестовая база 2. Собственная тестовая база изображений транспортных средств, содержащая 80 объектов двух классов: «Седан» и «Внедорожник» (рис. 5, слева). Объекты внешне схожи с [9]. Присутствуют сильные перепады освещения на изображениях. Среднее разрешение изображений 140×140 пикселей. Тренировочная и тестовая выборки состоят из 40 и 40 изображений соответственно, соотношение представителей обоих классов 1:1 в обеих выборках.

Рис. 5. Пример изображений из базы

Результаты экспериментов

Ниже приведены результаты классификации для описанных ранее тестовых баз. Таблицы содержат ошибку классификации «один против одного».

Результаты для тестовой базы 1

	Седан	Минивэн
Седан	0,785	0,215
Минивэн	0,2	0,8

Результаты для тестовой базы 2

	Седан	Внедорожник
Седан	0,75	0,25
Внедорожник	0,1	0,9

Для базы [9] и наилучших подобранных параметров метода классификации минимальное значение функционала качества $Q(X_l)$ на тестовой выборке равно 0,2075. Таким образом, доля неверно классифицированных объектов составила 20,75 %, причем 78,5 % «Седанов» и 80 % «Минивэнов» были проклассифицированы правильно. Авторы [6] показали, что на

ней может достигаться минимум в 0,0175. Указанное значение функционала качества воспроизвести не удалось, так как в работе [6] не приведены некоторые детали реализации и не указаны оптимальные значения для ряда параметров.

Для второй базы изображений было достигнуто минимальное значение функционала качества, равное 0,175. Доля верно распознанных объектов класса «Седан» составляет 75 %, и она ниже, чем для класса «Внедорожник».

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

Список литературы

1. Real-Time GPU Implementation of an Improved Cars, Pedestrians and Bicycles Detection and Classification System / A. Moro, E. Mumolo, M. Nolich, K. Umeda // Proc. Of the 14th Int. IEEE Conf. on Intelligent Trans. Systems (ITSC). – 2011. – P. 1343–1348.

2. Vehicle Class Recognition from Video based on 3D Curve Probes / D. Han, M.J. Leotta, D.B. Cooper, J.L. Mundy // Proc. of the 14th Int. Conf. on Computer Communications and Networks. – 2005. – P. 285–292.

3. Road Vehicle Classification using Support Vector Machines / Z. Chen, N. Pears, M. Freeman, J. Austin // IEEE Int. Conf. on ICIS. – 2009. – P. 214–218.

4. Kafai M. Dynamic Bayesian Networks for Vehicle Classification in Video // IEEE Trans. Industrial Informatics. – 2012. – P. 100–109.

5. Шапиро Л., Стокман Дж. Компьютерное зрение. – М.: Бином, 2006. – 752 с.

6. Ma X., Grimson W.E.L. Edge-based rich representation for vehicle classification // Proc. of the 10th IEEE Int. Conf. on CV (ICCV'05). – 2005. – Vol. 2. – P. 1185–1192.

7. Comaniciu D., Meer P. Mean shift: a robust approach toward feature space analysis // IEEE Transactions on PAMI. – 2002. – Vol. 24. – P. 603–619.

8. Библиотека компьютерного зрения OpenCV [Электронный ресурс]. – URL: http://opencv.org.

9. База видео транспортных средств [Электронный ресурс]. – URL: http://people.csail.mit.edu/ xiaoxuma/proj/vehi_reco/index_vehi_reco.htm.

10. Lowe D. Distinctive image features from scale-invariant keypoints // Int. J. of Computer Vision. – 2004. – № 60. – P. 91–110.

11. Воронцов К.В. Лекции по статистическим (байесовским) алгоритмам классификации [Электронный ресурс]. – URL: http://www.ccas.ru/ voron/download/Bayes.pdf.

И.В. Храмцов, П.В. Писарев, Р.В. Бульбович

Пермский национальный исследовательский политехнический университет

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ ГАЗОДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК ВИХРЕВОГО КОЛЬЦА С ИСПОЛЬЗОВАНИЕМ МЕТОДА КРУПНЫХ ВИХРЕЙ

Рассматривается задача численного моделирования интенсивных вихревых течений на примере элементарного объекта газовой динамики – вихревого кольца. Проведены расчеты в *ANSYS CFX* формирования вихревого кольца поршневым генератором. Результаты численного моделирования соответствуют автомодельному закону динамики вихревого кольца и экспериментальным результатам. Проведен анализ влияния скорости поршня на свойства вихревого кольца. Структура сформированного вихревого кольца остается устойчивой, что важно для экспериментальных исследований акустических свойств данного объекта.

Описание механизмов генерации шума турбулентными вихрями является сложнейшей задачей. На данный момент активно развивается численное моделирование акустических процессов для развития предсказательной теории генерации шума турбулентными течениями. Для этих целей может использоваться, например, метод Фокса Вильямса – Хокингса. Данный метод позволяет описывать звук в дальнем поле по колебаниям газодинамических параметров на контрольных поверхностях. При этом задача точного расчета турбулентных нестационарных течений даже для такого простого объекта, как вихревое кольцо, является достаточно сложной и требует существенных вычислительных ресурсов.

Вихревое кольцо является элементарным объектом аэроакустики. Будучи созданным, данный вихрь развивается только под действием собственной динамики, не испытывая влияния твердых границ. Это позволяет исследовать с его помощью многие вопросы газовой динамики в чистом виде. Вихревое кольцо обладает богатым набором характерных акустических проявлений: это излучение на дискретных частотах, резонансное поглощение звука и т.д., при этом оно доступно для экспериментального изучения.

В данной работе исследуется формирование вихревого кольца на поршневом генераторе вихревых колец НИО-9 Центрального аэрогидродинамического института (г. Москва). Схема генератора представлена на рис. 1. Данный генератор представляет собой полый цилиндр, на который навинчиваются сменные сопла. Внутри цилиндра свободно движется пор-

шень с последующей внезапной остановкой, импульсно выталкивающий порцию газа через сопло. Далее происходит отрыв образовавшейся вихревой структуры и ее перемещение в пространстве вдоль оси, совпадающей с осью поршня. С помощью геометрии данной модели осуществлялась верификация численных экспериментов с натурными.

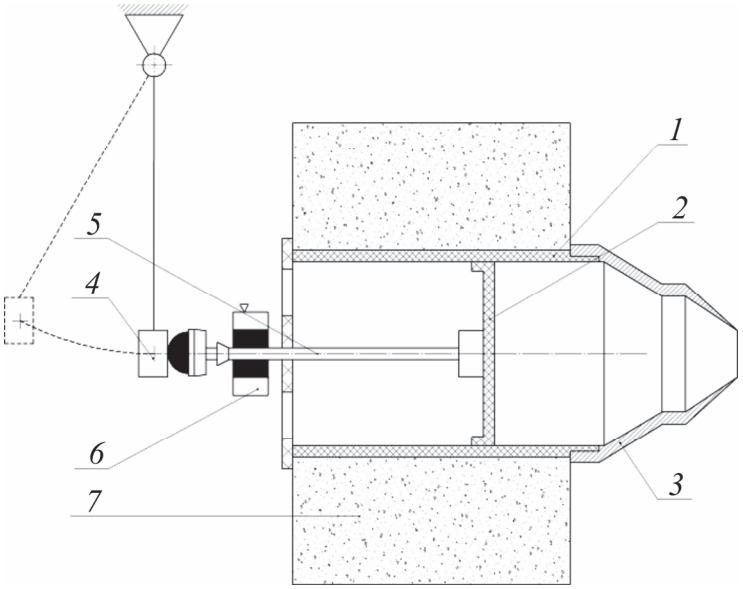


Рис. 1. Схема поршневого генератора вихревых колец
с ударным приводом: *1* – корпус; *2* – поршень; *3* – сопло;
4 – ударный механизм; *5* – шток; *6* – замок-ловушка штока;
7 – контейнер с песком

Данные генераторы позволяют получать достаточно интенсивные вихревые кольца с начальной скоростью более 30 м/с, тонким ядром и при этом обладают сравнительно низким уровнем собственного структурного шума. Кроме того, поршневые генераторы вихревых колец обеспечивают хорошую повторяемость параметров вихря от запуска к запуску.

Численное решение задачи проводилось на основе решения нестационарных уравнений Навье – Стокса. Процессы рассматривались в трехмерной динамической постановке. Расчеты проводились с учетом вязкости и сжимаемости среды. Моделирование движения поршня задавалось через движение стенки конструкции с постоянной скоростью. Шероховатость стенок и гравитация не учитывались.

Динамика вихревого кольца характеризуется сложными нестационарными турбулентными течениями. Для более точного описания данных течений используется метод крупных вихрей (*LES*). В качестве модели турбулентности принята *LES*-модель с подсеточной моделью Смагоринского, было использовано малое значение константы Смагоринского для предотвращения излишней диссипации.

Расчетная область представляет собой внутреннюю поверхность поршневого генератора и область за срезом сопла, в которой происходит формирование вихревого кольца. Использовалась модель сектора для снижения времени расчета. Рабочим телом принят идеальный газ воздух. Расчеты проводились при нормальных условиях. Схема граничных условий представлена на рис. 2.

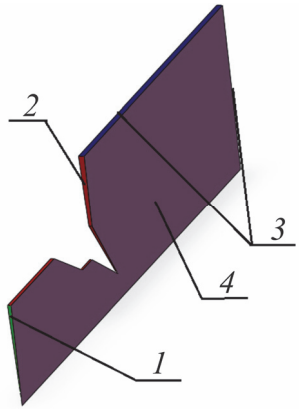


Рис. 2. Схема граничных условий:
1 – подвижная стенка; *2* – стенка;
3 – открытая граница; *4* – симметрия

Модель турбулентности *LES* требует высокого качества конечно-элементной модели. Расчеты проводились на модели сектора с центральным углом 5°. При этом конечно-элементная модель состоит из 15 млн элементов. Используются преимущественно шестигранные элементы со средней длиной ребра 0,1 мм. Размер элемента определялся по безразмерному параметру y^+, данный параметр позволяет определить размер элемента в пристеночном слое. В данном случае в ходе расчетов y^+ был в пределах 20–50 при разных скоростях движения поршня. Полученные величины позволяют достаточно точно описывать пограничный слой при использовании пристеночных функций.

При этом использование мелкой сетки и модели турбулентности *LES* требует малого шага по времени. В расчетах использовался шаг по времени $\tau = 1 \cdot 10^{-7}$ с. Таким образом, возникает неизбежная необходимость использования распараллеленного программного кода. В данном случае использовался коммерческий программный комплекс *ANSYS CFX*. Расчеты проводились на кластере Центра высокопроизводительных вычислительных систем ПНИПУ. Использовалось до 64 ядер, что позволяет сократить время расчета до нескольких дней.

Было проведено качественное сравнение формирования вихревого кольца с экспериментами, проведенными в институте гидродинамики им. Лаврентьева СО РАН [1]. Как можно увидеть на рис. 3, картина течения, полученная численно, хорошо соотносится с экспериментальными результатами. Движение кольца разделяется на два этапа. На первом этапе в области сопла течение формируется и сворачивается в тонкое вихревое кольцо, потом происходит отрыв от кромки сопла образовавшейся вихревой структуры с постепенным увеличением ее диаметра. На втором этапе

происходит увеличение поступательной скорости вихревого кольца, при этом значение радиуса практически не изменяется.

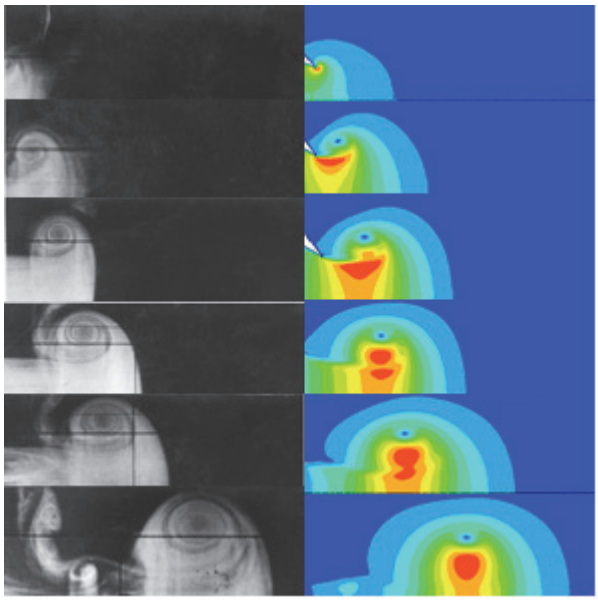


Рис. 3. Сравнение картины течения

Проведено сравнение полученных результатов с автомодельным законом динамики вихревого кольца [2], данный закон имеет хорошее совпадение с экспериментом. Система представляет собой закон движения вихревого кольца:

$$L(t) = \frac{R_0}{\alpha}\left[\left(1 + \frac{4\alpha U_0}{R_0}t\right)^{\frac{1}{4}} - 1\right], \quad R(t) = R_0 + \alpha L(t).$$

где R_0, U_0 – начальный радиус и скорость вихревого кольца соответственно; α – коэффициент, учитывающий скорость расширения вихревого кольца.

Для верификации вычислительного эксперимента проводилось сравнение начальной скорости вихревых колец, полученной в рамках проведенных расчетов, с результатами натурных экспериментов, для этого использовались две контрольные точки на расстоянии 100 и 153 мм от среза сопла соответственно. Скорость оценивалась исходя из времени прихода вихревого кольца на контрольные точки. Точки выбирались в соответствии с экспериментами [2]. Начальный радиус вихревого кольца оценивался по первой контрольной точке.

На рис. 4 представлены графики зависимости расстояния, пройденного вихревым кольцом, и радиуса от времени при скорости поршня 1,5 м/с. Как видим, получено хорошее совпадение с автомодельной теорией динамики вихревого кольца. Погрешность не превышает 3 %.

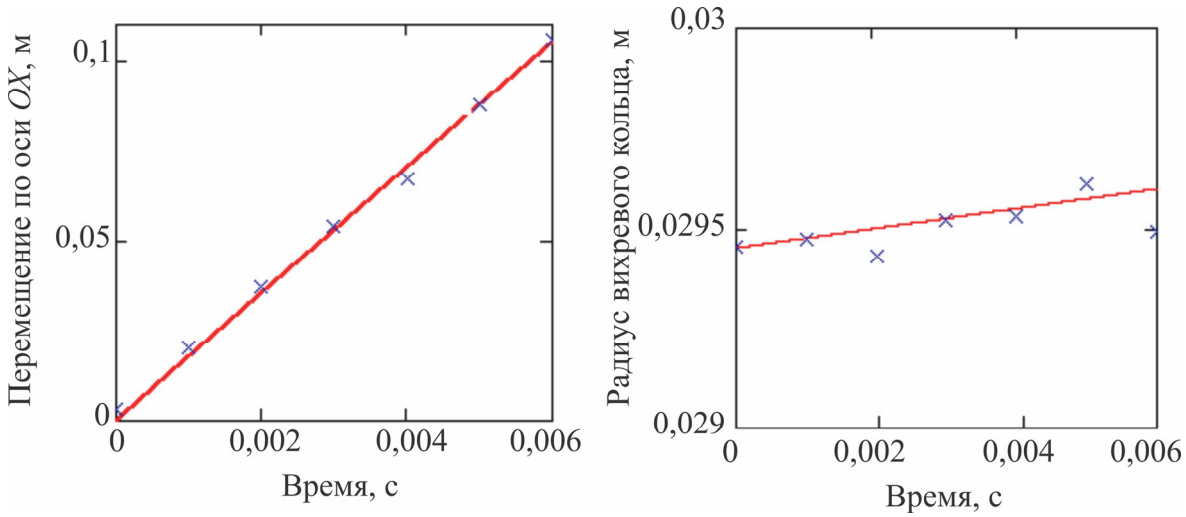


Рис. 4. Сравнение результатов численного моделирования и автомодельной теории: —— – автомодельная теория; × – численное моделирование

Проведен анализ влияния скорости поршня на свойства вихревого кольца. Оценивались начальный радиус, начальная скорость и коэффициент скорости расширения вихревого кольца. Результаты представлены в таблице.

Влияние скорости поршня на свойства вихревого кольца

Скорость поршня V, м/с	Начальный радиус R_0, м	Начальная скорость V_0, м/с	Коэффициент скорости расширения α
1,5	0,0295	17,6	0,0014
2	0,0312	23	0,005
3	0,0325	36,4	0,016

На основании проведенного исследования можно сделать следующие выводы:

Разработана методика численного эксперимента по исследованию газодинамических характеристик вихревого кольца.

Получено хорошее совпадение качественной картины формирования и динамики вихревого кольца с экспериментом.

Результаты численного моделирования соотносятся с автомодельным законом динамики вихревого кольца. Полученные характеристики вихревого кольца соответствуют экспериментальным результатам.

Проведен анализ влияния скорости поршня на свойства вихревого кольца.

Работа выполнена при поддержке гранта Правительства РФ по п. 220.

Список литературы

1. Ахметов Д.Г. Формирование и основные параметры вихревых колец // ПМТФ. – 2001. – Т. 42, № 5. – С. 70–83.

2. Лаврентьев М.А., Шабат Б.В. Проблемы гидродинамики и их математические модели. – М.: Наука, 1977.

3. Зайцев М.Ю., Копьев В.Ф. О смещении пика в спектре излучения вихревого кольца // Ученые записки ЦАГИ. – 1998. – Т. XXIX, № 3–4.

В.Е. Хромых, Д.И. Кислицын, А.Н. Супрун, А.И. Домрачев

Нижегородский государственный архитектурно-строительный университет

РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА ДЛЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ РАСЧЕТОМ СЛОЖНЫХ СТРОИТЕЛЬНЫХ ОБЪЕКТОВ НА КЛАСТЕРНЫХ СИСТЕМАХ

Дается описание архитектуры кластера на базе локальной компьютерной сети с веб-интерфейсом для эффективного расчета слабо распараллеливаемых вычислительных процессов, к которым относится конструкторский расчет сложных строительных объектов. При построении кластера используется известный метод разделения объекта на проектные единицы.

В строительной отрасли в России и за рубежом существуют тенденции к нарастанию сложности проектируемых объектов, повышению требований к надежности и устойчивости конструкций, сокращению сроков проектирования, повышению технико-экономических показателей строительных объектов. В то же время конструкторский расчет строительных сооружений относится к классу слабо распараллеливаемых вычислительных процессов, что не позволяет достаточно эффективно использовать возможности многопроцессорных вычислительных систем. В связи с этим в ННГАСУ разработана система «Менеджер конструкторских расчетов» (МКР) [1], позволяющая в соответствии с методом разделения объекта на проектные единицы (ПЕ) [2–7] сделать доступным решение задач неограниченной сложности на многопроцессорных вычислительных средствах, в частности в локальных компьютерных сетях на уровне персональных компьютеров.

Локальная версия системы МКР 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. С помощью данной системы происходит условное разделение

строительного объекта на проектные единицы, обеспечивающее их совместную работу как единой механической системы. В качестве базового программного средства (БПС) для расчета ПЕ используется известный вычислительный комплекс «Лира». Подробная работа локальной версии системы описана в публикациях [8–12].

В архитектуре локальной версии системы МКР компьютеры проектировщиков являются одновременно вычислительными узлами, на которых происходит расчет ПЕ. Поскольку современные тенденции развития информационных систем требуют реализацию в более компактном, мобильном виде, в настоящее время ведется разработка веб-сервиса для системы МКР. Добавление веб-сервиса привело к пересмотру архитектуры системы. Теперь компьютеры с БПС (вычислительные узлы) объединяются в кластер, которым управляет сервер управления (СУ) посредством установленных на вычислительных узлах клиентов СУ. Проектировщики же удаленно через Интернет получают возможность формировать модели ПЕ для расчета, передавая их и информацию о связях между ПЕ на сервер посредством веб-интерфейса. Тем самым обеспечивается применение параллельной технологии разработки расчетной схемы сооружения – операция, трудоемкость выполнение которой обычно во много раз превосходит затраты времени на конструкторский расчет.

Веб-сервис будет иметь интерфейс, доступный через браузер любого компьютера. Веб-интерфейс позволит работать как в режиме руководителя проекта, так и в режиме проектировщика. Руководитель проекта будет создавать проекты и следить за ходом их выполнения. Проектировщики будут заниматься проектированием своей ПЕ и загружать на веб-сервер все необходимые данные для расчета. Все результаты будут сохраняться в базе данных и будут всегда доступны через веб-интерфейс (рисунок).

Как видно из представленной схемы, система в целом объединяет в локальной сети как отдельные вычислительные узлы, используемые веб-сервисом, так и локальные компьютеры проектировщиков, а также серверы системы. Вычислительные узлы веб-сервиса содержат локально «Клиент системы управления», реализующий логику работы вычислительного узла, «Клиент сервера распределения файлов», необходимый для распределения файлов исходной модели ПЕ между вычислительными узлами, а также БПС для проведения необходимых расчетов. Вычислительные узлы проектировщиков в локальной сети имеют графический интерфейс, с помощью которого работает «Руководитель проекта», а также «Проектировщик». Графический интерфейс содержит основную логику работы для реализации автоматизации расчета ПЕ. Имеетя БПС для проведения расчетов. Сервер, если его

рассмотреть более детально, содержит в себе несколько компонентов: веб-сервер, сервер распределения файлов (СРФ) и сервер управления (СУ). Веб-сервер отвечает за работу веб-интерфейса. Данный сервер взаимодействует с вычислительными узлами, специально выделяемыми только для работы с веб-сервером. Имеющаяся база данных при данном сервере служит для корректной работы веб-интерефейса. Интерфейс может быть подгружен с помощью Интернета из любой точки мира.

