Министерство образования и науки Российской Федерации
Суперкомпьютерный консорциум университетов России
Министерство промышленности, инноваций и науки
Пермского края
Министерство образования Пермского края
Государственное образовательное учреждение
высшего профессионального образования
«Пермский государственный технический университет»

# Высокопроизводительные параллельные вычисления на кластерных системах (HPC–2010)

**Материалы X Международной конференции**

(г. Пермь, 1–3 ноября 2010 г.)

В двух томах

**Том 2**

Издательство
Пермского государственного технического университета
2010

УДК 004.382.2:51
ББК 32.973
В93

Сборник X Международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах», состоявшейся в Пермском государственном техническом университете 1–3 ноября 2010 г., содержит материалы докладов, посвященных высокопроизводительным параллельным вычислениям на кластерных системах применительно к науке и образованию.



**Финансовая поддержка конференции**

Российский фонд фундаментальных исследований
Министерство промышленности, инноваций и науки Пермского края
Корпорация «AMD»
Корпорация «Intel»
Компания «NVIDIA»
Компания «Microsoft»
Компания «Т-Платформы»
Компания «Делкам-Урал»
Компания «Тесис»
Корпорация «IBM»
Компания «РСК СКИФ»

**Информационная поддержка конференции**

Газета научного сообщества «Поиск»
Информационно-аналитический центр Parallel.Ru
Информационно-аналитический журнал «Rational Enterprise Management»
Молодежный парламент при Законодательном собрании Пермского края

ISBN 978-5-398-00506-6

**Г.Г. Кашеварова, И.Н. Фаизов, А.Ю. Зобачева**

Пермский государственный технический университет

# ИССЛЕДОВАНИЕ ПРОЦЕССА ДЕФОРМИРОВАНИЯ, РАЗРУШЕНИЯ И ВОЗМОЖНОСТИ УСИЛЕНИЯ ЗДАНИЯ НА ПОДРАБОТАННОЙ ТЕРРИТОРИИ С УЧЕТОМ ПРОГНОЗА РАЗВИТИЯ ДЕФОРМАЦИЙ ЗЕМНОЙ ПОВЕРХНОСТИ[1]

Начиная с осени 2007 года по февраль 2008-го происходило затопление камер пластов восточных панелей рудника БКПРУ-1 (г. Березники Пермского края). По стенным реперам, заложенным в жилых зданиях, проводились регулярные наблюдения оседания земной поверхности. Результаты ежемесячных наблюдений, начиная с апреля 2008 года, свидетельствуют о продолжающейся тенденции нарастания скоростей оседаний земной поверхности, что представляет опасность для расположенных там жилых зданий города.

Прогноз деформаций земной поверхности выполнен на конец 2009-го – начало 2010 года. Сравнение значений достигнутых деформаций кривизны и горизонтальных деформаций с прогнозными значениями показало, что они увеличатся в среднем на 15–25 %. Потенциальную опасность такого увеличения необходимо определять для каждого объекта отдельно, исходя из его состояния на настоящий момент, технических характеристик, положения в краевой части мульды сдвижения. Необходимо выполнить оценку каждого здания на необходимость ввода конструктивных мер защиты или возможность их дальнейшей эксплуатации и организовать мониторинг состояния этих зданий.

В данной работе на примере конкретного здания исследуется процесс его деформирования, появления трещин и рассматриваются варианты конструктивных мер защиты.

[1] Работа выполняется при финансовой поддержке гранта РФФИ №08-08-00702-а.

Экспертная оценка технического состояния строительных конструкций здания шахтно-бытового корпуса ШБК-3 на БКПРУ-1 выполнена специалистами отдела обследования строительных конструкций ОАО «Галургия».

Конструктивно здание выполнено в неполном каркасе. Несущими конструкциями являются наружные кирпичные стены толщиной 640 мм, внутренние кирпичные стены толщиной 380 мм и сборные железобетонные колонны и ригели (в продольном направлении), частично защемленные на опорах.

Фундаменты под стены ленточные из крупных блоков, под колонны – железобетонные стаканного типа из бетона марки 150. По подвальным стенам предусмотрен железобетонный пояс со стаканами для установки колонн.

Междуэтажные и чердачное перекрытия запроектированы из сборных железобетонных многопустотных плит с обычным армированием по серии ИИ-03-02 или плит с предварительно напряженной арматурой. Марка бетона М200.

Общая устойчивость здания обеспечивается внутренними лестничными клетками из сборных железобетонных элементов, поперечными и продольными наружными и внутренними стенами, дисками перекрытий.

Внутренние перегородки выполнены кирпичными и сборными железобетонными.