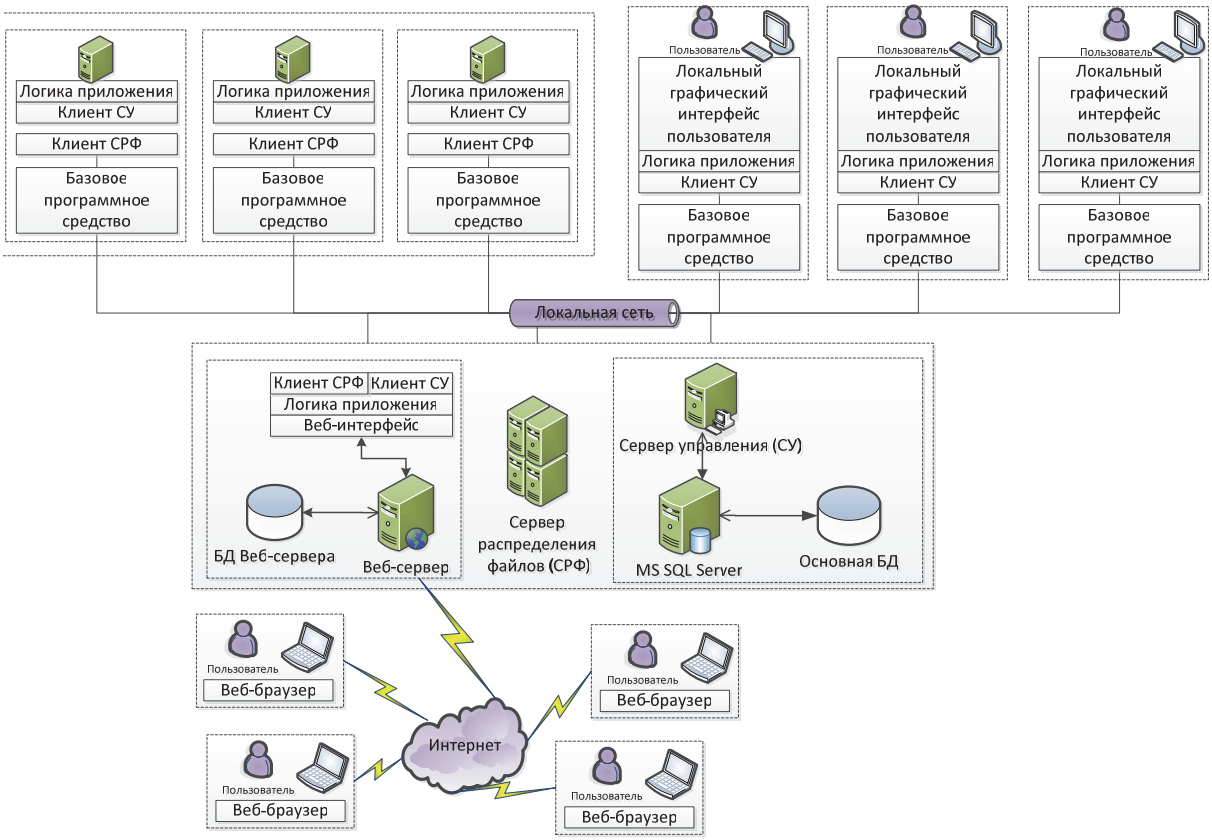


Рис. Архитектура кластера на базе локальной вычислительной сети с веб-интерфейсом

Сервер распределения файлов необходим для распределения файлов исходной модели ПЕ по определенным вычислительным узлам. Планируется, что сервер в дальнейшем будет в зависимости от загруженности вычислительных узлов выбирать оптимальный вариант загрузки имеющихся вычислительных узлов (кластера).

Сервер управления содержит в себе определенный набор логики, тем самым снижая нагрузку с клиентов, кроме того, это позволяет упростить администрирование системы в целом. СУ работает с БД *MS SQL Server*, в которой хранятся все необходимые данные для БД. Данные, содержащиеся

в данной БД, могут быть подгружены как через локальный графический интерфейс, так и через веб-интерфейс посредством взаимосвязи «Клиента СУ» с СУ. Новая архитектура позволит с одной стороны «отвязать» проектировщиков от своих рабочих мест в офисе, а с другой – перенести ресурсоемкие вычислительные узлы с персональных компьютеров на выделенный или виртуальный сервер или разместить их в *IaaS*-облаке.

Данный проект выполняется при поддержке программы Росмолодежи IT–START.

Список литературы

1. Супрун А.Н., Кислицын Д.И., Хромых В.Е. Менеджер конструкторских расчетов. Свидетельство о государственной регистрации программы для ЭВМ № 2014615169. – 2014.

2. Супрун А.Н., Кислицын Д.И. Распараллеливание вычислительных расчетов строительных конструкций на персональных компьютерах // Известия вузов. Строительство. – Новосибирск, 2006. – № 5.

3. Кислицын Д.И., Супрун А.Н. Программный модуль для расширения функциональных возможностей вычислительного комплекса «Лира» // International Journal for Computational Civil and Structural Engineering. – Moscow; New York, 2008. – Vol. 4, iss. 2.

4. Супрун А.Н., Кислицын Д.И., Скороходов В.В. Проблемы построения автоматизированных систем конструкторского расчета строительных объектов в распределенных вычислительных средах // Актуальные проблемы компьютерного моделирования конструкций и сооружений: материалы III Междунар. симпоз. – Новочеркасск: Изд-во Юж.-Рос. гос. техн. ун-та, 2010.

5. Кислицын Д.И., Хромых В.Е. Численная реализация метода разделения конструкции на проектные единицы // КОГРАФ – 2013: материалы 23-й Всерос. науч.-практ. конф. по графическим информационным технологиям и системам. – Н. Новгород, 2013.

6. Suprun A.N., Kislitsyn D.I. Distributed computing for construction project design by division into project design units // Computing in Civil and Building Engineering, Proceedings of the International Conference, 30 June – 2 July. – Nottingham, 2010.

7. 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.

8. Хромых В.Е., Кислицын Д.И. Укрупненная схема построения распределенных вычислений при расчетах сложных строительных конструкций методом разделения на проектные единицы // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XII Всерос. конф. / под ред. проф. В.П. Гергеля. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2012.

9. Хромых В.Е., Супрун А.Н., Кислицын Д.И. Проблема программной реализации вычислительной системы расчета сложных строительных объектов в распределенных компьютерных средах // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XIII Всерос. конф. / под ред. проф. В.П. Гергеля. – Н. Новгород: Изд-во Нижегор. гос. ун-та, 2013.

10. Мжачих А.С., Хромых В.Е. Проблемы реализации автоматизированной системы управления расчетом строительных объектов на базе метода разделения объекта на проектные единицы // КОГРАФ – 2012: 22-я Всерос. науч.-практ. конф. по графическим информационным технологиям и системам. – Н. Новгород, 2012.

11. Кислицын Д.И., Хромых В.Е., Численная реализация метода разделения конструкции на проектные единицы // КОГРАФ – 2013: 23-я Всерос. науч.-практ. конф. по графическим информационным технологиям и системам. – Н. Новгород, 2013.

12. Хромых В.Е. Разработка автоматизированной системы управления расчетом на прочность сложных строительных и промышленных объектов в распределенных компьютерных средах // Материалы I межвуз. науч.-практ. конф., посвященной науч. достижениями выпускников бакалавриата 2013 г. и стипендиатов Президента России и Правительства РФ. – СПб., 2013.

Н.И. Червяков, О.В. Аникуева

Северо-Кавказский федеральный университет, Ставрополь

СИСТЕМА ОСТАТОЧНЫХ КЛАССОВ И ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ ДЛЯ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Применение программируемых логических интегральных схем позволяет повысить производительность модульных арифметических процессоров цифровой обработки сигналов. Это приводит к слиянию систем остаточных классов и программируемых логических интегральных схем и позволяет достигнуть требуемой точности вычислений.

Совместное использование системы остаточных классов и современных программируемых логических интегральных схем обеспечивает высокую точность и пропускную способность выполнения двух операций – сложения и умножения, используемых при цифровой обработке сигналов реального времени.

На сегодняшний день с применением программируемой логики успешно реализуются архитектуры, типичные для суперЭВМ – лидеров мирового компьютеростроения. Программируемая логика служит прекрасной элементной базой для организации специфических вычислений, в частности нейросетевых, для реализации алгоритмов нечеткой логики и т.п. Отдельного упоминания заслуживает возможность широкого выбора принципов организации вычислений, например использование систем остаточных классов, арифметики рациональных дробей, различных систем счисления и много другого. Без особых сложностей на программируемых логических интегральных схемах (ПЛИС) аппаратно реализуются некоторые функции и алгоритмы с большим числом нестандартных операций, например с применением специфической битовой обработки. На многопроцессорных вычислительных средствах подобные операции реализуются крайне неэффективно [4].

Совместное использование ПЛИС и СОК (систем остаточных классов) позволяет ускорить основные операции цифровой обработки сигналов, таких как свертка, и за счет этого повысить производительность и обрабатывать сигналы в режиме реального времени.

Педположим, что система остаточных классов определена рядом L с положительными и попарными относительно главных целыми числами $\{m_1, m_2, \ldots, m_i\}$. Эти на числа называются модулями. Пусть $M = \prod_{i=1}^{L} m_i$; таким образом, кольцо модуля целых чисел $Z(M)$ представляет собой динамический диапазон *RNS*. Следовательно, любое положительное целое число $X \in Z(M)$ имеет единственное представление $L = [x_1, x_2, ..., x_L]$, где $x_i = X \bmod m_i$. Учитывая, что два целых числа X, $Y \in Z(M)$ и их соответствующие представления остатками $[x_1, x_2, ..., x_L]$ и $[y_1, y_2, ..., y_L]$, то арифметика в *RNS* определена следующим образом:

$$X \circ Y \leftrightarrow \left\lfloor \left|x_1 \circ y_1\right|_{m_1}, \left|x_2 \circ y_2\right|_{m_2}, ..., \left|x_L \circ y_L\right|_{m_L} \right\rfloor, \quad (1)$$

где $\circ$ – операция сложения, вычитания или умножение. Таким образом, арифметика СОК определена по прямой сумме кольца модуля целого числа m_i, $Z(m_i)$, и это выполняется параллельно без зависимости между остатками чисел.

Преобразование от представления остатка $[x_1, x_2,..., x_L]$ к целому числу X основано на китайской теореме об остатке. Теорема утверждает, что существует уникальный $X \in Z(M)$ для существующих остатков.

Таким образом, СОК обеспечивает фундаментальную методологию для разделения больших динамических систем диапазона на множество меньших, но независимых каналов, по которым вычисления выполнены параллельно [3].

Алгоритмы цифровой обработки сигналов, которые могут быть сформулированы как свертка, являются основными для числовой системы.

Учитывая последовательность входа $x(n)$, фильтр КИХ характеризуется операцией свертки:

$$y(n) = \sum_{k=0}^{N-1} h_k x(n-k), \tag{3}$$

где $y(n)$ – продукция фильтра; сроки h_k – коэффициенты фильтра; N обозначает длину фильтра. Если (3) закодировать по арифметике СОК, то уравнения примут вид

$$\left|y(n)\right|_(m_k) = \left|\sum_{k=0}^{N-1} h_k^i x_j(n-k)\right| m_j,\ j = 1, \ldots, L, \tag{4}$$

где h_k^j и $x_j(n-k)$ обозначают остатки h_k и $x(n-k)$ по модулю m_j. В этой схеме каждый модуль m в (4) для $j = 1, 2, ..., L$ функционирует независимо. Таким образом, для реализации каждого фильтра по модулю необходимо N множителей и $N-1$ сумматоров. Использование распределенной арифметики позволяет уменьшить количество конфигурируемых логических блоков программируемых логических интегральных схем за счет использования системы остаточных классов [1].

Устройства программируемых логических интегральных схем обеспечивают потенциальные решения для системы остаточных классов, базирующейся на *MAC*-алгоритмах. Эти устройства состоят из логических элементов. В зависимости от группы каждый логический элемент включает одну или более переменных на входе программируемых логических интегральных схем.

Сложные логические операции и управление были первыми задачами, для решения которых создавались ПЛИС. Ввиду их многообразия, разного уровня сложности, а также подробного освещения в литературе нецелесообразно останавливаться на их рассмотрении в настоящей работе. Тем не менее в качестве иллюстрации возможностей ПЛИС в данной области представляется показательным приведенный ниже пример реализации на программируемой логике ускоренной обработки *XML*-запросов.

В настоящее время в информационных системах сбора и обработки больших объемов данных для хранения информации в электронном виде стандартом де-факто является формат *XML* (*Extensible Markup Language*). Для навигации по *XML*-дереву и обращения к его элементам разработан специальный язык – *XPath* (*XML Path Language*). Суть задачи в упрощенном виде может быть сформулирована следующим образом. На сервере хранится большой документ в формате *XML*. От удаленных пользователей поступает интенсивных поток *XPath*-запросов. Необходимо для каждого запроса в реальном масштабе времени по известному алгоритму вычислить путь адресации, выбрать из имеющейся базы необходимые данные и сформировать результирующий *XML*-документ.

Данный процесс можно разделить на два последовательных этапа – синтаксический анализ и фильтрацию. Известное решение проблемы основано на кластеризации запросов, т.е. на их группировке по сходным признакам с целью сузить область поиска в пространстве элементов. Но реализация данного подхода на кластерной архитектуре приводит к интенсивным обменам между процессорными элементами. Возникает противоречие: параллельная по своей природе задача обработки большого количества независимых запросов становится тесно связанной, из-за чего теряются все преимущества кластеризации. Ситуация выглядит совершенно иначе при решении той же задачи на программируемой логике.

Для реализации вышеупомянутого подхода в реконфигурируемой структуре ПЛИС синтезируются цифровые автоматы (ЦА), на состояния которых отображаются элементы пользовательских запросов. Собственно кластеризация осуществляется накапливанием многих запросов в одном ЦА путем анализа элементов *XPath*-сообщений и обнаружения схожих путей адресации. За счет однообразия, а также использования априорно известной информации о запросах анализ и кластеризацию удается выполнить до начала процесса фильтрации. В результате синтаксический анализатор (*parser*) и фильтр (*filter*), представляющий собой большую комбинационную схему, могут функционировать как две независимые ступени конвейера. Следует заметить, что схема фильтра, синтезируемая в ПЛИС, оптимизируется специальными программными средствами под конкретный *XML*-документ, хранящийся на сервере. Таким образом достигается беспрецедентная степень адаптации вычислительной структуры к решаемой задаче, принципиально невозможная для микропроцессорных систем. Как следствие, в практических разработках удается достичь ускорения более чем на порядок по сравнению с традиционными подходами [2].

Список литературы

1. RNS Implementation of FIR Fillers based on Distributed Arithmetic using Field-Programmable Logic / A. Garcia, U. Meyer-Base, A. Lloris, F. Taylor // Proc. IEEE Int. Symp. on Circuits and Systems. – Orlando, 1999. – Vol. 1. – P. 486–489.

2. Boosting XML Filtering with a Scalable FPGA-based Architecture / A. Mitra, M.R. Vieira, P. Bakalov, W. Najjar, V.J. Tsotras // 4th Biennial Conference on Innovative Data Systems Research (CIDR), 4–7 January, Asilomar, 2009.

3. Actel Corporation, ProASIC 500K Family Data Sheet, 2000.

4. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах программируемой логики. – СПб.: БХВ-Петербург, 2002. – 608 с.

5. Зотов В.Ю. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPACK ISE. – М.: Горячая линия. – Телеком, 2003.

Н.И. Червяков, Д.Г. Шалалыгин, И.В. Мелишан

Северо-Кавказский федеральный университет, Ставрополь

НЕЙРОСЕТЕВАЯ РЕАЛИЗАЦИЯ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ НА ТОЧКАХ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ

Предлагается метод генерации псевдослучайных чисел, основанный на применении эллиптической арифметики. Предложенная схема основывается на переходе к проективным координатам, что позволяет не использовать операцию модулярной инверсии и применять в вычислениях систему остаточных классов. При этом метод модульного умножения в системе остаточных классов, основанный на алгоритме Монтгомери, позволяет получить высокую скорость работы генератора.

Введение

Псевдослучайные последовательности используются для генерации секретных ключей шифрования, вычисления цифровой подписи, а также для работы многих алгоритмов аутентификации. Как известно, эллиптическая кривая широко используется для построения криптосистем [1]. Одним из инструментов построения генераторов псевдослучайных последовательностей является эллиптическая кривая над конечным полем.

Эллиптическая кривая E над простым полем F_p, где $p > 3$, задается уравнением в форме Вейерштрасса: $E(F_p): y^2 = x^3 + ax + b$, где $4a^3 + 27b^2 \neq 0$. Пусть порядок $|E(F_p)|$ группы $E(F_p)$ равен r.

S. Hallgren в 1994 г. в работе [2] предложил генератор псевдослучайной последовательности, который называется арифметической прогрессией на E с начальным членом $P_0 \in E$ и разностью $G \in E$ и задается следующим рекуррентным соотношением:

$$P_n = P_{n-1} \oplus G = nG \oplus P_0, \; n = 1, \; 2, \; 3, ..., \qquad (1)$$

где $\oplus$ – групповая операция в $E(F_p)$.

Выходными значениями датчика (1) могут быть либо точки P_n, либо только их абсциссы x_n, либо только их ординаты y_n.

Генератор псевдослучайной последовательности на *E*

Генератор псевдослучайных последовательностей должен удовлетворять следующим двум требованиям, предложенным в работе [3]:

1. Статистической безопасности, а последовательность, созданная генератором псевдослучайных чисел, должна статистически ничем не отличаться от абсолютно случайной последовательности.

2. Криптографической безопасности: возможности, зная k-битов последовательности, предсказать $k+1$-бит.

Период линейного конгруэнтного генератора на эллиптической кривой, заданного формулой (1), равен порядку группы точек эллиптической кривой [4–6]. При выборе большого простого p получается последовательность большой длины.

Исследуемый генератор характеризуется равномерной распределенностью с отклонением от равномерности $O\left(\frac{\sqrt{p}\lg^2 p}{t}\right)$.

Для случая, когда известны разность G и старшие биты P_n и P_{n-1}, предложен эффективный алгоритм нахождения P_0 для генератора, построенного на базе арифметической прогрессии на эллиптической кривой, следовательно, он не обладает криптографической безопасностью. Поэтому секретным ключом в генераторе псевдослучайных чисел (1) должны являться P_0 и G. В рассматриваемом случае неизвестны эффективные алго-

ритмы предсказания бит, значит, генератор 1 является криптографически безопасным.

Для увеличения длины последовательности будем переходить от эллиптической кривой, заданной формулой $E\left(F_p\right): y^2 = x^3 + ax + b$, к эллиптической кривой $E'\left(F_p\right): y'^2 = x'^3 + a'x' + b'$ с использованием формул

$$\begin{cases} a' = k^2 a \bmod p, \\ b' = k^3 b \bmod p, \end{cases} \tag{2}$$

где $k = u^{-2} \bmod p$ и u – образующий элемент F_p^*.

Тогда $G' = \left(x_{G'}, y_{G'}\right) \in E'\left(F_p\right)$ и $P' = \left(x_{P'}, y_{P'}\right) \in E'\left(F_p\right)$, где точки G' и P' вычисляются по формулам

$$\begin{cases} x_{G'} = u^2 x_G, \\ y_{G'} = u^3 y_G, \\ x_{P'} = u^2 x_P, \\ y_{P'} = u^3 y_P. \end{cases} \tag{3}$$

Применение формул (2) и (3) для перехода от одной арифметической кривой к другой позволяет построить псевдослучайную последовательность, период которой в $\frac{p-1}{2}$ раз больше, чем у арифметической последовательности на E, так как $\operatorname{ord}\left(k\right) = \frac{p-1}{2}$ имеет такие же статистические и криптографические свойства.

Умножение Монтгомери

Элементами поля $GF\left(p\right)$ являются целые числа $\left[0, p-1\right]$, и арифметические операции в нем выполняются по модулю p. Алгоритм Монтгомери для модулярного умножения без деления предложен в работе [3] и представлен ниже как алгоритм 1, состоящий из пяти шагов, где R – основание системы счисления Монтгомери, $\gcd\left(R, p\right) = 1$ и $p < R$. R выбирается таким образом, чтобы можно было эффективно вычислить шаг 2 и шаг 5. Обычно выбирается степень 2 с представлением в двоичной системе счисления.

Алгоритм 1. Модулярное умножение Монтгомери

Вход: a, b, p, R, R^{-1}, где $a, b < p$.

Выход: $c \equiv abR^{-1} \bmod p$, $c < 2p$.

1) $s = a \cdot b$;

2) $t = s \cdot \left(-p^{-1}\right) \bmod R$;

3) $u = t \cdot p$;

4) $v = s + u$;

5) $c = \dfrac{v}{R}$ (операция сдвиг).

Поскольку алгоритм умножения Монтгомери не содержит операции деления, то его можно эффективно реализовать в системе остаточных классов (СОК).

Алгоритм на базе обобщенно-позиционной системы счисления (ОПСС) с работы [2] является основой для алгоритма умножения Монтгомери на базе СОК, так как позволяет избежать оценки ранга. Алгоритм 2 умножения Монтгомери в СОК из работы [3] представлен ниже.

Алгоритм 2. Умножение Монтгомери в СОК

Вход: $a_T, b_T, \left(-p^{-1}\right)_B$, Q_A^{-1}, p_A, где $a, b < 2p$.

Выход: c_E, где $c < 2p$ и $c \equiv abQ^{-1} \bmod p$.

1) $s_T = a_T \cdot b_T$;

2) $t_B = s_B \cdot \left(-p^{-1}\right)_B$;

3) $t_A = t_B /$ переход от базиса B к базису A СОК;

4) $U_A = t_A \cdot p_A$;

5) $v_A = s_A + u_A$;

6) $c_A = v_A \cdot Q_A^{-1}$;

7) $c_B = c_A /$ переход от базиса A к базису B СОК.

В алгоритме 2 используются два базиса СОК, заданные следующим образом: $A = \left(p_1, p_2, ..., p_L\right)$ и $B = \left(q_1, q_2, ..., q_L\right)$, так что $\gcd\left(p_i, q_j\right) = 1$, $\forall i, j \in [1, L]$, за T обозначено $T = A \cup B$.

На вход поступают числа a и b, представленные в двух базисах СОК a_T и b_T. На шаге 1, 3 и 4 используются операции сложения и умножения, которые легко реализуются в СОК. На шаге 2 и 5 основание системы Монтгомери R заменяется на $Q = \prod_{i=1}^{L} q_i$, которая является диапазоном B. Обозначим $P = \prod_{i=1}^{L} p_i$ диапазон A. Затем на втором этапе вычисляем t в ба-

зисе B СОК $t_B = s_B \cdot \left(-p^{-1}\right)_B$. Вычисления на шаге 3, 4 и 5 в базисе B не могут быть выполнены, так как на шаге 5 нам необходимо вычислить $Q^{-1} \bmod q_i$, которая не существует, так как $\gcd(Q, q_i) = q_i$. Таким образом, произведем конвертирование t из СОК с базисом B в СОК с базисом A, следовательно, вычисления на 3, 4 и 5-м шаге выполняются в СОК с базисом A. В результате работы этого алгоритма вычисляется c_T в СОК, заданной формулой $c = abQ^{-1} \bmod p$. Использование модулярного алгоритма умножения Монтгомери позволяет эффективно реализовать генератор псевдослучайных чисел в системе остаточных классов.

Отличие предложенного алгоритма от алгоритмов Монтгомери состоит в том, что реализация операций перевода из позиционного представления чисел в СОК и из СОК в позиционное представление производится с помощью нейронной сети конечного кольца, которая позволяет уменьшить сложность алгоритмов и улучшить возможность распараллеливания операций.

Алгоритм 3 на этапах 1–8 реализует формулу

$$\xi = U_1 + W_2 U_2 + \ldots + W_L U_L, \tag{4}$$

где $W_i = \prod_{j=1}^{i} p_j$, $\forall i \in [2, L]$, U_i вычисляются по следующим формулам:

$$U_1 = \xi_1,$$

$$U_2 = \left\langle \xi_2 - \xi_1 \right\rangle_{m_2},$$

$$U_3 = \left\langle \xi_3 - \xi_1 - W_2 U_2 \right\rangle_{m_3},$$

$$\ldots\ldots,$$

$$U_L = \left\langle \xi_L - \xi_1 - U_2 W_2 - U_3 W_3 - \ldots - U_{L-1} W_{L-1} \right\rangle_{m_L},$$

где $\left\langle x \right\rangle_{m_i}$ обозначает операцию $x \bmod m_i$.

На этапах 9–15 реализуется формула (4), где все вычисления производятся по модулям p_i, являющимися модулями нового базиса A.

Две операции конвертирования базиса в алгоритме модулярного умножения Монтгомери являются безошибочными и отличаются от других реализаций алгоритма, которые используют в Китайской теореме об остатках, так как не требуют оценки ранга. Достаточным условием существования $\left(-p_B^{-1}\right)$ и Q_A^{-1} является выполнение $\gcd(Q, p) = 1$ и $\gcd(P, Q) = 1$. Считается, что

$$c = \frac{v}{Q} = \frac{ab + tp}{Q} < \frac{(2p)^2 + Qp}{Q} = \left(\frac{4p}{Q} + 1\right) p \le 2p, \tag{5}$$

отсюда следует, что $4p < Q$ обеспечивает выполнение условия $c < 2p$, когда $a, b < 2p$. Из формулы (5) также видно, что $2p \le P$ также является достаточным условием $c < Q$ и $v < PQ$, поскольку v есть максимально возможное значение, которое меньше, чем PQ [6].

Перевод из двоичного представления в СОК

Пусть z – целое число, представленное в системе счисления по основанию 2^r в виде картежа $\left(z^{(L-1)}, z^{(L-2)}, ..., z^{(0)}\right)$ длиной L. Тогда z можно вычислить с использованием формулы

$$z = \sum_{i=0}^{L-1} z^{(i)} 2^{ri} = \left(2^{r(L-1)}, ..., 2^{r,1}\right) \begin{bmatrix} z^{(L-1)} \\ \vdots \\ z^{(1)} \\ z^{(0)} \end{bmatrix}, \tag{6}$$

где $0 \le z^{(i)} \le 2^r - 1$. Вычисление z_A может быть реализовано с использованием нейронной сети конечного кольца [5], которая позволяет вычислить значение z по модулю p_i, где z выражается формулой (7).

$$\left\langle z \right\rangle_{p_j} = \left\langle \sum_{i=0}^{L-1} z^{(i)} \left\langle 2^{ri} \right\rangle_{pj} \right\rangle_{p_j}, \ \forall j \in [1, L]. \tag{7}$$

Предварительно вычислив константы (синаптические веса) $\left\langle 2^{ri} \right\rangle_{p_j}$, получим структуру, использующую умножение с накоплением, которая может быть вычислена за L шагов при выполнении L модулей параллельно.

Преобразование из СОК в двоичное представление

Суть приближенного метода сравнения модулярных чисел основана на использовании относительной величины исходного числа к полному диапазону Китайской теоремы об остатках, которая связывает позиционное число A с его представлением в остатках $(\alpha_1, \alpha_2, ..., \alpha_n)$, где α_i – наименьшие неотрицательные вычеты числа, относительно модулей системы остаточных классов $p_1, p_2, ..., p_n$ следующим выражением:

$$A = \left\langle \sum_{i=1}^{n} \frac{P}{p_i} \left\langle P_i^{-1} \right\rangle_{p_i} \alpha_i \right\rangle_P, \tag{8}$$

где $P = \prod_{i=1}^{n} p_i$, p_i – модули СОК; $\left\langle P_i^{-1} \right\rangle_{p_i}$ – мультипликативная инверсия P_i относительно p_i, $P_i = \frac{P}{p_i} = p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_n$ [27].

Если (8) разделить на константу P, то получим приближенное значение

$$A = \left\lceil \left| \sum_{i=1}^{n} \frac{\left\langle P_i^{-1} \right\rangle_{p_i}}{p_i} \alpha_i \right|_1 P \right\rceil = \left\lceil \left| \sum_{i=1}^{n} k_i \alpha_i \right|_1 P \right\rceil, \quad (9)$$

где k_i – константы выбранной системы, $k_i = \frac{\left| P_i^{-1} \right|_{p_i}}{p_i}$; 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 может быть записана также как $A \bmod 1$, потому что $A = \lfloor A \rfloor + A \bmod 1$. Количество разрядов дробной части числа определяется максимально возможной разностью между соседними числами.

Заключение

Предложена модификация линейного конгруэнтного генератора на точках эллиптической кривой, позволяющая увеличить период в $\frac{p-1}{2}$ раз по сравнению с существующим линейными конгруэнтными генераторами на точках эллиптической кривой с сохранением равномерной распределенности последовательности и криптографической безопасности.

Усовершенствован модулярный умножитель Монтгомери за счет применения нейронной сети конечного кольца и нейронной сети обратного преобразования, которые позволяют повысить скорость работы и распараллелить операции перевода числа из двоичного представления в СОК и из СОК в двоичное представление.

Работа выполнена в рамках Государственного задания Министерства образования и науки Российской Федерации.

Список литературы

1. Koblitz N. Elliptic curve cryptosystems // Mathematics of Computation. – 1987. – Vol. 48, № 177. – P. 203–209.

2. Hallgren S. Linear congruential generators over elliptic curve // Cornegie Mellon Univ. – 1994. – CS-94-M3. – P. 1–10.

3. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации: учеб. пособие для вузов. – М.: Горячая линия – Телеком, 2005. – 229 с.

4. Mahassni E.El., Shparlinski I.E. On the uniformity of distribution of congruential generators over elliptic curves // Proc. Intern. Conf. on Sequences and their Applications, Bergen 2001. – London: Springer-Verlag, 2002. – P. 257–264.

5. Нейрокомпьютеры в остаточных классах / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, А.Н. Макоха. – М.: Радиотехника, 2003. – 272 с.

6. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникационные технологии. – 2011. – Т. 9, № 4. – С. 4–12.

Н.И. Червяков, Д.Г. Шалалыгин, Н.Н. Нагорнов, И.В. Мелишан

Северо-Кавказский федеральный университет, Ставрополь

НЕЙРОСЕТЕВАЯ ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ СХЕМЫ РАЗДЕЛЕНИЯ СЕКРЕТА ВИДЕОИЗОБРАЖЕНИЯ НА ОСНОВЕ МАТРИЦЫ АДАМАРА

Предлагается графическая схема разделения секрета с использованием матрицы Адамара в нейросетевом модулярном базисе. Особенностью данной схемы разделения секрета является параллелизм машинных операций, достигаемый за счет использования системы остаточных классов, что позволяет повысить скорость шифрования и дешифрования видеопотока в реальном времени, что подтверждается моделированием.

Введение

В связи с развитием систем передачи и обработки мультимедийной информации становится возможной передача изображений, аудио- и видеоинформации из одной части земного шара в другую в режиме реального времени. Беспроводные сети быстрого развертывания используются при организации системы передачи информации в чрезвычайных ситуациях, краткосрочных войнах, управлении движением автотранспорта и др. Сети быстрого развертывания (*MANET*) являются сетями, имеющими динамическую топологию, и, как предполагается, предлагают услуги без фиксированной инфраструктуры. Мобильные одноранговые се-

ти входят в класс сетей с мобильными узлами, выступающими как в качестве маршрутизаторов, так и в качестве абонентов. Для построения систем передачи информации в мобильных сетях предъявляются особые требования: надежность, безопасность, энергоэффективность, низкие аппаратные затраты и др.

Всё вышесказанное приводит к необходимости построения математических структур с особыми свойствами, удовлетворяющих всем требованиями мобильных сетей быстрого развертывания. Примером такой структуры может служить система остаточных классов (СОК), реализованная в нейросетевом базисе, что хорошо подходит для проектирования конечных устройств. Таким образом, особую актуальность приобретает разработка схемы разделения секрета в нейросетевом модулярном базисе. В качестве математического аппарата для построения секрета удобно использовать матрицу Адамара, которая позволяет повысить эффективность операции умножения.

Основная часть

Матрица Адамара и ее свойства

Матрица Адамара $H(n)$ – это квадратная матрица размера $n \times n$, составленная из чисел 1 и −1, столбцы которой ортогональны, так что справедливо соотношение

$$H \times H^T = nE_n,$$

где E_n – это единичная матрица размера n.

Теорема 1. Пусть H – матрица Адамара n порядка, тогда

$$H(2n) = \begin{pmatrix} H(n) & H(n) \\ H(n) & -H(n) \end{pmatrix} -$$

матрица Адамара порядка $2n$.

Использование теоремы 1 позволит построить матрицы Адамара порядка 2^k, где $k \in N$.

Теорема 2. Если A и B циркулянт $n \times n$ с элементами ±1 удовлетворяющими условию

$$AA^T + BB^T = 2nE_n,$$

тогда матрица $H = \begin{pmatrix} A & B \\ -B^T & A^T \end{pmatrix}$ является матрицей Адамара порядка $2n$.

Использование теоремы 3 в работе [3] позволяет построить матрицы Адамара порядка $n = 2^a \cdot 10^b \cdot 26^c$, где $a, b, c \in N \cup \{0\}$.

Схема разделения секрета (k, n)

Инициализация

Пусть имеется некий несжатый поток видеоинформации расширением $M \times N$. Первоначально осуществляется деление данного потока видеоинформации на фреймы (от англ. *frame* – кадр), затем для каждого фрейма осуществляется построчное чтение пикселей (от 1 до M). Для каждого прочитанного пикселя каждой строки находится его представление в формате *RGB*, т.е. три десятичных числа от 0 до 255, значения которых переводятся в двоичную систему исчисления и записываются последовательно друг за другом. В результате получается битовая последовательность T, состоящая из 24 бит. Далее:

1. Выбираем простые числа $2^{24} < p_1 < p_2 < ..., < p_n < 2^{32}$ так, чтобы $P = \prod_{i=1}^{k} p_i > 3 \cdot 2^{23} \cdot N,$ и сообщаем каждому из участников обмена (открытый ключ).

2. Используя теоремы 1, 2, генерируем случайным образом матрицу Адамара порядка N.

3. Для удобства передачи матрицы Адамара значения «-1» заменить на «0», таким образом *i*-я строка матрицы Адамара H_i является записью *N*-битного двоичного числа. Каждое из *N*-битных двоичных чисел переводится в систему остаточных классов с использованием нейронной сети конечного кольца. Каждому *i*-му участнику схемы сообщается вектор столбец $\begin{pmatrix} H_1 \bmod p_i \\ H_2 \bmod p_2 \\ \vdots \\ H_n \bmod p_n \end{pmatrix}$, содержащий остатки от деления *N*-битных чисел (строк матрицы Адамара) по простому модулю p_i, где $i = \overline{1, n}$ (секретный ключ).

Замечание 1. В случае, если для генерации случайной матрицы Адамара использовалась теорема 2, достаточно передать только первую строку матрицы Адамара.

Разделение секрета

1. Вычислим значение $S_1 = H \times S.$

2. Каждому *i*-му участнику схемы передаем видеоизображение, полученное при вычислении $S_1 \bmod p_i$.

Восстановление секрета

Пусть участник с номером R получил $S_1^{R_1}, S_1^{R_2}$, ..., $S_1^{R_k}$ и H^{R_1}, H^{R_2}, ..., H^{R_k} проекции секрета, где $S_1^{R_i}$ обозначена *i*-я проекция секрета, а H^{R_i} – *i*-я проекция матрицы Адамара.

1. Используя приближенный метод из работ [3–6], восстановим из H^{R_1}, H^{R_2}, ..., H^{R_k} матрицу Адамара.

2. Вычислим представления секрета S в системе остаточных классов $S^{R_i} = \left\langle N^{-1} \right\rangle_{p_{R_i}} \cdot H^T \times S_1^{R_i}$, где $\left\langle N^{-1} \right\rangle_{p_{R_i}}$ – мультипликативная инверсия N относительно p_{R_i}, которая является константой, вычисляемой один раз для каждого модуля при дешифровании всего видеопотока.

3. Используя приближенный метод [4] и нейронную сеть конечного кольца, восстановим исходное видеоизображение из набора $S^{R_1}, S^{R_2}, ..., S^{R_k}$.

Пример. Схема разделения секрета $(3,5)$ при использовании 176×144 пикселей приведена на рис. 1.

Пусть $p_1 = 2\,147\,483\,659$, $p_2 = 2\,147\,483\,693$, $p_3 = 2\,147\,483\,713$, $p_4 = 2\,147\,483\,743$, $p_5 = 2\,147\,483\,777$, тогда

$$P = \prod_{i=1}^{3} p_i = 393\,530\,562\,412\,621\,947\,477 > 3 \cdot 2^{23} \cdot N = 3\,623\,878\,656.$$

Для разделения секрета используем матрицу Адамара размером 144×144, заданную в виде черно-белого изображения (рис. 2), где черный – «+1», а белый – «–1», которую можно задать двумя векторами строками x_A и x_B по 72 бита.

Рис. 1. Исходное изображение

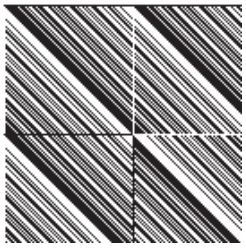
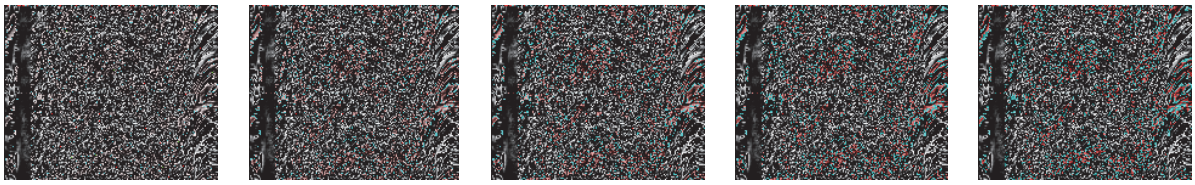


Рис. 2. Матрица Адамара 144-го порядка

В результате выполнения операции $H \times S \bmod p_i$ разделения секрета получим пять изображений (рис. 3).

Из рис. 3 видно, что каждое из изображений, передаваемых по каналам связи, не дает никакой информации о передаваемом объекте и выглядит как шум.



а *б* *в* *г* *д*

Рис. 3. Зашифрованная информация: *а* – $H \times S \bmod p_1$; *б* – $H \times S \bmod p_2$; *в* – $H \times S \bmod p_3$; *г* – $H \times S \bmod p_4$; *д* – $H \times S \bmod p_5$

Получив, например, рис. 3, *а*, *в*, *д*, восстанавливаем секретное изображение (см. рис. 1).

Таким образом, предложенный метод, основанный на использовании системы остаточных классов в нейросетевом базисе и матрицы Адамара заданного порядка, позволяет построить эффективную графическую схему разделения секрета с заданным порогом.

Данная работа выполнена в рамках Государственного задания Российской Федерации.

Список литературы

1. Wallis W.D., Street A.P., Wallis J.S. Combinatorics: Room Squares, Sum-Free Sets, Hadamard Matrices. Lecture Notes in Mathematics, Berlin; Heidelberg; New York: Springer-Verlag, 1972. – Vol. 292.

2. Turyn R.J. Hadamard matrices, Baumert-Hall units, four-symbol sequences, pulse compression, and surface wave encoding // J. Combin. Theory A. – 1974. – № 16. – P. 313–333.

3. Нейрокомпьютеры в остаточных классах / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, А.Н. Макоха. – М.: Радиотехника, 2003. – 272 с.

4. Модулярные параллельные вычислительные структуры нейропроцессоных систем / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, С.А. Ряднов. – М.: Физматлит, 2003. – 288 с.

5. Применение искусственных нейронных сетей и системы остаточных классов в криптографии / Н.И. Червяков, А.А. Евдокимов, А.И. Галушкин, И.Н. Лавриенко, А.В. Лавриенко. – М.: Физматлит, 2012. – 280 с.

6. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникационные технологии. – 2011. – Т. 9, № 4. – С. 4–12.

И.С. Чирков, С.Ю. Федосеева, П.В. Писарев

Пермский национальный исследовательский политехнический университет

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ НЕСУЩИХ ЭЛЕМЕНТОВ КОНСТРУКЦИИ СТОМАТОЛОГИЧЕСКОГО КРЕСЛА

Разрабатывалась несущая конструкция нестандартного медицинского оборудования – стоматологического кресла, а также проводился ряд вычислительных экспериментов с применением высокопроизводительного вычислительного комплекса ПНИПУ по расчету максимальных перемещений и напряжений, возникающих в конструкции в процессе эксплуатации.

Введение

Вопросы организации стоматологической помощи всегда были в центре внимания отечественного здравоохранения. В советские времена для наращивания производительности труда нанимали большее количество специалистов. В настоящее время к данному вопросу подходят с другой стороны: идет внедрение высокотехнологичного оборудования. На сегодняшний день вопросу разработки нестандартного медицинского оборудования уделено недостаточное внимание. Вместе с тем с развитием технологий, материалов, а также ростом требований к современному медицинскому оборудованию увеличился спрос на высокотехнологичные изделия в этой отрасли. Оказывать стоматологическую помощь стали не только государственные медицинские учреждения, но и частные негосударственные поликлиники, отделения и кабинеты. Присутствие на отечественном рынке преимущественно иностранных дорогостоящих производителей делает задачу создания конкурентоспособной отечественной продукции особенно актуальной.