В ходе обследования здания выявлен ряд дефектов и повреждений строительных конструкций. Наиболее поврежденными являются наружные и внутренние несущие стены из силикатного кирпича и покрытие здания.

В наружных стенах большое количество наклонных и вертикальных трещин. Трещины преимущественно расположены в подоконных частях и проходят по вертикальным и горизонтальным швам кирпичной кладки. Ширина раскрытия трещин достигает 10 мм. У отдельных оконных железобетонных перемычек обнаружено разрушение бетона, оголение и коррозия арматуры. На момент обследования цокольная часть здания отремонтирована, однако в некоторых местах вновь образовались

трещины. Отдельные участки стен проморожены и разрушены. Во внутренних несущих стенах центральных лестничных клеток выявлены трещины шириной раскрытия до 10 мм на всю высоту здания. Внутри здания выявлены дефекты и повреждения, обусловленные длительной эксплуатацией здания без проведения своевременных ремонтов.

Данное здание было смоделировано в конечно-элементном программном комплексе ANSYS, выполнены расчеты на действие эксплуатационных нагрузок, замеренных и прогнозируемых осадок. Расчет выполнялся с использованием пространственных конечных элементов SOLID65, позволяющих учесть возможность трещинообразования и раскрашивания материалов конструкций. Характер разрушения конструкций здания, полученный в результате обследования и расчета достаточно близок, что позволило исследовать возможность применения некоторых конструктивных мер защиты здания (введение поясов усиления, возможность разрезки здания на отсеки).

**А.С. Кириллов**

Оренбургский государственный университет

## ПОДХОДЫ К РАЗРАБОТКЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РЕШЕНИЯ КООПЕРАТИВНЫХ ИГР ДЛЯ КЛАСТЕРНЫХ СИСТЕМ[1]

В последние годы значение теории игр существенно возросло во многих областях экономических и социальных наук. В экономике она применяется не только для решения общехозяйственных задач, но и для анализа стратегических проблем предприятий, разработок организационных структур и систем стимулирования.

[1] Работа выполнена при финансовой поддержке Министерства образования и науки Российской Федерации в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг.

Аппарат теории игр активно используется при создании систем поддержки принятия решений. При этом очень актуальна проблема разработки эффективных алгоритмов для поиска оптимальных стратегий в игровых моделях. На практике требуется, чтобы алгоритмы были способны обрабатывать большое количество информации об участниках рынка в режиме реального времени. Направление теории игр, связанное с разработкой и анализом эффективных алгоритмов их решения, получило название «Алгоритмическая теория игр» и очень активно развивается именно в последние 15 лет. Коллективная монография [1], подводящая первые итоги становления алгоритмической теории игр, была издана совсем недавно – в 2007 году.

Среди направлений теории игр особое место занимают кооперативные игры, они используются, например, в разработке оптимальных стратегий безопасности систем, эффективной ценовой политики и др. В кооперативных играх игроки имеют возможность принимать совместные решения и, в некоторых случаях, перераспределять выигрыши между собой. В кооперативной игре задаются возможности и предпочтения различных групп игроков (коалиций), для которых строятся оптимальные стратегии и задаются распределения между ними суммарных выигрышей. Результатом решения кооперативной игры являются оптимальная коалиция, в которую следует вступить игрокам, их оптимальные смешанные стратегии и выигрыш коалиции. Распределение выигрыша между игроками определяется дележом.

Особого внимания заслуживают задачи моделирования конкурентных рынков. Примером подобной задачи может быть моделирование конкурентного рынка отрасли информационных технологий, где игроками являются производители компьютерной техники и программного обеспечения, имеющие возможность объединяться в коалиции, конкурирующие за соответствующие ниши рынка.

Существуют различные способы решения кооперативных игр, например: решение на основе функции полезности Неймана-Моргенштерна, решение игры в виде С-ядра и в стратегиях

угроз, однако эти подходы не определяют оптимальных стратегий игроков внутри коалиций. А решение игры в виде С-ядра дает целое множество допустимых решений, не определяя оптимального решения.

Попытки построения количественного подхода к решению кооперативной игры описаны в работах Протасова [2] и Фергюсона [3]. Предложен подход, основанный на представлении кооперативной игры набором биматричных игр для каждой пары игроков. Недостатком этого подхода является его высокая вычислительная сложность, поскольку количество биматричных игр экспоненциально растет при увеличении количества игроков.

Таким образом, разработанные в настоящее время алгоритмы решения кооперативных игр имеют серьезные практические ограничения по количеству игроков и не могут использоваться в современных системах поддержки принятия решений. В нашей работе мы предлагаем подходы к решению кооперативных игр с использованием параллельных вычислений для различных типов параллельных архитектур.