В данной работе стоит задача конструирования и разработки нестандартного медицинского оборудования – стоматологического кресла, а также проведения ряда вычислительных экспериментов с применением высокопроизводительного вычислительного комплекса ПНИПУ по расчету максимальных перемещений и напряжений, возникающих в конструкции в процессе эксплуатации.

Цель работы – разработка стоматологического кресла, удовлетворяющего следующим требованиям: масса конструкции не более 120 кг; статическая рабочая нагрузка до 3 кН; габариты конструкции в стандартном положении до 2×1,5 м;

Физическая модель

Процессы рассматриваются в трехмерной статической линейной постановке задачи с учетом малых деформаций. При снятии нагрузок детали конструкции полностью восстанавливают свою исходную форму (учитывается необратимая деформация). Вместе с тем принималось, что максимальное расчетное перемещение значительно меньше характерного размера детали.

При формировании физической модели были приняты следующие допущения:

1) не учитывается действие силы трения;

2) влияние напряжений в конструкции на ее жесткость не учитывается;

3) не учитываются эффекты термопрочности;

4) в зоне контакта рассматривается полное прилипание.

5) стенки конструкции не нагреваются и не поглощают тепло.

В соответствии с принятой физической моделью разработана математическая модель, которая содержит уравнения равновесия, геометрические соотношения Коши, определяющий закон деформируемой сплошной среды, краевые условия.

Математическое описание упругих процессов для каждого из контактирующих тел рассматриваемой конструкции в указанной постановке включает в себя следующие соотношения:

Дифференциальные уравнения движения

$$\sigma_{ij,j} + \rho\left(F_i - u_{i,tt}\right) = 0. \tag{1}$$

Кинематические (геометрические) уравнения – уравнения Коши

$$\varepsilon_{ij} = \frac{1}{2}\left(U_{i,j} + U_{j,i}\right). \tag{2}$$

Физические уравнения (закон Гука)

$$\sigma_{ij} = C_{ijmn}\varepsilon_{mn}. \tag{3}$$

Краевые условия:

Граничные условия в местах крепления опорной панели

$$U_i\big|_A = 0; \tag{4}$$

$$\left.\frac{\partial U}{dn}\right|_A = 0. \tag{5}$$

Условия на нагруженной поверхности в напряжениях

$$\sigma_{ij} n_j\big|_S = P_i; \tag{7}$$

$$\sigma_{ij} n_j \Big| = 0 \tag{8}$$

Начальные условия

$$u_i = u_{0i}. \tag{9}$$

Все виды статического анализа основываются на следующем общем уравнении движения в конечно-элементной форме:

$$[K]\{u\} = \{F\}, \tag{10}$$

где $[K]$ – матрица жесткостей; $\{F\}$ – вектор сил; $\{u\}$ – вектор перемещений.

Компоненты вектора сил $\{F\}$ могут представлять собой сосредоточенные силы, тепловые нагрузки, давления и силы инерции.

В рамках расчета в указанной постановке проводится расчет по определению таких значений ускорений, которые обеспечивают статическое уравновешивание приложенных к системе нагрузок.

Разработка и описание сеточной модели

Для решения поставленной задачи была сгенерирована сеточная модель, в которой количество элементов составило 7 млн (рис. 1).

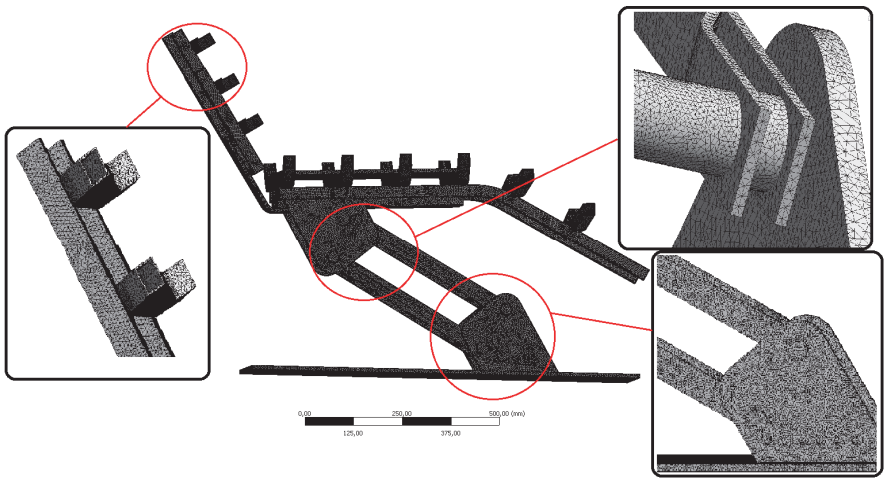


Рис. 1. Общий вид конечно-элементной расчетной сетки

По результатам исследования зависимости погрешности от размера элемента было установлено:

1) что при уменьшении размера элемента, начиная с 7 мм, относительная погрешность существенно не меняется;

2) существуют области, требующие адаптации конечно-элементной сетки;

3) адаптацию первого уровня необходимо провести в местах «жесткой заделки», а также в местах переходов толщины конструкции, размер элемента адаптации 0,1–1 мм.

Вместе с тем при измельчении сетки избегались резкие отличия геометрических размеров соседних ячеек – линейные размеры не должны отличаться более чем в два раза.

Проведение численных экспериментов

Сидячее место на стоматологическом кресле предназначено для размещения пациента. После посадки человек долгое время находится непосредственно на кресле, тем самым нагружая конструкцию. Задачей данного эксперимента является определение возможности конструкции выдержать вес человека при статической и динамической нагрузке (падение человека с небольшой высоты). На рис. 2 представлена схема нагружения в *ANSYS*.

По результатам проведения вычислительных экспериментов были получены распределение полей напряжений (рис. 3), перемещений (рис. 4), определен запас прочности, минимальное значение составило 2,5, что в два раза выше, чем у аналогичных изделий китайского производства.

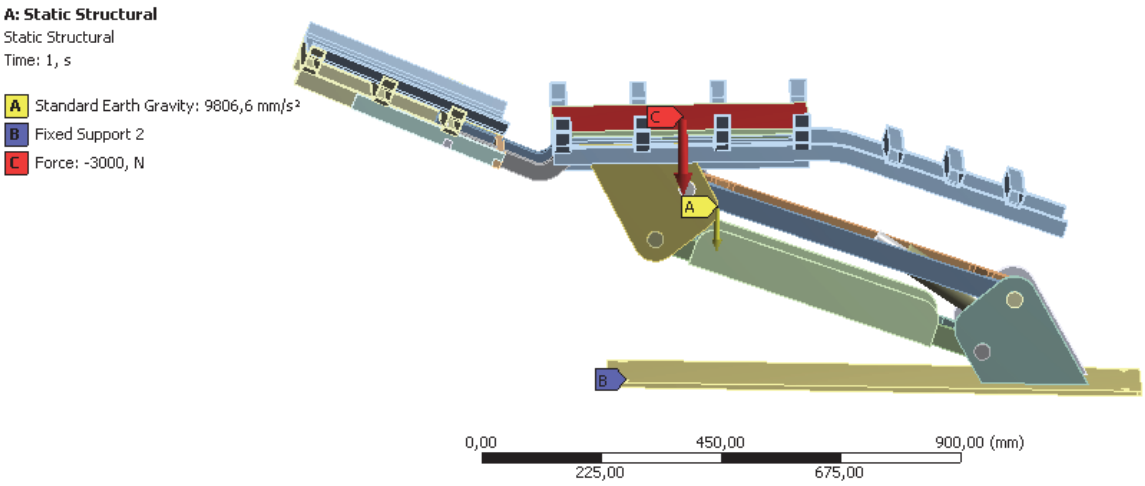


Рис. 2. Схема нагружения

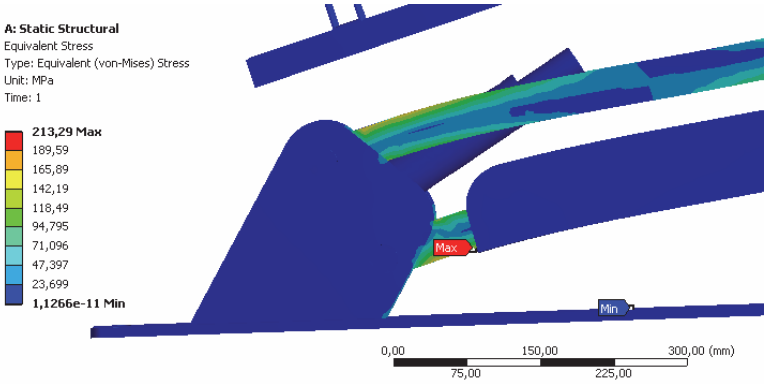


Рис. 3. Распределение напряжений в месте их максимального значения

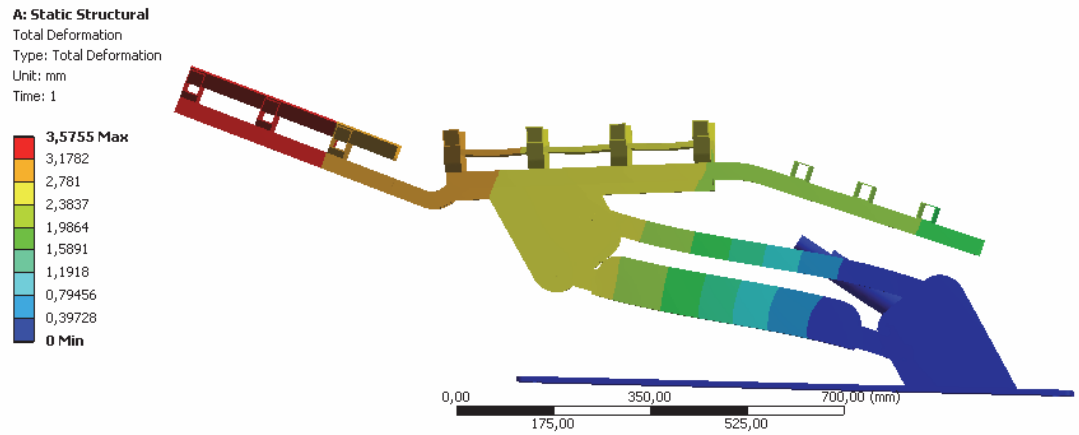


Рис. 4. Поле распределений общих перемещений

Заключение

На основании проведенных исследований можно сделать следующие выводы:

1. Разработана конструкция стоматологического кресла с учетом всех требований. Вес изделия равен 87 кг, габариты конструкции в стандартном положении не превышают 2000×1500 мм.

2. Разработан технологический процесс изготовления элементов конструкции и сборки стоматологического кресла.

3. Разработана методика численного моделирования НДС конструкции с использованием высокопроизводительного вычислительного комплекса ПНИПУ, определен минимальный запас прочности при нагрузке 3 кН, он равен 2,5. Определено максимальное перемещение, которое равно 3,5 мм и является допустимым для данного класса конструкций.

Л.П. Шингель

Пермский национальный исследовательский политехнический университет

ОБТЕКАНИЕ САМОЛЕТА ВЕРТИКАЛЬНОГО ВЗЛЕТА ПРИ АВАРИЙНОМ СНИЖЕНИИ

Представлены прикладная задача и ее решение, посвященные обтеканию самолета вертикального взлета при аварийном снижении, полученные при использовании *ANSYS* на кластерной системе.

В области решения прикладных задач, требующих вычислений с применением высокоскоростных кластерных систем, существует ряд задач вычислительной гидродинамики (*CFD*), успешно решаемых в *ANSYS* и позволяющих описывать процессы, ранее недоступные ввиду большого объ-

ема вычислений. К таким задачам относится задача моделирования обтекания летательного аппарата с целью нахождения технических решений, обеспечивающих наиболее благоприятное распределение скоростей потоков. При разработке новых проектов представляет интерес анализ обтекания при нетрадиционных и мало описанных условиях эксплуатации. Эти задачи в двумерной постановке, без использования кластерных систем, были представлены в работах [1, 2].

В этих работах описано обтекание крыла экспериментального самолета, когда самолет начинает падение под действием силы тяжести, практически не имея вертикальной скорости. К такому случаю относится и аварийное снижение самолета вертикального взлета при внезапной остановке двигателя в режиме висения. Продольная ось самолета и крыло при этом расположены практически перпендикулярно набегающему потоку, а скорость потока невелика (до 10 м/с). Задачей рассмотрения является определение возмущений в потоке, вызванных поперечным расположением самолета, имеющего крыло и фюзеляж. Задача существенно отличается от рассмотренных в работах [1, 2] тем, что, во-первых, является трехмерной и, во-вторых, рассматривается совокупность крыла и фюзеляжа. Решение таких задач наталкивается на ограничения используемого объема памяти и быстродействия и может быть получено с помощью высокопроизводительных вычислений.

В качестве расчетной модели использовалось крыло профиля *NACA 2309* (аналогичное приведенному в [1, 2]), установленное на фюзеляже длиной 4,07 м, максимальной шириной 1,3 м, при общем размахе крыла 8,76 м. Использовалась расчетная модель фюзеляжа, созданная для экспериментального самолета вертикального взлета. Расчетная область первоначально выбиралась сопоставимой с рабочей областью в [1, 2].

При расчетах [1, 2] использовалась конечно-элементная сеть, содержащая свыше 0,5 млн узлов. При сопоставимой плотности конечно-элементной сети трехмерной задачи требуется свыше 15 млн узлов.

Расчетная область имеет вид цилиндра диаметром 12 м, высотой 10 м. Данная расчетная область использована при нахождении распределения скоростей, давлений и ряда других параметров потока. В качестве параметров газа использовались параметры воздуха, задаваемые в *ANSYS* опцией *AIR-SI*. Скорость набегающего потока при рассматриваемом движении направлена вертикально вверх, в качестве граничных условий задавались вертикальная компонента скорости потока (составляющая 9,5 м/с) и горизонтальные компоненты скорости потока (равные нулю), а также нормальное атмосферное давление на входе и на выходе расчетной области.

При решении задачи применялись опции *ANSYS*, учитывающие турбулентность.

Рассматриваемая модель проиллюстрирована на рис. 1, расчетная область показана на рис. 2.

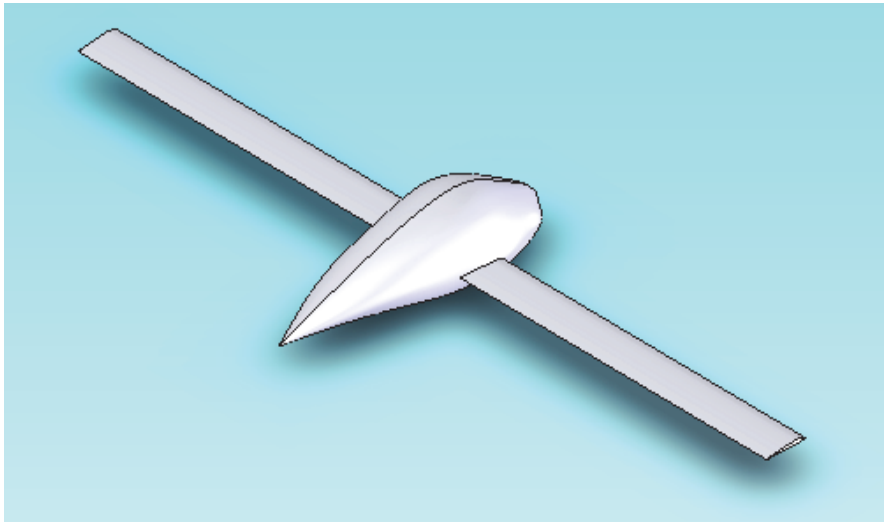


Рис. 1. Модель самолета вертикального взлета

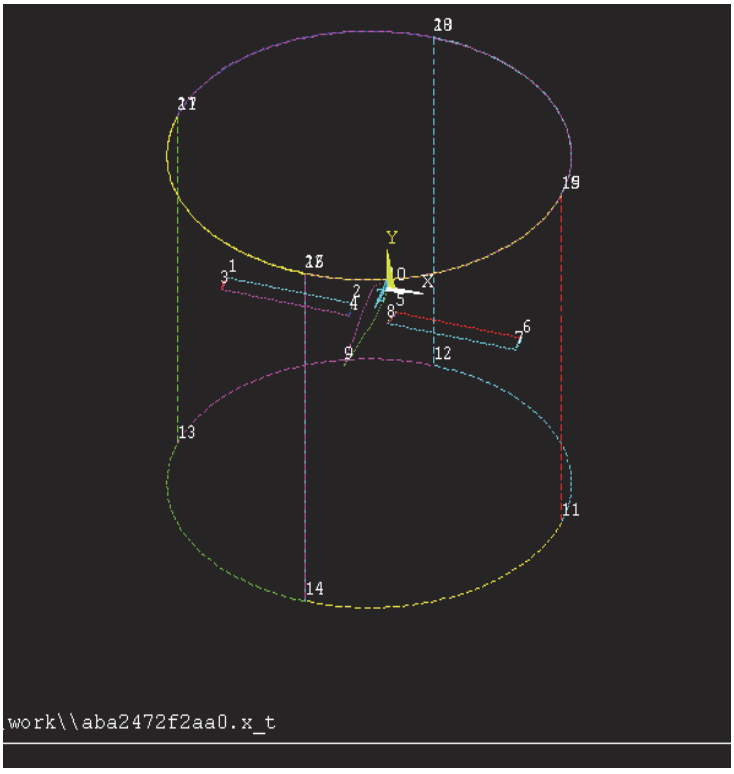


Рис. 2. Расчетная область

В результате расчетов, выполненных в ПНИПУ, получены результаты, выдаваемые *ANSYS*: в частности распределения скоростей потоков при рассматриваемом движении модели самолета вертикального взлета. На

рис. 3 приведен пример распределения скоростей обтекания в плоскости, проходящей через центр тяжести самолета. Следует отметить, что картина обтекания соответствует обтеканию, известному из существующих теорий, видны даже вихри на концах крыла.

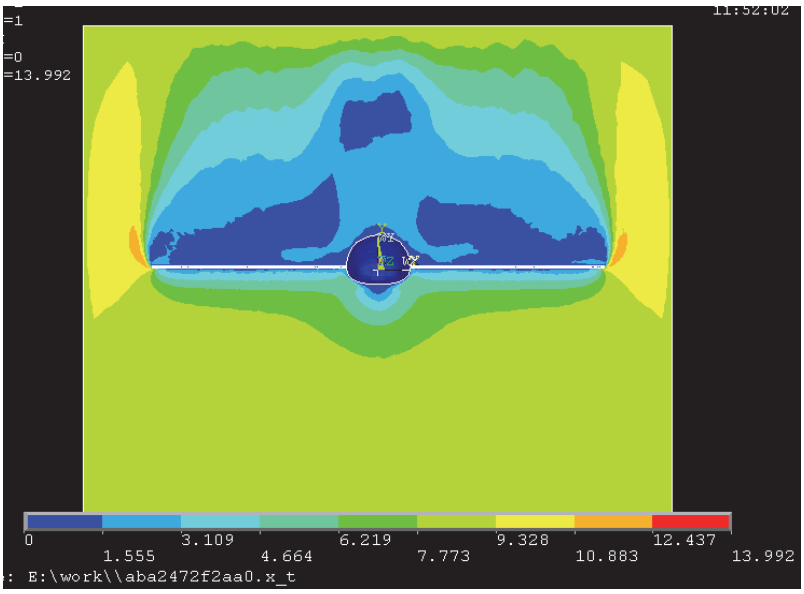


Рис. 3. Пример распределения скоростей обтекания
в плоскости, проходящей через центр тяжести самолета

Обработка результатов решений позволяет вносить в рассматриваемую конструкцию неочевидные технические решения, а применение кластерных систем позволяет существенно увеличить качество проектирования и поднять его на новый уровень.

Список литературы

1. Шингель Л.П. Особенности обтекания крыла легкого самолета // Известия Самар. науч. центра Рос. акад. наук. – 2012. – Т. 14, № 4(5) – С. 1391–1393.

2. Шингель Л.П. Сравнение продольного и поперечного обтекания крыла летательного аппарата // Известия Самар. науч. центра Рос. акад. наук. – 2013. – Т. 15, № 6(2). – С. 538–540.

3. Аэродинамика / Н.Ф. Краснов, В.Н. Кошевой, А.Н. Данилов, В.Ф. Захаренко – М.: Высшая школа, 1968. – 772 с.

4. ANSYS Solutions. Рус. ред. – 2005. – № 1 (1). – С. 12.

В.А. Щапов[1, 2], Г.Ф. Масич[2]

[1] Пермский национальный исследовательский политехнический университет
[2] Институт механики сплошных сред УрО РАН, Пермь

АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ОБРАБОТКИ ИНТЕНСИВНОГО ПОТОКА СТРУКТУРИРОВАННЫХ ДАННЫХ НА СУПЕРКОМПЬЮТЕРАХ

Разработана модель и программная архитектура ввода структурированного потока экспериментальных данных в удаленный суперкомпьютер. Показаны инфраструктура системы распределенной обработки данных и результаты измерений, иллюстрирующие работоспособность разработанных архитектурных решений.

Введение

В предлагаемой вниманию работе исследуется процесс ввода интенсивного потока экспериментальных данных в суперкомпьютер и разрабатывается технология параллельной передачи данных. Обычным способом обработки данных является их сохранение на внешних носителях и последующая обработка на компьютере при экспериментальной установке. Однако не все получаемые в экспериментах данные можно обработать на месте, поскольку требуется слишком большая вычислительная мощность и/или система хранения данных.

В случае обработки больших объемов данных на суперкомпьютере классический режим имеет три этапа: 1) загрузка данных в хранилище суперкомпьютера; 2) обработка данных на суперкомпьютере; 3) выгрузка результатов обработки из хранилища суперкомпьютера. Загрузка-выгрузка данных в/из хранилища (этапы 1 и 3) и последующая обработка (этап 2) выполняются либо с использованием протоколов передачи файлов (*FTP/GridFTP* и *SCP*), либо при прямом доступе к хранилищу данных при помощи протоколов работы с файловой системой (*CIFS* и *NFS/pNFS*).

Нами разрабатывается архитектурное решение, при котором элементы структурированного потока данных вводятся в вычислительные узлы удаленного суперкомпьютера в темпе их генерации и обрабатываются параллельно согласно модели память – память, минуя систему хранения данных.

Под структурированным потоком понимается последовательность отдельных сообщений (измерений), которые могут обрабатываться прикладными алгоритмами независимо друг от друга, и, следовательно, допускается их параллельная обработка. Одним из примеров такого потока

данных являются пары изображений от камер в установках *PIV* [1], по перемещению трассеров в которых определяется векторное поле.

Однако общепризнанное несоответствие между вычислительной производительностью и компонентами ввода-вывода высокопроизводительных систем текущего поколения сделал ввод-вывод наиболее узким местом. И одним из основных источников ухудшения совокупной производительности территориально распределенных высокоскоростных приложений является плохая *end-to-end* производительность повсеместно используемого протокола *TCP*, а основным механизмом повышения пропускной способности является параллельная передача (*GrtidFTP*, *pNFS*).

В этом случае на первый план выходит необходимость обеспечения эффективной передачи данных на вычислительные узлы удаленного суперкомпьютера и решение задачи распределения элементов потока данных по вычислительным узлам.

Модель обработки потока данных

Разработанная модель обработки потока данных основана на представлении потока данных от *источника* в виде единой очереди сообщений (рис. 1), которые распределяются менеджером очередей по вычислительным узлам по запросам от вычислительных узлов в порядке *FIFO* (*First In*, *First Out*).

Инициаторами обмена данными являются оконечные системы – источник данных и вычислительные узлы, занимающиеся обработкой данных.

При этом передача данных, в ответ на запросы, может производиться параллельно, что позволяет компенсировать недостатки транспортных протоколов и работать в условиях несимметричных каналов связи, когда скорость потока исходных данных больше, чем скорость подключения к сети передачи данных каждого вычислительного узла.

Выделение задачи диспетчеризации в отдельный слой позволило отказаться от межузлового обмена данными на стороне суперкомпьютера, что дает следующие преимущества [2]:

– автоматическая балансировка нагрузки по вычислительным узлам – более быстрые узлы будут чаще посылать запросы новых данных и, таким образом, будут получать больше данных для обработки (при этом, диспетчер будет иметь полную информацию о свободных узлах, поэтому, при необходимости, могут быть реализованы и другие стратегии выбора узла для отправки следующего сообщения);

– возможность изменять число используемых вычислительных узлов во время проведения эксперимента;

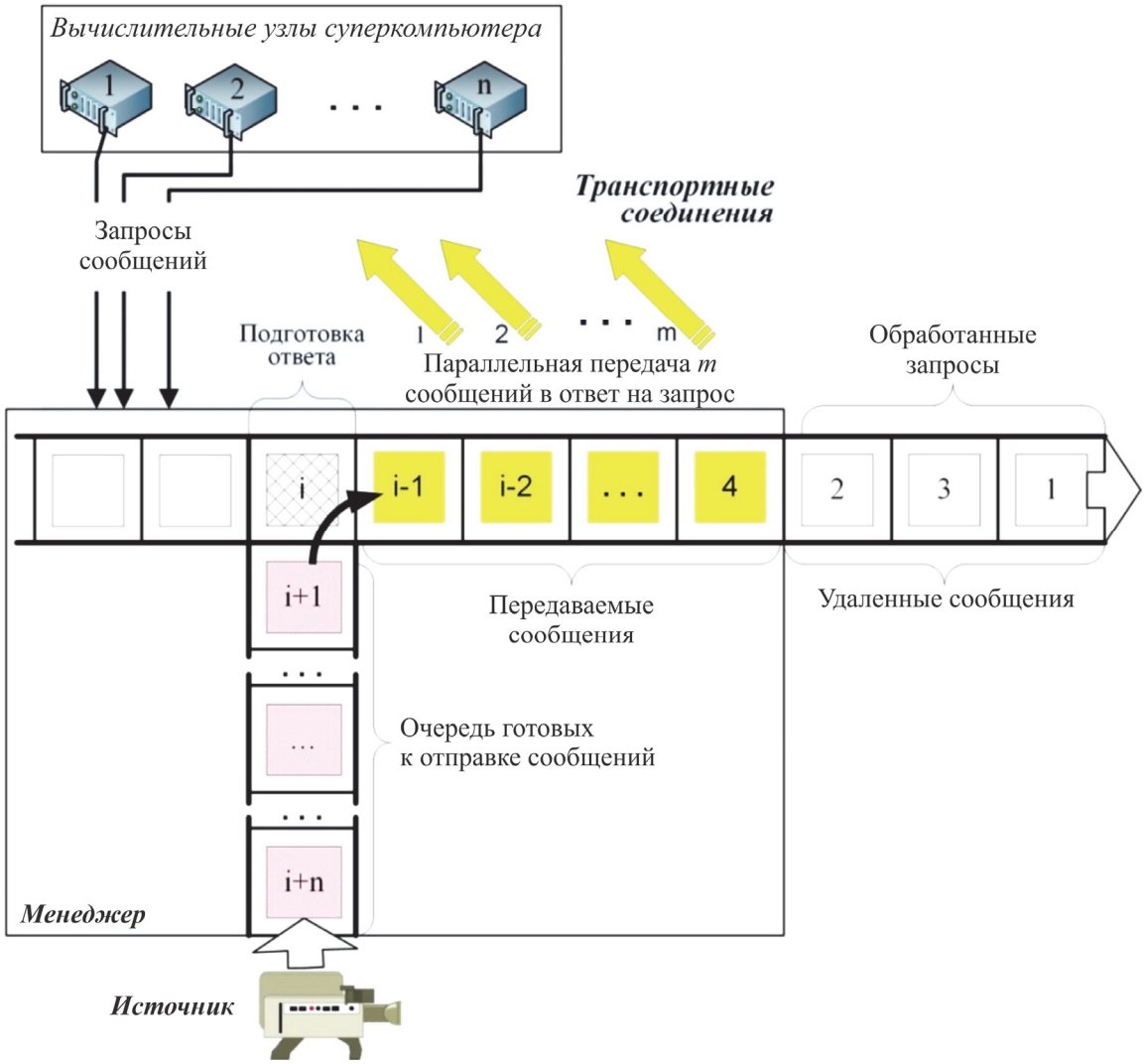


Рис. 1. Модель распределения сообщений по вычислительным узлам

– возможность использовать вычислительную мощность нескольких суперкомпьютеров;

– минимизация потери данных в случае выхода из строя одного или нескольких вычислительных узлов (в худшем случае потеряется только текущее обрабатываемое измерение сбойного узла).

Инфраструктура

Апробация разрабатываемых архитектурных решений выполняется на примере задачи обработки интенсивного потока данных от экспериментальной установки *PIV* (ИМСС УрО РАН, Пермь) на суперкомпьютере «УРАН» (ИММ УрО РАН, Екатеринбург) в рамках проекта «Распределенный *PIV*», являющегося одним из компонентов создаваемой на Урале киберинфраструктуры [3], и на тестовой петле Пермь – Екатеринбург – Пермь, обеспечивающей канал 10 Гбит/с, протяженностью 900 км между тестовым сервером и суперкомпьютером *Triton* в ИМСС УрО РАН (рис. 2).

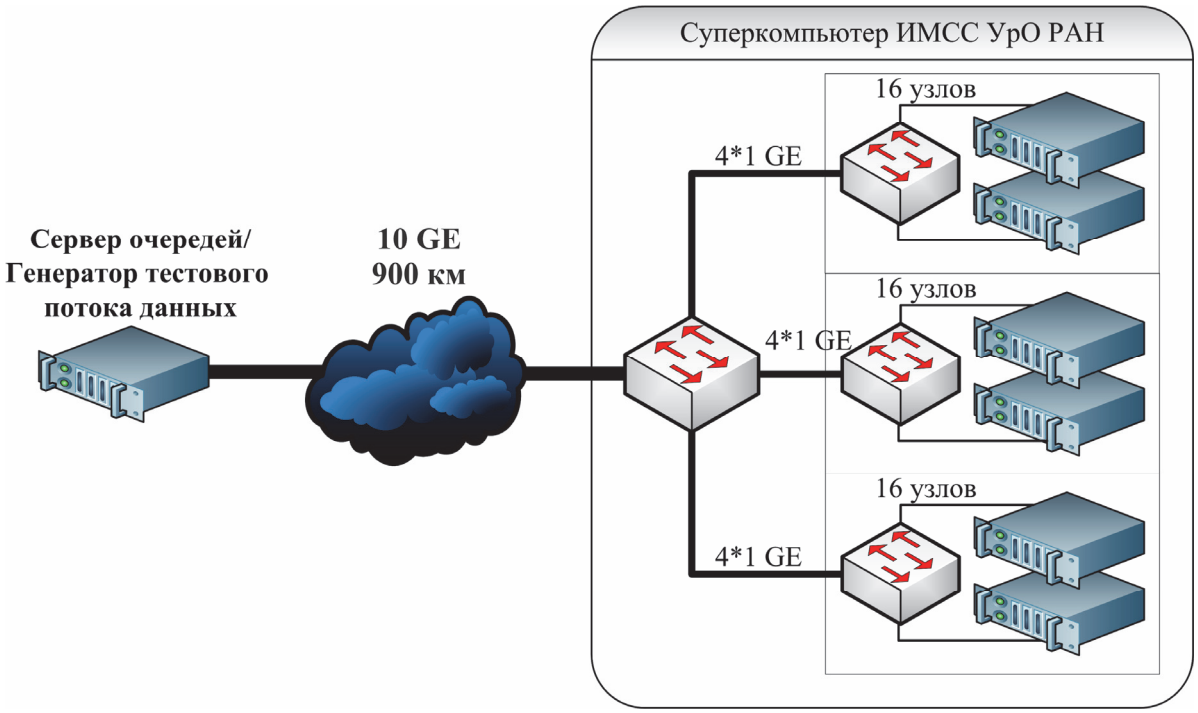


Рис. 2. Схема подключения суперкомпьютера *Triton* к экспериментальному стенду

Для соединения *источника* с *суперкомпьютером* используются каналы связи научно-образовательной оптической магистрали со спектральным уплотнением каналов, создаваемой в рамках проекта *Initiative GIGA UrB RAS* [3]. Для трассы Пермь – Екатеринбург с оптической длиной 456 км расчетное время распространения сигнала равно 2,28 мс. Эта величина задержки экспериментально подтверждается измерением *RTT* (*Round Trip Time*) между пограничными коммутаторами в Перми и Екатеринбурге, которая с учетом задержек в коммутационном облаке (*L2 OSI RM*)

rtt min/avg/max/mdev = 5.285/5.300/5.341/0.102 мс (канал связи 1 Гбит/с);

rtt min/avg/max/mdev = 5.198/5.210/5.237/0.077 мс (канал связи 10 Гбит/с).

Менеджер/диспетчер [2, 4] установлен на отдельном сервере *HP ProLiant DL360p Gen8* (*2x Intel Xeon CPU E5-2660*, 2.20 ГГц, 16 потоков; *RAM* 128 Гб; операционная система *CentOS 7.0*) вблизи установки *PIV*, имеющем 1-10 *GE* интерфейсы. Менеджер *SciMQ* является основной частью реализованного комплекса специализированного программного обеспечения *middleware*. В паре с менеджером *SciMQ* работают приложения уровней генерации и обработки данных (оконечных систем) по протоколу *SciMP*. Приложения оконечных систем взаимодействуют с сервером очередей через *API*, предоставляемый клиентской библиотекой. Все компоненты комплекса программного обеспечения разработаны на языке программирования *C++* с использованием библиотек *Boost*.

Размеры окна для используемых в *SciMP* транспортных протоколов с обратной связью определяются посредством вычисления *BDP* = = *bandwidth*RTT*, который равен 0,6625 или 6,5125 Мбайт для скоростей 1 Гбит/с (*RTT* = 5,3 мс) и 10 Гбит/с (*RTT* = 5,21 мс) соответственно. Для эффективной работы алгоритмов управления перегрузкой в *TCP* требуется, чтобы размер окна перегрузки *cwnd* был не меньше *BDP*.

Исследование работы программного обеспечения при обработке длительных по времени потоков данных

Для изучения стабильности работы системы во времени было проведено тестирование. Для этого сервер очередей был подключен на скорости 10 Гбит/с к внутреннему коммутатору интерконнекта суперкомпьютера Triton ИМСС УрО РАН через тестовую петлю Пермь – Екатеринбург – Пермь (900 км).

Поток исходных данных состоял из сообщений размером 16 Мбайт, которые передавались каждые 20 мс, порождая трафик со скоростью 6,4 Гбит/с. В эксперименте было передано 1 800 000 сообщений общим объемом 27,5 Тбайт. За получение данных отвечали 376 процессов, запущенных на суперкомпьютере ИМСС УрО РАН.

Результаты эксперимента показаны на рис. 3.

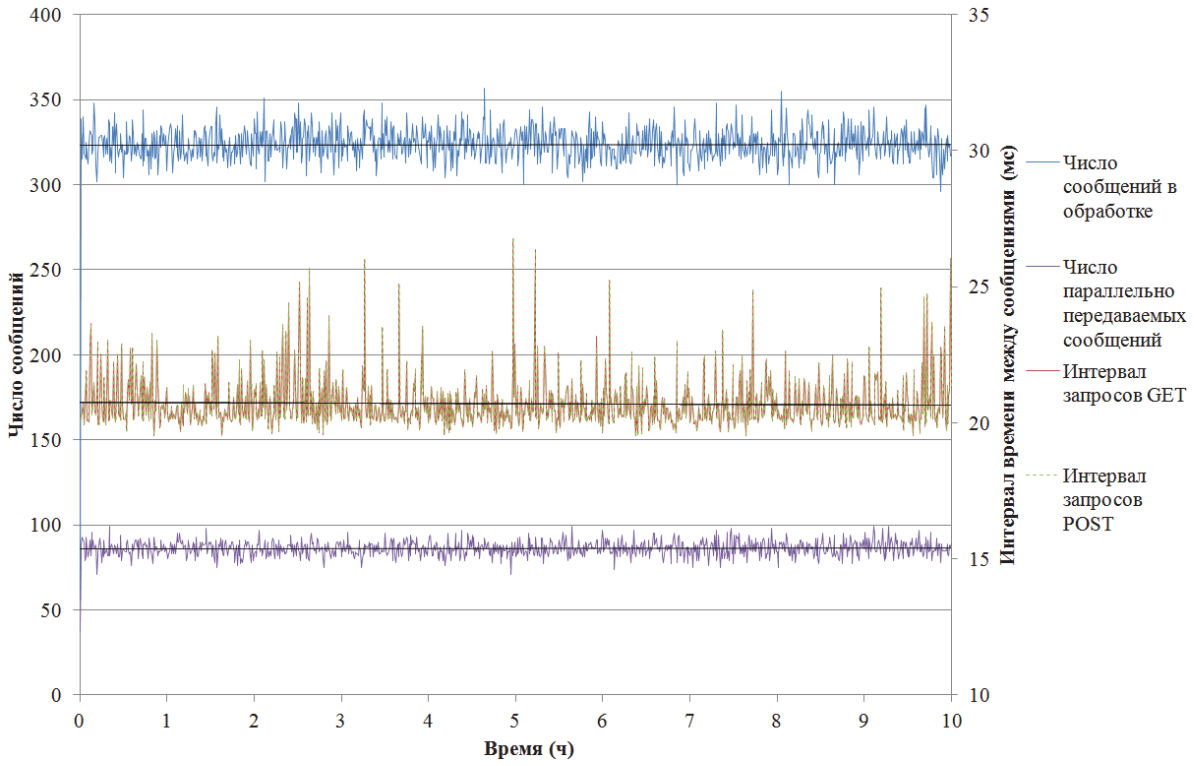


Рис. 3. Графики изменения числа сообщений в обработке, числа параллельно передаваемых сообщений и средних интервалов между поступлением исходных сообщений и отправкой сообщений клиентам

Из приведенных графиков следует:

− система работает без перегрузок и обеспечивает обработку потока данных без дополнительных задержек и потерь сообщений;

− средние параметры работы системы не меняются во времени;

− программное обеспечение способно передавать продолжительные по времени интенсивные потоки данных.

Анализ данных эксперимента показал, что в условиях несимметричных каналов связи (сервер очередей – 10 Гбит/с, вычислительный узел – 1 Гбит/с) и использования технологии агрегации каналов (*LACP*) внутри суперкомпьютера (четыре канала по 1 Гбит/с) уменьшается эффективность алгоритмов управления потоком *TCP*. При этом алгоритмы распределения трафика по физическим линиям связи в *LACP*-группе могут приводить к тому, что эффективная пропускная способность группы существенно снижается. Так, наблюдались ситуации, когда реально использовались только один или два канала в случае передачи данных на некоторое число вычислительных узлов (например, на 4–8 из 16), подключенных к исследуемой *LACP*-группе.

Заключение

Разработанная технология предоставляет принципиально новый инструмент проведения уникальных физических экспериментов в исследовательских лабораториях и на производстве. Отличительной особенностью созданной технологии является асинхронная модель передачи потока данных от источника в удаленный суперкомпьютер, основанная на идее параллельного прямого ввода структурированного потока данных в вычислительные узлы суперкомпьютера. Скоростные каналы передачи данных позволяют связать место проведения измерений с суперкомпьютерами, устраняя затраты на наращивание вычислительных ресурсов на местах.

Областью применения являются измерительные системы нового поколения, включающие высокотехнологичное измерительное оборудование на местах и высокопроизводительную вычислительную технику в суперкомпьютерных центрах. Начаты работы по внедрению разработанных технологий в проекте по созданию технологической платформы для экспериментальных и вычислительных исследований быстропротекающих процессов гидроупругости на базе связки измерительной системы с суперЭВМ.

Исследование проводится при поддержке РФФИ (гранты № 14-07-96001 и № 14-07-96003).

Список литературы

1. Обработка на супервычислителе потока экспериментальных данных / Р.А. Степанов, А.Г. Масич, А.Н. Сухановский, В.А. Щапов, А.С. Игумнов, Г.Ф. Масич // Вестник УГАТУ. – Уфа, 2012. – Т. 16, № 3(48). – С. 126–133.

2. Щапов В.А., Масич А.Г., Масич Г.Ф. Модель потоковой обработки экспериментальных данных в распределенных системах // Вычислительные методы и программирование. – 2012. – С. 139–145.

3. Масич А.Г., Масич Г.Ф. От «Инициативы GIGA UrB RAS» к Киберинфраструктуре УрО РАН // Вестник Пермского научного центра / ПНЦ УрО РАН. – Пермь, 2009. – № 4. – С. 41–56.

4. Shchapov V., Masich A. Protocol of High Speed Data Transfer from Particle Image Velocimetry System to Supercomputer // Proc. of The 7th International Forum on Strategic Technology (IFOST 2012), September 18–21 / Tomsk Polytechnic University. – Tomsk, 2012, – Vol. 2. – P. 653–657. DOI: 10.1109/IFOST.2012.6357642

И.И. Юсипов, Т.В. Лаптева, М.В. Иванченко

Нижегородский государственный университет им. Н.И. Лобачевского

ДИНАМИКА ВЗАИМОДЕЙСТВУЮЩИХ ЧАСТИЦ В ПРОСТРАНСТВЕННО-НЕОДНОРОДНЫХ РЕШЕТКАХ

Предложен новый численный метод интегрирования математических моделей квантовых систем, находящихся под периодическим внешним воздействием, который позволяет корректно провести Флоке-анализ заданного числа состояний стационарного гамильтониана. Реализован параллельный численный алгоритм для исследования систем с большим числом состояний, продемонстрирована его хорошая масштабируемость.

Андерсоновская локализация – явление, возникающее при распространении волн в среде с пространственными неоднородностями и состоящее в остановке, исчезновении бегущих волн вследствие многократного рассеяния на неоднородностях и деструктивной интерференции; при этом колебания приобретают характер стоячей волны, сконцентрированной (локализованной) в ограниченной области пространства [1, 2]. Экспериментальные наблюдения этого явления относятся к целому спектру различных систем, включая конденсаты ультрахолодных атомов и световые пучки в оптических кристаллах. Вместе с тем открытым остается вопрос о влиянии нелинейности или взаимодействия между квантовыми частицами

на локализацию. Дополнительный интерес подобные фундаментальные вопросы многочастичной квантовой физики представляют для прикладных исследований в области квантовых вычислений.

Аналитическое и численное исследование квантовых многочастичных волн в системах с беспорядком очень сложно и трудоемко, хотя в последнее время и были достигнуты определенные успехи [3]. В связи с этим весьма привлекательным путем является изучение динамики нескольких взаимодействующих частиц, в предельном случае – двух. Понижение размерности позволяет реализовать численные эксперименты на современных вычислительных кластерах, в том числе с применением параллельных вычислений. Целью данной работы являлось численное моделирование динамики двух взаимодействующих частиц в пространственно-неоднородных решетках, а также параллельная реализация для исследования локализации и распространения волновых пакетов на больших пространственно-временных масштабах.

В данной работе рассматривается математическая модель двух частиц (бозонов), взаимодействующих на решетке с двумя видами неоднородностей: квазипериодической и случайной.

Гамильтониан модели имеет следующий вид:

$$\hat{H} = \sum_j \left[\hat{b}_{j+1}^{+} \hat{b}_j + \hat{b}_j^{+} \hat{b}_{j+1} + \varepsilon_j \hat{b}_j^{+} \hat{b}_j + \frac{U}{2} \hat{b}_j^{+} \hat{b}_j^{+} \hat{b}_j \hat{b}_j \right], \qquad (1)$$

где $\hat{b}_j^{+}$ и $\hat{b}_j$ – операторы образования и разрушения неразличимых бозонов на j-м узле решетки, а U – сила взаимодействия между частицами. Энергии на каждом узле решетки являются либо квазипериодической функцией координаты $\varepsilon_j = W\cos(2\pi\alpha j + \beta)$, $\alpha = (\sqrt{5}-1)/2$, несоизмеримой с периодом решетки, либо случайными некоррелированными величинами с равномерным распределением на интервале $\varepsilon_j \in \left[-\frac{W}{2}, \frac{W}{2} \right]$.

Обозначим вакуумные состояния $|0\rangle$ и в базисе $|j,k\rangle \equiv \hat{b}_j^{+} b_k^{+} |0\rangle$ перепишем волновую функцию двух взаимодействующих частиц $\psi = \sum_{j,k} |j,k\rangle$. Неразличимость частиц дает дополнительное условие $\psi_{j,k} = \psi_{k,j}$. Подставляя в уравнение Шредингера $i\dot{\psi} = \hat{H}\psi$, перейдем к задаче на виртуальной двумерной решетке:

$$i\dot{\varphi}_{i,k} = \varepsilon_{i,k}\varphi_{i,k} + \sum\nolimits_{\pm} \left(\varphi_{j,k\pm1} + \varphi_{j\pm1,k} \right), \qquad (2)$$

где $\varepsilon_{i,k} = \varepsilon_j + \varepsilon_k + U\delta_{i,k}$ и $\delta_{i,k}$ – символ Кронекера. В отсутствии взаимодействия между частицами максимальная длина локализации со случайной

модуляцией имеет следующую зависимость от параметра пространственной неоднородности: $\xi \approx \frac{96}{W^2}$, а в случае квазипериодического потенциала – $\xi \approx \frac{1}{\mathrm{In}\left(\frac{W}{2}\right)}$.

В численном интегрировании использовались уравнения для действительной и мнимой части волновой функции ψ:

$$\psi_{n,m} = \left(\frac{q_{n,m} + ip_{n,m}}{\sqrt{2}} \right), \tag{3}$$

$$\begin{cases} -\dot{p}_{n,m} = \left(\varepsilon_{n,m} + \delta_{n,m} U\right) q_{n,m} + q_{n-1,m} + q_{n+1,m} + q_{n,m-1} + q_{n,m+1}, \\ \dot{q}_{n,m} = \left(\varepsilon_{n,m} + \delta_{n,m} U\right) p_{n,m} + p_{n-1,m} + p_{n+1,m} + p_{n,m-1} + p_{n,m+1}. \end{cases} \tag{4}$$

Для численного решения последней системы предпочтительны интеграторы, с хорошей точностью сохраняющие гамильтониан; наш выбор остановился на симплектическом *PQ*-методе [4].

Основным фактором, определяющим вычислительную трудоемкость задачи и использование высокопроизводительных вычислений, является необходимость численного моделирования динамических уравнений (4) для оценки асимптотики временной зависимости эффективной ширины, дисперсии волнового пакета D и числа эффективно возбужденных узлов решетки P_n. На больших временах интегрирования, однако, распространяющийся волновой пакет может достигать границ решетки и распространение возмущений по системе искажается или вовсе прекращается. Таким образом, численное моделирование на более длительных временах неразрывно связано с большими размерами системы. В целом трудоемкость алгоритма может быть выражена как $O(N^2 T)$, где N – размерность решетки; T – количество временных итераций. Нижняя граница допустимых размеров решетки вытекает из сравнения с длиной одночастичной локализации $N \gg \xi$, что, с учетом интересующего нас режима $W \leq 1$, отвечает $N \gg 100$ и, соответственно, $N^2 \gg 10^4$.

Принимая во внимание вышеизложенное, для проведения численных экспериментов разработали программный комплекс с использованием некоторых технологий параллельного программирования. Значительное ускорение было получено при переносе приложения на графические процессоры при помощи технологии *CUDA C* (рис. 1).

Следующим шагом стала версия для работы с распределенной памятью посредством технологии *MPI*, а в настоящее время реализуется возможность использования *GPU* как сопроцессоров на узлах кластера «Ло-

бачевский»[1]. Дальнейшей задачей, стоящей перед нами, является уменьшение количества пересылаемых данных между узлами, например с использованием альтернативных структур хранения данных.

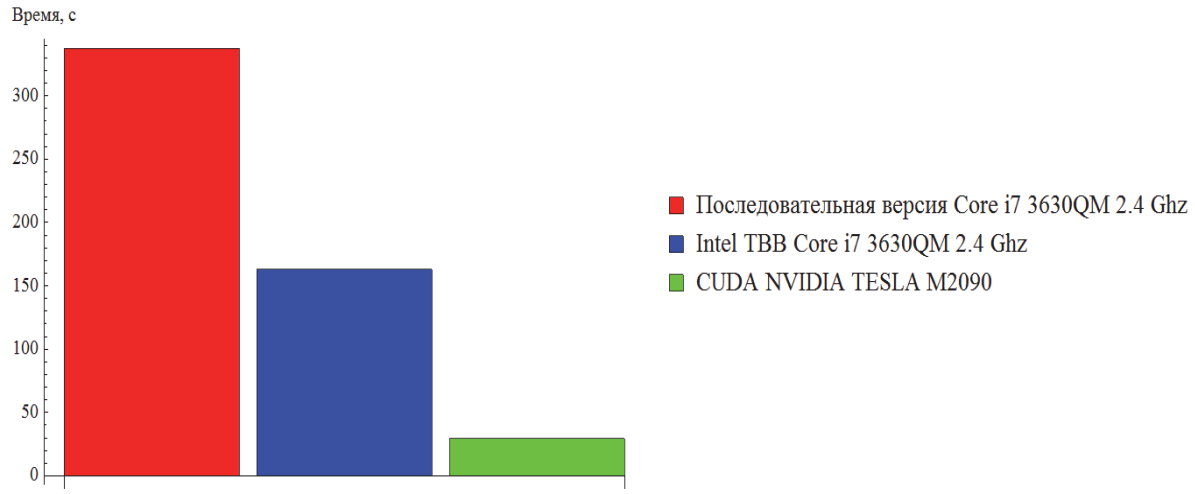


Рис. 1. Сравнение реализаций на модельном эксперименте

Тем не менее уже реализованные параллельные решения позволили получить следующие практические результаты. Для квазипериодической модуляции делокализация взаимодействующих частиц была подтверждена в некоторых предполагаемых ранее областях параметров [5]. Кроме того, во всех наблюдавшихся ранее областях делокализация была подтверждена на существенно больших пространственных и временных масштабах. Это позволило провести численную оценку характера распространения волновых пакетов, показав близость режима к баллистическому (рис. 2, *а–в*).

Для случайного потенциала установлено, что наличие взаимодействия между частицами способствует делокализации в системе (рис. 3) [6]. Распространение возмущения носит субдиффузионный характер.

Проведенные исследования представляют собой яркое свидетельство того факта, что особенности нелинейной волновой динамики в неупорядоченных решетках проявляются уже при двух взаимодействующих частицах.

Дальнейшие исследования могут проводиться в области распространения результатов на всю плоскость параметров и исследовании влияния увеличения числа взаимодействующих частиц. Еще один очень интересный вопрос касается проблемы делокализованных состояний двух взаимодействующих частиц в присутствии большого числа частиц, вплоть до некоторой конечной плотности.

[1] Суперкомпьютер ННГУ «Лобачевский». – URL: http://hpc-education.unn.ru/-ru/ресурсы.

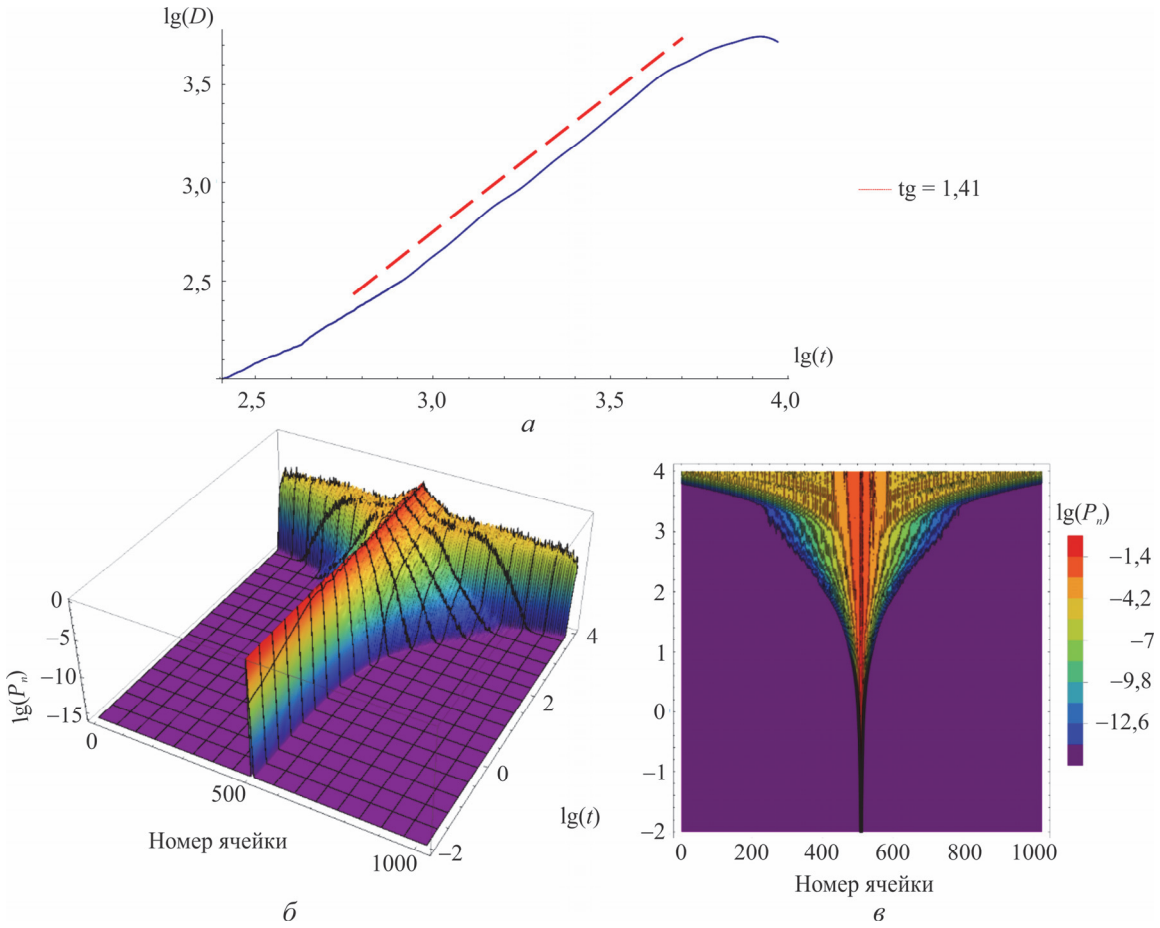


Рис. 2. Динамика разрастания ширины пакета во времени в случае квазипериодической пространственной неоднородности. Суббалистический тип распределения

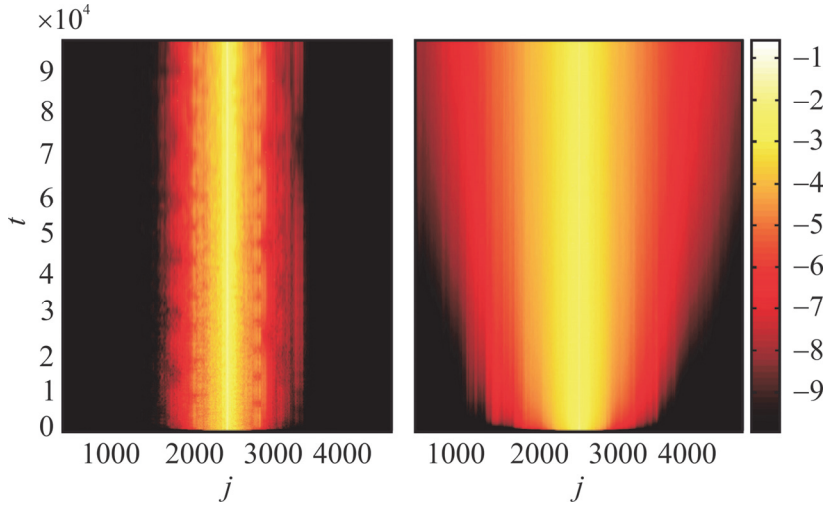


Рис. 3. Локализация и делокализация волновых пакетов двух частиц в случае пространственного беспорядка. Цветом обозначен десятичный логарифм вероятности нахождения частиц в j-м узле в зависимости от времени: слева – $U = 0$, справа – $U = 2$ [6]

Работа поддержана грантом МОН РФ, соглашение № 02.B.49.21.0003.

Список литературы

1. Kramer B., MacKinnon A. Localization – theory and experiments // Rep. Prog. Phys. – 1993. – Vol. 56. – P. 1469.

2. Aubry S., Andre G. Analyticity breaking and Anderson localization in incommensurate lattices // Proc. Isr. Phys. Soc. – 1980. – Vol. 3. – P. 133.

3. Basko D.M., Aleiner I.L., Altshuler B.L. On the problem of many-body localization // Ann. Phys. (N.Y.). – 2006. – Vol. 321. – P. 1126.

4. Delocalization of wave packets in disordered nonlinear chains / Ch. Skokos, D.O. Krimer, S. Flach, S. Komineas // Phys. Rev. E. – 2009. – Vol. 79. – P. 056211.

5. Flach S., Ivanchenko M., Khomeriki R. Correlated metallic two-particle bound states in quasiperiodic chains // EPL. – 2012. – Vol. 98. – P. 66002.

6. Ivanchenko M.V., Laptyeva T.V., Flach S. Quantum chaotic subdiffusion in random potentials // Phys. Rev. B. – 2014. – Vol. 89(R). – P. 060301.

С.Л. Калюлин, А.Ф. Шмаков, В.Я. Модорский

Пермский национальный исследовательский политехнический университет

ОПТИМИЗАЦИЯ КРУПНОГАБАРИТНОЙ КОНСТРУКЦИИ ИЗ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ В ПРИКЛАДНОМ ПАКЕТЕ IOSO PM

Работа посвящена оптимизации конструкции крупногабаритного вентилятора, реализованной с использованием трехшагового алгоритма работы в IOSO PM. Проведена оптимизация внутренней трубы лонжерона при допустимой толщине стенки трубы. Параметры оптимизации – геометрические характеристики стеклопластикового лонжерона лопасти вентилятора.

Введение

В энергетике для охлаждения рабочего тела широко применяются осевые вентиляторы для градирен, которые необходимы для создания искусственной тяги систем оборотного водоснабжения.

В Российской Федерации насчитывается несколько тысяч градирен различных типоразмеров с составляющими, как правило, импортного производства. Одним из их основных элементов является вентилятор, оптимизация элементов которого представлена в данной работе [1–3]. Интеграция требуемых моделей в единый расчетный блок и автоматизация процесса

поиска альтернативных оптимальных технических решений возможна с использованием пакета оптимизации IOSO PM.

Трехшаговый алгоритм работы IOSO PM позволяет исследователю повысить эффективность своей работы за счет уменьшения времени и стоимости разработки проекта.

При этом обеспечивается:

- Поиск нетрадиционных решений, обеспечивающих высокие значения показателей эффективности.
- Снижение рисков получения и внедрения неэффективных решений.
- Повышение качества и надежности разрабатываемой системы.

Схема поиска оптимизационных решений в пакете IOSO PM представлена на рис. 1.

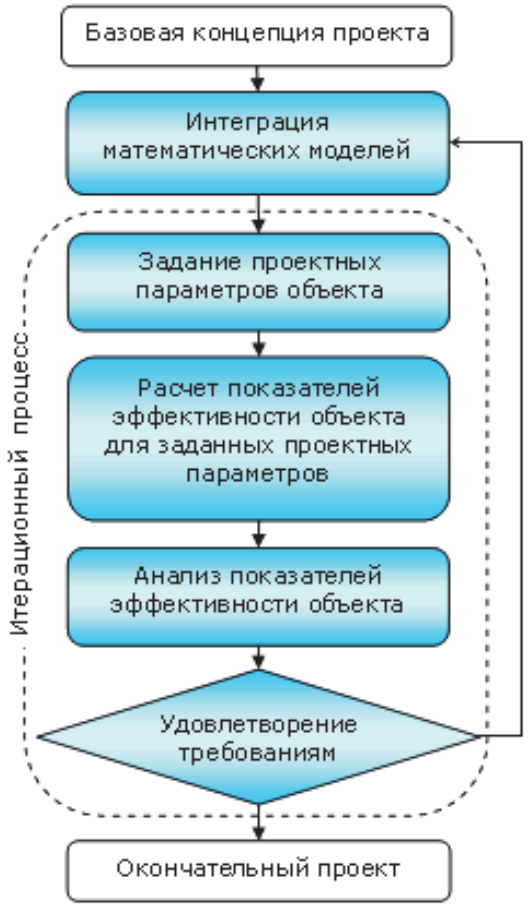


Рис. 1. Блок-схема поиска
оптимизационных решений в IOSO PM

Физическая модель вентилятора для оценки НДС

Сформулирована следующая физическая модель:

Учитывается направление выкладки слоев. Материал конструкции – стеклопластик. Рассчитывается одна лопасть (1/4 часть вентилятора).

Платформа выполнена из стали. Материал рассматривается в нелинейной постановке.

Учитывается гравитация, инерционные нагрузки. Воздействие газодинамических нагрузок передается из газодинамического расчета. Расчет ведется в трехмерной постановке в масштабе 1:1.

Решение задачи о динамическом деформировании трехмерной неоднородной модели вентилятора находится численно с использованием метода конечных элементов в среде ANSYS WorkBench, который основывается на нескольких последовательных шагах, а именно: построение геометрии модели; создание конечно-элементной сетки конструкции (рис. 2); приложение к сетке нагрузок, начальных и граничных условий; решение системы линейных алгебраических уравнений с целью получения значений узловых неизвестных; нахождение по известным узловым перемещениям значений напряжений и деформаций.

Условия равновесия и совместимости деформаций выполняются только в узловых точках – точках соединения конечных элементов. Каждый элемент является частью заменяемой среды, т.е. сплошное тело лишь условно делится на отдельные элементы конечных размеров. Выделенный элемент имеет те же физические свойства и геометрические характеристики, что и рассматриваемая конструкция в месте расположения элемента. Все внешние силы считаются приложенными в узлах по направлению их возможных перемещений.

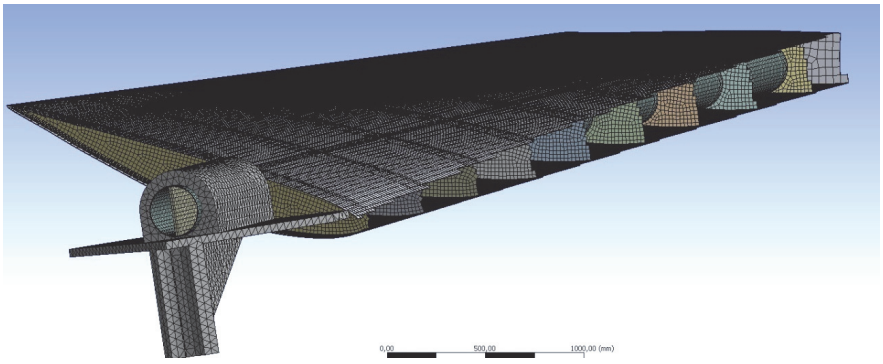


Рис. 2. Конечно-элементная модель

При задании опций расчета следует руководствоваться следующими рекомендациями:

– учет влияния напряженного состояния на жесткость системы делается в следующих случаях:

• если можно ожидать, что при проведении расчета с опцией малых перемещений напряжения в конструкции значительно влияют на ее жесткость (как в случае тонкой круглой пластинки под действием нормального давления);

• если нужно ускорить сходимость решения при анализе больших перемещений;

– по умолчанию влияние напряжений в конструкции на ее жесткость не учитывается;

– опция анализа начальных напряжений включается в том случае, если нужно использовать эту же модель в других типах анализа при наличии предварительного нагружения;

– формирование матрицы масс используется только в том случае, если предполагается нагружение конструкции инерционными силами; можно использовать матрицу сосредоточенных масс или же воспользоваться свойствами того или иного конечного элемента (по умолчанию); при статическом анализе применение матрицы масс мало влияет на точность решения (если сетка достаточно мелкая), однако это важно для динамического расчета при наличии начальных напряжений.

При задании нагрузок (в том числе и перемещений) нужно иметь в виду следующее:

– значения перемещений (три смещения и три угла поворота) задаются:

• на границах модели в точках жесткого опирания модели;

• для указания условий симметрии;

• в точках, где перемещения известны;

– положительное направление – по осям узловой (локальной) системы координат.

Математическая модель НДС вентилятора

Математическая модель крупногабаритного вентилятора описывается следующей системой уравнений:

– геометрические соотношения Коши (6 уравнений, 9 неизвестных),

$$\varepsilon_{ij} = \frac{1}{2}\left(U_{i,j} + U_{j,i}\right); \tag{1}$$

– уравнения равновесия в напряжениях (3 уравнения, 6 неизвестных),

$$\sigma_{ij,j} + f_i = 0; \tag{2}$$

– обобщенный закон Гука (6 уравнений),

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl}; \tag{3}$$

– граничные условия в напряжениях полностью соответствуют условиям на поверхности в напряжениях (3 уравнения, 3 неизвестных),

$$\sigma_{ij,j}\, n_j\Big|_S = T_i, \tag{4}$$

где S – поверхность тела; σ_{ij} – неизвестный тензор напряжений в точках поверхности тела; n_j – проекции вектора единичной нормали $\vec{n}$ к поверх-

ности тела, T_i – проекции вектора поверхностных сил $\bar{T}(X_1, X_2, X_3)$, заданного к поверхности тела S.

Таким образом, система дифференциальных уравнений (1)–(4) включает в себя 18 уравнений и 18 неизвестных.

Аппаратное обеспечение

В качестве аппаратного обеспечения для вычислительных экспериментов использовались ресурсы высокопроизводительного вычислительного комплекса ПНИПУ, имеющего следующие основные технические характеристики:

– 88 вычислительных узлов, 128 четырехядерных процессоров AMD Opteron Barcelona-3 (всего 512 ядер);

– 12 вычислительных модулей GPU NVIDIA Tesla M 2090 (каждый по 512 ядер, 6 Гб);

– 24 вычислительных узла, 48 восьмиядерных процессоров Intel Xeon E5-2680 (всего 384 ядра);

– пиковая производительность 24 TFlops;

– производительность в тестовом пакете Linpack 78 % от пиковой;

– объем системы хранения информации 12 ТБ;

– объем оперативной памяти с 128 Гб/узел;

– 3 файловых сервера Supermicro.

Алгоритм работы IOSO PM

В целом работа в пакете IOSO PM включает в себя три этапа: настройка проекта исследования, постановка задачи оптимизации, оптимизация и анализ результатов.

Первый этап связан с обеспечением интеграции различных программных систем и проведением общих настроек. Второй этап связан с заданием настроек, обеспечивающих оптимальный поиск. Третий этап связан с решением оптимальной задачи и позволяет пользователю осуществлять интерактивное управление проектом.

Получение на каждом итерационном шаге информации обеспечивает ускорение счета при дальнейшем поиске оптимизационного решения.

Постановка задачи оптимизации конструкции вентилятора

В данной работе рассматривается задача оптимизации модельной конструкции вентилятора. Элемент конструкции, для которого рассчитывается оптимальный вариант, – лонжерон из стеклопластика, который рас-

полагается внутри лопасти (рис. 3). В качестве параметра оптимизации была выбрана длина внутренней трубы. Критериями оптимизации являются: минимизация максимальных напряжений по Мизесу и минимизация прогиба трубы.

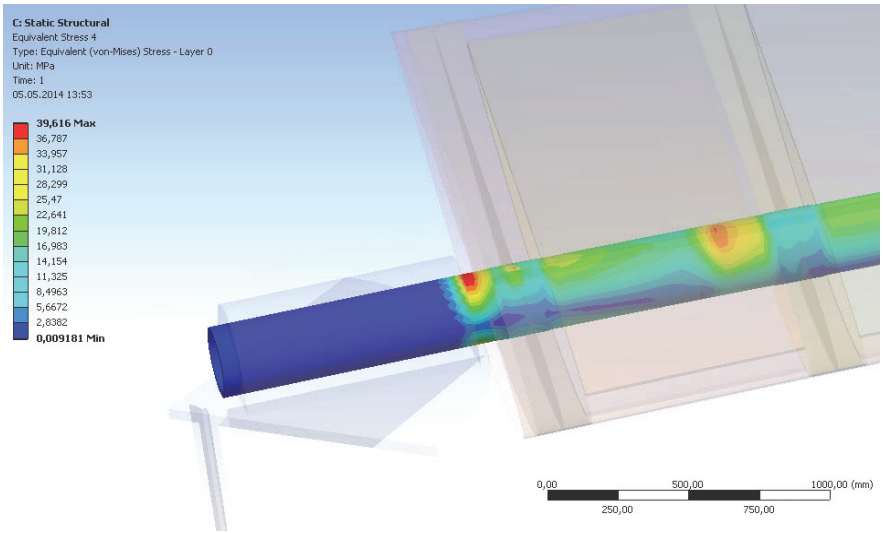


Рис. 3. Модель лонжерона из стеклопластика внутри лопасти вентилятора

Размерность задачи:

- Общее количество переменных: 1.
- Независимых переменных: 1.
- Критериев: 2.
- Ограничений: 0.
- Количество используемых ядер: 64.

Параметры алгоритма оптимизации:

- Максимальное количество итераций: 15 000.
- Максимальное время решения задачи: 100 дней 0 часов 0 минут.
- Точность решения задачи: 0,001.
- Заданное количество Парето-оптимальных решений: 20.

Входные данные для алгоритма оптимизации являются независимыми переменными – длина внутренней трубы из стеклопластика: 1000–9200 мм.

Выходные данные (критерии) – минимизация максимальных напряжений по Мизесу и минимизация максимальных перемещений (прогиба трубы).

Расчеты проводились для трех конструктивных вариантов с использованием инженерного комплекса IOSO PM, интегрированного с ANSYS Workbench.

Анализ результатов расчетов (рис. 4, 5).

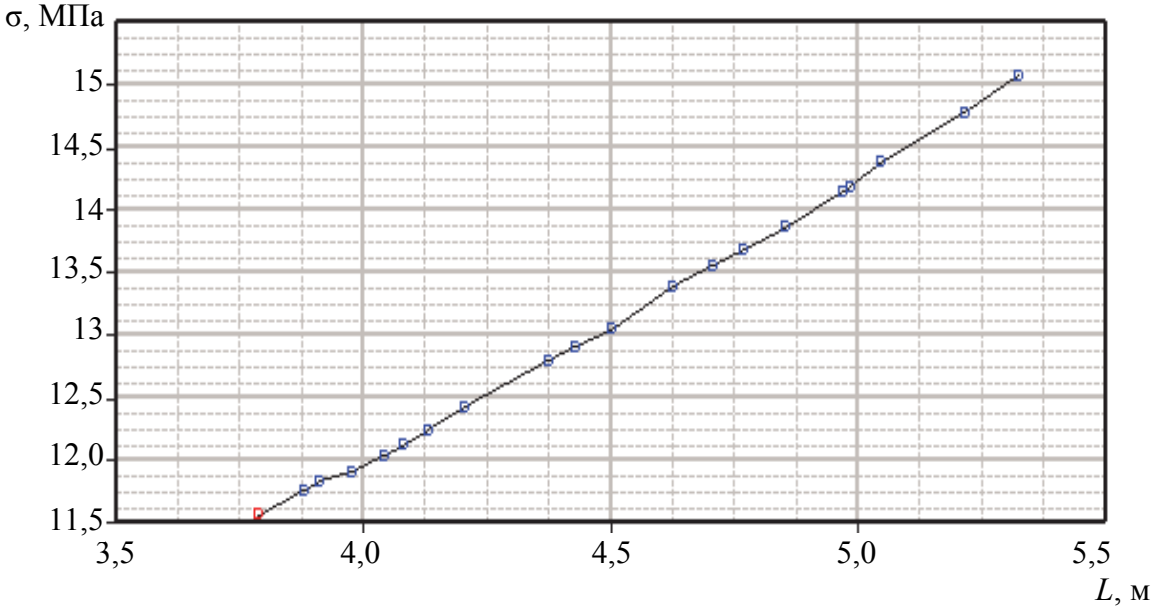


Рис. 4. Зависимость максимальных эквивалентных напряжения по Мизесу от длины внутренней трубы из Парето-оптимального множества при ее толщине 34 мм (IOSO PM)

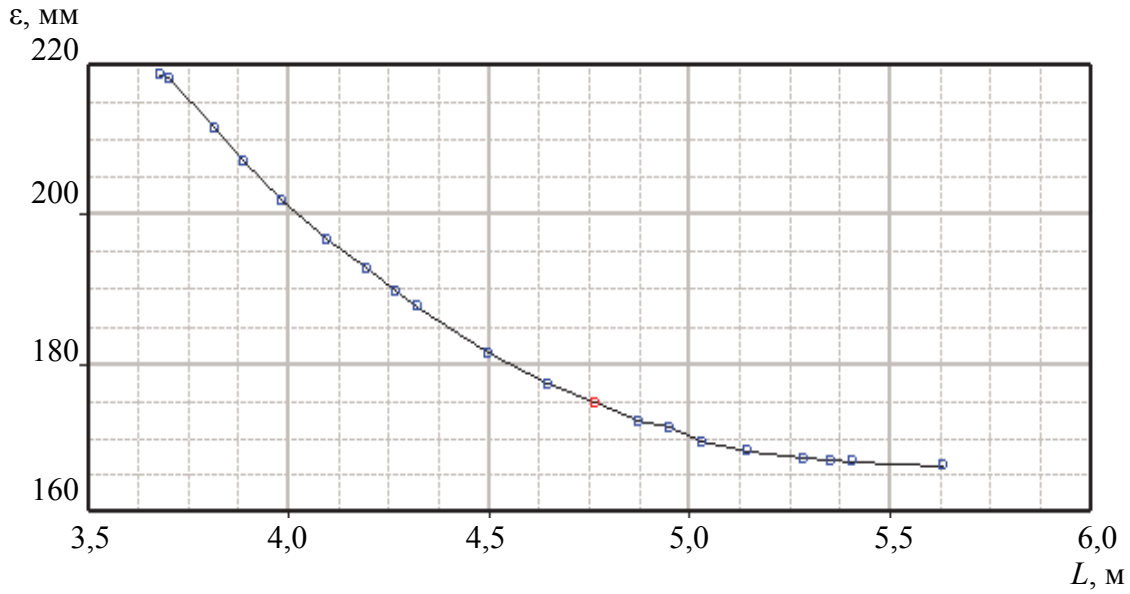


Рис. 5. Зависимость максимального прогиба от длины внутренней трубы из Парето-оптимального множества при ее толщине 34 мм (IOSO PM)

Представлена зависимость диапазона оптимальной длины внутренней трубы от ее толщины (рис. 6).

В результате проведения оптимизационных расчетов задачи изгиба лонжерона можно отметить, что при толщине внутренней трубы 24 мм разброс оптимальной длины внутренней трубы лонжерона существенно выше, чем при 34 и 49 мм. Также при толщине внутренней трубы 34 и 49 мм показатели минимизации максимальных эквивалентных напряжений, макси-

Рис. 6. Зависимость диапазона оптимальной длины внутренней трубы от ее толщины

мального прогиба близки, следовательно, принимать толщину трубы более 34 мм нецелесообразно. Оптимальная длина внутренней трубы лонжерона при толщине 34 мм – от 3679 до 5631 мм.

Список литературы

1. Градирни промышленных и энергетических предприятий / под общ. ред. В.С. Пономаренко. – М.: Энергоатомиздат, 1998. – 376 с.

2. Hensley J.C. Cooling Tower Fundamentals. Compiled from the knowledge and experience of the entire SPX Cooling Technologies staff. SPX Cooling Technologies, Inc. Overland Park, Kansas USA.

3. Detlev G. Kröger. Air-cooled Heat Exchangers and Cooling Towers. Tulsa, Oklahoma, PennWell Books, 2004.

Д.В. Лебедев[1], В.А. Перепелкин[2]

[1] Казахский национальный университет им. аль-Фараби, Алматы, Казахстан
[2] Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

РЕАЛИЗАЦИЯ ОДНОМЕРНОЙ КРАЕВОЙ ЗАДАЧИ НЕФТЬ – ВОДА В СИСТЕМЕ ФРАГМЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ LUNA

Кратко представлена технология фрагментированного программирования и реализующие ее язык и система фрагментированного программирования LuNA, на примере решения одномерной краевой задачи нефть – вода. Технология ориентирована на поддержку разработки параллельных программ, реализующих большие численные модели, и их исполнения на суперкомпьютерах. Система LuNA автоматизирует программирование

коммуникаций, распределение ресурсов, управление памятью, синхронизацию доступа к общим данным, динамическую балансировку нагрузки на вычислительные узлы и т.п.

Введение

При параллельной реализации больших численных моделей на суперкомпьютерах возникают проблемы из области системного параллельного программирования, такие как декомпозиция данных и вычислений на части и распределение этих частей по различным узлам мультикомпьютера, обеспечение и синхронизация доступа к распределенным данным, планирование вычислений, обеспечение динамической балансировки нагрузки на процессоры и т.д. [1]. Решение этих проблем важно автоматизировать, так как это повышает уровень программирования, снижает требования к квалификации программиста, обычно повышает качество параллельной программы, увеличивает скорость и снижает трудоемкость разработки параллельной программы, снижает число возникающих ошибок. Особенно это существенно при программировании на неоднородных вычислителях, так как приходится учитывать соответствующую специфику – разные объемы оперативной и кэш-памяти узлов, разную производительность вычислительных узлов, разные характеристики коммуникационных сетей – топологию, пропускную способность, латентность и т.п.

Подобная автоматизация – сложная задача, которую целесообразно решать в ограниченной предметной области и для ограниченного класса вычислителей. Технология фрагментированного программирования и поддерживающая ее система LuNA рассчитаны на автоматизацию параллельной реализации итерационных численных алгоритмов на регулярных сетках и на неоднородные суперкомпьютеры, получающиеся в результате объединения нескольких кластеров в один, например, с помощью коммуникационной среды NumGRID [3].

Описание исходного последовательного алгоритма

Будем рассматривать одномерную краевую задачу нефть – вода. Данная задача описывается следующими уравнениями:

$$\frac{\partial}{\partial x}\left[K_0\left(1+C_f P_h\left(P_0-1\right)\right)\left(\frac{\partial P_0}{\partial x}-\frac{\gamma_0 L}{P_h}\frac{\partial z}{\partial x}\right)\right]+\frac{\partial}{\partial x}\left[K_w\left(\frac{\partial P_0}{\partial x}-\frac{\partial P_{cow}}{\partial x}-\gamma_w\frac{L}{P_H}\frac{\partial z}{\partial x}\right)\right]= \quad (1)$$

$$=C_f P_H\frac{\partial}{\partial\tau}\left(S_0 P_0\right)+\frac{\partial}{\partial\tau}\left[\left(1-C_f P_H\right)S_0\right]+\frac{\mu_w}{\mu_0}\frac{\partial S_w}{\partial\tau}+\frac{\mu_0 L^2}{K\rho_H P_H}\left(q_w+q_0\right),$$

$$\frac{\partial}{\partial x}\left[K_w\left(\frac{\partial P_w}{\partial x}-\gamma_w\frac{L}{P_H}\frac{\partial z}{\partial x}\right)\right]=\frac{\mu_w}{\mu_0}\frac{\partial S_w}{\partial \tau}+\frac{\mu_0 L^2}{K\rho_H P_H}q_w \tag{2}$$

и начальными условиями:

$$\begin{cases} P_0(x,0)=P_0^H(x),\ \ P_w(x,0)=P_w^H(x), \\ S_0(x,0)=S_0^H(x),\ \ S_w(x,0)=S_w^H(x),\ \ 0<x<L, \end{cases} \tag{3}$$

$$\left.\frac{\partial P_l}{\partial x}\right|_{x=0}=0,\left.\frac{\partial P_l}{\partial x}\right|_{x=L}=0,\ l=(0,w).$$

Для решения системы (1)–(3) применим итерационный метод. Самым эффективным является метод простой итерации.

Для этого запишем уравнение (1) в конечно-разностном виде:

$$a_i^{(r-1)}P_{0i-1}^{(r)}-b_i^{(r-1)}P_{0i}^{(r)}+c_i^{(r-1)}P_{0i+1}^{(r)}=-f_i^{(r-1)},\ i=1,N-1. \tag{4}$$

Здесь

$$a_i^{(r-1)}=K_{0i-1/2}^{(l-1)}\left(1+C_fP_H\left(P_{0i-1/2}^{(r-1)}-1\right)\right)+K_{wi-1/2}^{(l-1)},$$

$$c_i^{(r-1)}=K_{0i+1/2}^{(l-1)}\left(1+C_fP_H\left(P_{0i+1/2}^{(r-1)}-1\right)\right)+K_{wi+1/2}^{(l-1)},$$

$$b_i^{(r-1)}=a_i^{(r-1)}+c_i^{(r-1)}+\frac{h^2}{\tau}C_fP_HS_{0i}^{(l-1)},$$

$$f_i^{(r-1)}=\frac{h^2}{\tau}\left[C_fP_H\overline{S}_{0i}\overline{P}_{0i}-\left(1-C_fP_H\right)\left(S_{0i}^{(l-1)}-\overline{S_{0i}}\right)-\frac{\mu_w}{\mu_0}\left(S_{wi}^{(l-1)}-\overline{S_{wi}}\right)\right]-$$

$$-K_{0i+1/2}^{(l-1)}\left(1+C_fP_H\left(P_{0i+1/2}^{(r-1)}-1\right)\right)\frac{\gamma_0 L}{P_H}\left(z_{i+1}-z_i\right)+$$

$$+K_{0i-1/2}^{(l-1)}\left(1+C_fP_H\left(P_{0i-1/2}^{(r-1)}-1\right)\right)\frac{\gamma_0 L}{P_H}\left(z_i-z_{i-1}\right)-$$

$$-K_{wi+1/2}^{(l-1)}\left[P_{cowi+1}^{(r-1)}-P_{cowi}^{(r-1)}+\frac{\gamma_w L}{P_H}\left(z_{i+1}-z_i\right)\right]+$$

$$+K_{wi-1/2}^{(l-1)}\left[P_{cowi}^{(r-1)}-P_{cowi-1}^{(r-1)}+\frac{\gamma_w L}{P_H}\left(z_i-z_{i-1}\right)\right]-h^2\frac{\mu_0 L^2}{K\rho_H P_H}\left(q_w+q_0\right).$$

Для решения системы (4) на каждом итерационном слое r применим метод прогонки [7, 8]. Для завершения процесса необходимо выполнение условия

$$\left\| P_{0i}^{(r)} - P_{0i}^{(r-1)} \right\| \le \varepsilon_1.$$

Следующий шаг – нахождение P_w по формуле $P_w = P_0 - P_{cow}$.

Зная P_w, можно найти q_w по формуле

$$q_w = \frac{KK_w}{\mu_w}\left(\frac{dP_w}{dx} - \gamma_w \frac{dz}{dx} \right),$$

которая следует из закона Дарси.

Затем необходимо определить $S_{wi}^{(l)}$, для этого запишем уравнение (2) в конечно-разностном виде и разрешенном относительно S_{wi}. При этом это значение будет являться итерационным, так как при его нахождении будут использоваться те значения, которые определены на предыдущих шагах итерации.

$$\begin{aligned} S_{wi}^{(l)} = \overline{S}_{wi} + \frac{\mu_0 \tau}{\mu_w h^2}\Bigg[K_{wi+1/2}^{(l-1)}\left(P_{wi+1} - P_{wi} - \gamma_w \frac{L}{P_H}\left(Z_{i+1} - Z_i\right)\right) - \\ - K_{wi-1/2}^{(l-1)}\left(P_{wi} - P_{wi-1} - \gamma_w \frac{L}{P_H}\left(Z_i - Z_{i-1}\right)\right)\Bigg] - \frac{\mu_0^2 \tau L^2}{\mu_w K \rho_w P_H} q_w. \end{aligned} \tag{5}$$

Значения $P_0^{(r)}$, $S_w^{(l)}$ являются промежуточными, для уточнения необходимо подставить найденные значения в коэффициенты системы (4) и заново решить эту систему, затем найденные значения подставить в (5). Процесс остановится при достижении условия

$$\left\| S_w^{(l)} - S_w^{(l-1)} \right\| \le \varepsilon_2. \tag{6}$$

Затем находим $S_0 = 1 - S_w, q_0 = 1 - q_w$.

За первые приближения $P_0^{(0)}$, $S_w^{(0)}$ берутся значения из предыдущего временного слоя P_0 и S_w.

Фрагментация исходного алгоритма и реализация его в системе LuNA

В системе LuNA описанный алгоритм может быть запрограммирован как фрагментированный. Для этого произведем разбиение области пространства на m частей и будем решать исходную задачу внутри каждой подобласти для уравнения (1) методом параллельной прогонки [9], а для уравнения (2) использовать явную схему.

Пусть каждому фрагменту выделено одинаковое количество узлов сетки $m = M/N$, где M – число узлов сетки, N – число фрагментов, индекса-

ция глобальная; если M не делится на N, то для последнего фрагмента число узлов увеличиваем на остаток от деления. Таким образом, на отдельном процессоре с номером j решается лишь часть уравнений системы (1) с номерами от $(j-1)m+1$ до jm, где j – номер фрагмента. Обозначим Po_{jm} через $z_j, j = 0, ..., N$, и представим искомые значения в виде

$$Po_i^{(r)} = z_{j-1} \cdot u_i + z_j \cdot v_i + w_i, \tag{7}$$

где u, v, w – решения следующих систем уравнений:

$$\begin{cases} a_i^{(r-1)} \cdot u_{i-1} - b_i^{(r-1)} \cdot u_i + c_i^{(r-1)} \cdot u_{i+1} = 0, \\ u_{(j-1)m} = 1, u_{jm} = 0, \end{cases}$$

$$\begin{cases} a_i^{(r-1)} \cdot v_{i-1} - b_i^{(r-1)} \cdot v_i + c_i^{(r-1)} \cdot v_{i+1} = 0, \\ v_{(j-1)m} = 0, v_{jm} = 1, \end{cases} \tag{8}$$

$$\begin{cases} a_i^{(r-1)} \cdot w_{i-1} - b_i^{(r-1)} \cdot w_i + c_i^{(r-1)} \cdot w_{i+1} = -f_i, \\ w_{(j-1)m} = 0, w_{jm} = 0, \end{cases}$$

Решения этих трех систем можно найти методом прогонки, причем независимо на каждом из фрагментов. Будем называть эти решения предрешениями, а этот этап решения задачи – этапом нахождения предрешений.

В уравнения с номерами jm подставляем вместо Po комбинации (7). Таким образом, получаем систему трехточечных уравнений для вычисления z_j, имеющую следующий вид:

$$\begin{aligned} &A_j \cdot z_{j-1} - B_j \cdot z_j + C_j \cdot z_{j+1} = -F_j \, j = 1, \ldots, N-1, \\ &-B_0 \cdot z_0 + C_0 \cdot z_1 = -F_0, A_N \cdot z_{N-1} - B_N \cdot z_N = -F_N, \\ &B_0 = b_0 - c_0 \cdot u_1, C_0 = c_0 \cdot v_1, F_0 = f_0 + c_0 \cdot w_1, \\ &A_j = a_{jm} \cdot u_{jm-1}, B_j = b_{jm} - a_{jm} \cdot v_{jm-1} - c_{jm} \cdot u_{jm+1}, C_j = c_{jm} \cdot v_{jm+1}, \\ &F_j = f_{jm} + a_{jm} \cdot w_{jm-1} + c_{jm} \cdot w_{jm+1}, \quad j = 1, \ldots, N-1, \\ &A_N = a_{Nm} \cdot u_{Nm-1}, B_N = b_{Nm} - a_{Nm} \cdot v_{Nm-1}, F_N = f_{Nm} + a_{Nm} \cdot w_{Nm-1}. \end{aligned} \tag{9}$$

Назовем этот этап решения задачи этапом нахождения решений на границе процессоров. Решение системы (9) и составляет нераспараллеливаемую часть данного алгоритма. Размерность этой системы уравнений равна количеству фрагментов, а не количеству точек задачи. Систему будем решать методом прогонки на одном процессоре. Окончательное решение восстанавливаем по формуле (7) и проверяем условие сходимости

$\left\| P_{0i}^{(r)} - P_{0i}^{(r-1)} \right\| \le \varepsilon_1$ для всех подобластей; если оно не выполняется хотя бы в одной подобласти, то снова решаем уравнения (8), а затем (9) и (7).

Следующим шагом находим P_w по формуле $P_w = P_0 - P_{cow}$. Зная P_w, можно найти q_w.

Затем вычисляем $S_{wi}^{(l)}$ по формуле (5) для каждой подобласти и проверяем условие сходимости $S_w^{(l)} - S_w^{(l-1)} \le \varepsilon_2$.

Заключение

Применена технология и система фрагментированного программирования LuNA для реализации алгоритма решения одномерной краевой задачи нефть – вода. Выполнено сравнительное тестирование эффективности различных реализацийалгоритма. На основании результатов тестирования можно сделать вывод, что самым оптимальным по времени выполнения является параллельная реализация. Однако реализация в LuNA обладает рядом преимуществ, таких как относительная простота разработки параллельной программы, так как программист не должен заботиться о коммуникациях между процессами, распределении ресурсов, управлением памятью, что в общем случае является сложным в реализации.

Список литературы

1. Malyshkin V., Perepelkin V. Optimization methods of parallel execution of numericalp rograms in the LuNA fragmented programming system // The Journal of Supercomputing, Springer. 2012. – Vol. 61, N. 1. – P. 235–248. DOI: 10.1007/s11227-011-0649-6.

2. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука. Сиб. отд-е, 1988. – 129 с.

3. Городничев М.А. Оъединение вычислительных кластеров для крупномасштабного численного моделирования в проекте NumGRID // Весник Новосибирского государственного университета (Информационные технологии). – Т. 10, вып. 4. – С. 63–73.

4. Kireev S., Malyshkin V., Fujita H. The LuNA Library of Parallel Numerical Fragmented Subroutines // Lecture Notes in Computer Science. – 2011. – Vol. 6873. – P. 290–301.

5. Kireev S., Malyshkin V. Fragmentation of numerical algorithms for parallel subroutines library // The J. of Supercomputing, Springer. – 2011. – Vol. 57, N. 2. – P. 161–171.

6. Тихонов А.Н., Самарский А.А. Уравнения математической физики. – 5-е изд. – М.: Наука, 1977. – 736 с.

7. Самарский А.А., Гулин А.В. Численные методы: учеб. пособие для вузов. – М.: Наука, 1989. – 432 с.

8. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Наука, 1987. – 630 с.

9. Об организации параллельных вычислений и «распараллеливании» прогонки / Н.Н. Яненко, А.Н. Коновалов, А.Н. Бугров, Г.В. Шустов // Численные методы механики сплошной среды. – Новосибирск, 1978. – № 7. – С. 139–146.

СОДЕРЖАНИЕ

Н.В. Аболмазова, А.Н. Коварцев
К вопросу о возможности оценки надежности сложных программных продуктов 4

Н.Г. Аксак, А.Н. Феденко
Параллелизм в статистических системах машинного перевода 7

С.В. Аксёнов, К.А. Костин, Н.И. Герасимова
Использование фильтров Лавса для классификации текстур на GPU 14

А.С. Антонов, А.М. Теплов
О практической сложности понятия масштабируемости параллельных программ 20

С.С. Артемьев, В.Л. Лукинов
Особенности стохастических вычислений для СДУ и краевых задач на суперкомпьютерах 27

Д.Р. Ахмеджанов, С.А. Лебедев
Об одном подходе к реализации символьной фазы разложения Холецкого для разреженных матриц 34

И.В. Ашарина
Метод построения параллельного алгоритма взаимного информационного согласования в многокластерных вычислительных системах 39

К.А. Баркалов
Об одном классе тестовых задач для параллельных методов глобальной оптимизации 45

Ю.Г. Бартенев, В.А. Ерзунов, А.П. Карпов, Д.А. Петров, И.А. Пищулин, А.Н. Стаканов, Е.Б. Щаникова, И.Е. Капорин, О.Ю. Милюкова, С.А. Харченко, И.Н. Коньшин, И.Б. Мееров, А.В. Сысоев
Комплекс библиотек параллельных решателей СЛАУ LPARSOL версии 3 49

Д.С. Батищев, В.М. Михелев
Инфраструктура облачной системы для обработки медицинских данных 53

А.Т. Бектемесов, Д.Ж. Ахмед-Заки
Распределение и верификация моделей параллельных программ на базе MPI 57

В.А. Биллиг
Сравнительный анализ инструментария параллельных вычислений – OPEN MP и DOT NET 63

А.В. Бовырин, П.Н. Дружков, В.Л. Ерухимов, Н.Ю. Золотых, К.В. Корняков, В.Д. Кустикова, И.Д. Лысенков, И.Б. Мееров, В.Н. Писаревский, А.Н. Половинкин, А.В. Сысоев
Образовательный курс «Разработка мультимедийных приложений с использованием библиотек OPENCV и IPP» 69

И.П. Болодурина, Д.И. Парфёнов
Подходы к решению задачи оптимального использования вычислительных ресурсов в облачных системах..... 73

М.Г. Бояршинов, А.А. Скутин
Использование параллельных вычислительных технологий и ресурсов ЦВВС ПНИПУ для моделирования взаимодействия твердой частицы с потоком жидкости 78

М.Г. Бояршинов, В.А. Трушков, А.В. Казанцев
Использование ресурсов вычислительного кластера ЦВВС ПНИПУ для определения динамических и прочностных характеристик механического оборудования 85

Д.С. Бугаев, В.В. Муромцев
Электронный стетофонендоскоп, поддерживающий диагностику сердечно-сосудистой системы 95

А.Н. Волкович
Использование GPGPU в системах компьютерного зрения реального времени 99

Р.Ф. Гибадуллин
Параллельная система управления базами данных картографических сцен с ассоциативной защитой 105

Т.И. Габдуллин, П.В. Писарев
Численное моделирование деформаций корпусной детали и оснастки при проведении финишной обработки..... 111

Д.Ф. Гайнутдинова, В.Я. Модорский, Г.Ф.Масич
Разработка элементов экспериментально-вычислительного комплекса на базе суперЭВМ для анализа гидроупругих быстропротекающих процессов 117

Е.А. Глазкова
Гибридный параллельный алгоритм построения линий тока 122

Е.Н. Головченко, Е.Ю. Дорофеева
Разбиение графов микродоменов 128

С.Ю. Городецкий
О математических принципах управляемого синтеза классов многоэкстремальных задач с ограничениями для представительного тестирования методов многоэкстремальной оптимизации..... 134

А.В. Горшков, М.Ю. Кириллин, В.П. Гергель
Модифицированный метод Монте-Карло для моделирования распространения зондирующего излучения в задачах оптической диффузионной спектроскопии 141

К.И. Дикарев
Применение параллельной вычислительной среды в задачах поиска оптимальных режимов насосных станций 148

К.А. Дробных, А.Н. Половинкин
Использование алгоритмов глубокого обучения для улучшения точности алгоритмов оценки возраста по фотографии лица человека 154

М.Р. Еникеев, И.М. Губайдуллин
Параллельная обработка изображений анализа питтинговой коррозии 158

Л.В. Еникеева, Н.Ф. Мурзашева, И.М. Губайдуллин
Распределенные вычисления при исследовании кинетики реакции окисления 4-третбутилфенола 165

А.В. Ермаков, И.К. Марчевский, Г.А. Щеглов
Численное моделирование вынужденных колебаний балки в пространственном потоке методом вихревых элементов 169

Е.Б. Замятина, А.А. Митраков
Алгоритмы синхронизации, основанные на знаниях о модели, в распределенных системах имитации 175

А.В. Захарчук, П.А. Смирнов, Д.А. Насонов, С.В. Ковальчук
Динамический предметно-ориентированный язык для задач распределенной обработки больших данных в средах облачных вычислений 182

А.Р. Кабирова, И.М. Губайдуллин
Искусственные нейронные сети при прогнозировании осложнений в бурении 190

Г.Р. Карамутдинова, А.Ф. Шайхнурова, И.М. Губайдуллин
Математическое моделирование и оптимизация транспортных потоков миллионного города Уфы 197

Г.Г. Кашеварова, А. С. Мартиросян
Взаимодействие волны прорыва с сооружениями при гидродинамической аварии 202

А.Н. Коварцев, Д.А. Попова-Коварцева, Е.Е. Серповская
Алгоритм глобального поиска точек разрыва функций многих переменных 208

Е.А. Козинов, Т.В. Лаптева, И.Б. Мееров, М.В. Иванченко, С.В. Денисов
Высокопроизводительный численный Флоке-анализ нестационарных квантовых систем 216

А.В. Коржиманов, М. Марклунд
Моделирование процесса лазерного ускорения ионов
в мишенях с субволновыми неоднородностями поверхности 224

Е.В. Коромыслов, М.В. Усанин, А.А. Синер, Л.Ю. Гомзиков
Расчет задач авиационного двигателестроения методами
высокого порядка на графических процессорах 230

А.Г. Коротченко, В.М. Сморякова
О построении формул численного интегрирования
обыкновенных дифференциальных уравнений с использованием
параллельного алгоритма решения линейных систем 235

М.А. Кривов, М.Н. Притула, П.С. Иванов
Опыт использования технологии автоадаптации программ
для решения системы уравнений Эйлера на GPU-кластере 238

К.С. Кузьмина, И.К. Марчевский
Об использовании вихревых методов для математического
моделирования в сопряженных задачах аэрогидроупругости 244

В.Д. Кустикова, П.Н. Дружков
Применение методов глубинного обучения
для классификации изображений 252

И.Г. Лебедев, К.А. Баркалов
GPU-реализация алгоритма глобального поиска 254

С.А. Лебедев, Е.А. Козинов
Параллельный алгоритм выполнения
численной фазы разложения Холецкого 262

А.В. Лобанов
Интерфейс отказоустойчивости в замкнутых
многокластерных системах 268

А.Г. Масич, Г.Ф. Масич, В.П. Матвеенко, Г.Г. Тирон, И.А. Хохлов
Скоростные сети и распределенные вычисления в УрО РАН 275

Г.Ф. Масич, А.Г. Масич, В.А. Щапов, С.Р. Латыпов,
А.В. Созыкин, Е.Ю. Куклин
Архитектура распределенной вычислительной среды УРО РАН 281

Е.В. Мехоношина, В.Я. Модорский
Газодинамический расчет модельного компрессора 287

Ю.С. Митрохин
Моделирование процесса плавления твердых тел
на суперкомпьютерах фирмы Intel 291

В.Я. Модорский, А.Ф. Шмаков
Параллельный расчет газодинамического процесса
в крупногабаритном вентиляторе 295

А.А. Новиков, Р.Ф. Гибадуллин
Параллельное исполнение пространственных запросов к защищенной картографической базе данных 303

М. Новожилов, А. Белокаменская
Распараллеливание метода марширующих кубов для интерактивной трехмерной цефалометрии 308

С.А. Носова, В.И. Турлапов
Сегментация и 3D-реконструкция митохондрий мозга по снимкам с электронного микроскопа 313

Л.Ф. Нурисламова, И.М. Губайдуллин
Редукция математической модели реакции окисления водорода 318

С.А. Орлов
Суперкомпьютерный центр как конкурентное преимущество национального исследовательского университета 325

Д.А. Павлов, Р.Ш. Минязев
Разработка системы управления базами данных для платформы с графическими ускорителями 330

С.В. Панков
Формальная спецификация однородных вычислительных систем 333

А.В. Пачин, Ба Хла Тхан, М.В. Коваленко, Д.А. Федяшин
Распределенные системы вычислений на примере открытого программного обеспечения BOINC 339

А.Ю. Пирова
Разработка переупорядочивателя симметричных разреженных матриц для систем с общей памятью 344

П.В. Писарев, В.Я. Модорский
Численное моделирование гидроупругих процессов в соединительном канале двухступенчатого центробежного насоса с использованием GPU 351

Р.В. Писарев, П.В. Писарев, В.Я. Модорский
Численное моделирование вибраций лопаток центробежного насоса, нагруженных потоком жидкости 357

А.С. Порозов
Применение многоядерных сопроцессоров INTEL XEON PHI для выполнения агрегатных функций в параллельных СУБД 362

С.В. Пыстогов
Исследование параллельной системы управления полнообъектными базами данных картографических сцен с ассоциативной защитой 366

А.Г. Савельев
Исследование эффективности использования графических процессоров для анализа бинарных сцен 373

М.А. Савин, Ю.А. Селянинов, К.С. Галягин
Построение сеток большой размерности средствами STAR-CCM+ 378

Е.С. Саломатова, Д.Н. Трушников, А.И. Цаплин, В.Я. Беленький
Применение параллельных вычислений для исследования испарения при электронно-лучевой сварке 382

Я.Д. Сергеев, Д.Е. Квасов
Безызбыточная стратегия разбиения в задачах глобальной оптимизации с липшицевыми производными 386

А.А. Сиднев, В.П. Гергель
Применение макромодульного подхода разработки программ 393

А.М. Сипатов, В.Я. Модорский, А.В. Козлова, Л.Н. Бутымова
Численное моделирование газодинамических процессов в модельном вентиляторе с применением высокопроизводительных систем 400

Н.В. Старостин, М.А. Панкратова, Д.В. Седаков
Технология планирования вычислительных ресурсов в параллельной среде 407

Н.В. Старостин, А.В. Филимонов
Декомпозиция графовых структур в топологическом синтезе интегральных схем 414

К.С. Стефанов
Об одном подходе к мониторингу производительности потока задач на суперкомпьютере 418

А.В. Сысоев, К.А. Баркалов, В.П. Гергель
Блочная многошаговая схема параллельного решения задач многомерной глобальной оптимизации 425

М.А. Ташкинов
Применение системы WOLFRAM MATHEMATICA для вычисления многомерных интегралов в задачах стохастической механики 432

Д.И. Федоровцев, П.В. Писарев, А.А. Брюханова
Численное моделирование теплообменных процессов в термоэлектрическом преобразователе энергии с использованием высокопроизводительного вычислительного комплекса ПНИПУ 439

Ю.В. Халевицкий, А.В. Коновалов
Эффективные структуры данных для решения упругопластических задач на гетерогенных кластерных вычислителях 446

Т.А. Ханова, В.Д. Кустикова
Об одном подходе к решению задачи классификации транспортных средств 453

И.В. Храмцов, П.В. Писарев, Р.В. Бульбович
Распределенные вычисления газодинамических характеристик вихревого кольца с использованием метода крупных вихрей 460

В.Е. Хромых, Д.И. Кислицын, А.Н. Супрун, А.И. Домрачев
Разработка веб-интерфейса для автоматизированной системы управления расчетом сложных строительных объектов на кластерных системах 465

Н.И. Червяков, О.В. Аникуева
Система остаточных классов и программируемые логические интегральные схемы для цифровой обработки сигналов 469

Н.И. Червяков, Д.Г. Шалалыгин, И.В. Мелишан
Нейросетевая реализация ГПЧ на точках эллиптической кривой 473

Н.И. Червяков, Д.Г. Шалалыгин, Н.Н. Нагорнов, И.В. Мелишан
Нейросетевая эффективная реализация схемы разделения секрета видеоизображения на основе матрицы Адамара 480

И.С. Чирков, С.Ю. Федосеева, П.В. Писарев
Численное моделирование напряженно-деформированного состояния несущих элементов конструкции стоматологического кресла 485

Л.П. Шингель
Обтекание самолета вертикального взлета при аварийном снижении..... 489

В.А. Щапов, Г.Ф. Масич
Архитектура распределенной системы обработки интенсивного потока структурированных данных на суперкомпьютерах..... 493

И.И. Юсипов, Т.В. Лаптева, М.В. Иванченко
Динамика взаимодействующих частиц в пространственно-неоднородных решетках 499

С.Л. Калюлин, А.Ф. Шмаков, В.Я. Модорский
Оптимизация крупногабаритной конструкции из композиционных материалов в прикладном пакете IOSO PM..... 504

Д.В. Лебедев, В.А. Перепелкин
Реализация одномерной краевой задачи нефть – вода в системе фрагментированного программирования LuNA 511

Научное издание

Высокопроизводительные параллельные вычисления на кластерных системах (HPC 2014)

Материалы
XIV Международной конференции

(г. Пермь, 10–12 ноября 2014 г.)

Редакторы и корректоры: *Е.Б. Денисова, И.Н. Жеганина, И.А. Мангасарова, В.В. Мальцева, Н.А. Панова*

Подписано в печать 5.11.2014. Формат 90×60/16.
Усл. печ. л. 32,87. Тираж 150 экз. Заказ № 194/2014.

Издательство
Пермского национального исследовательского
политехнического университета.
Адрес: 614990, г. Пермь, Комсомольский проспект, 29, к. 113.
Тел. (342) 219-80-33.