**Анализ подходов к распараллеливанию решения кооперативной игры.** Один из актуальных современных практических подходов к решению задач высокой вычислительной сложности связан с использованием параллельных вычислений. Разработка эффективных высокомасштабируемых параллельных алгоритмов и широкое распространение многоядерных архитектур, вычислительных кластеров, грид-систем позволяют аккумулировать при решении задачи большие вычислительные ресурсы и резко сократить время поиска решения.

Разработка параллельных алгоритмов для решения игр – новое направление в алгоритмической теории игр. На сегодняшний день имеются лишь единичные публикации в этой области, например работы Уиджера и Гросу для некооперативных игр [4, 5]. Для решения кооперативных игр параллельные алгоритмы пока не разработаны. Отметим, что разнообразие свойств параллельных архитектур требует исследования и адаптации параллельных алгоритмов под различные типы архитектур. Так, алгоритмы для систем с общей памятью могут неэффективно

работать в распределенных системах, для оптимизации скорости алгоритмов требуется учитывать характеристики коммуникационной системы и т.д.

Для решения кооперативных игр можно применить сведение задачи к множеству стратегических игр и решение каждой из них симплекс-методом – для нахождения выигрыша каждой коалиции (более подробно описано в работе [6]). Всего коалиций может быть $2^n$, где $n$ – количество игроков, следовательно, необходимо решить $2^n$ стратегических игр симплекс-методом Дж. Данцига, который обладает экспоненциальной сложностью [7]. При достаточно больших $n$ это занимает длительное время.

Кроме того, большую вычислительную сложность имеет процесс формирования матриц, которыми задаются коалиции. Трудоемкость этого процесса составляет $O(n2)$. Поскольку матрицы, описывающие коалиции, не зависят друг от друга, а зависят только от матриц, задающих саму игру, генерация матриц может осуществляться параллельно.

Исходя из вышесказанного, можно сделать вывод, что формирование матриц, характеризующих коалиции, и их расчет симплекс-методом являются независимыми участками кода, которые можно выполнять параллельно.

Нами разработан алгоритм решения кооперативных игр для кластерных систем. Суть разработанного параллельного алгоритма состоит в следующем.

Нулевым процессом выполняется считывание матриц, задающих игру, после чего они рассылаются остальным незанятым работающим процессам, которые, в свою очередь, формируют матрицу, характеризующую коалицию. Номера игроков, входящих в коалицию, определяются по соответствию позиций единичных элементов двоичного представления порядкового номера формируемой матрицы. После чего для сформированной коалиции строится соответствующая задача линейного программирования и решается симплекс-методом. Результат отсылается нулевому процессу, который проверяет игру на сущест-

венность, и если игра существенна, то вычисляется дележ по Шепли. Затем все результаты выводятся в выходной файл.

Схема работы параллельной программы представлена на рис. 1.

Алгоритм реализован на C с использованием библиотеки MPI (Message Passing Interface).

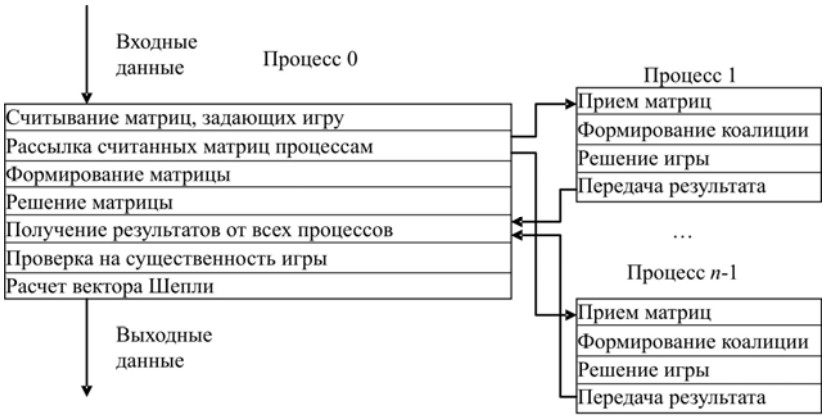



Рис. 1. Схема работы параллельного алгоритма

В приведенном выше алгоритме доля последовательного участка подчиняется следующему закону: $f = \frac{k}{k^{n-1} + k}$, из которого видно, что при возрастании $n$ или $k$ доля последовательных вычислений стремится к нулю, где $n$ – количество игроков, $k$ – количество стратегий у игроков.

Тестирование программы производилось на кластере Оренбургского государственного университета, построенного на базе 2-процессорных Xeon 3Ггц под управлением операционной системы SuSe Linux.

Тестирование производилось для 7 игроков, и у каждого игрока имелось по 4 стратегии поведения.

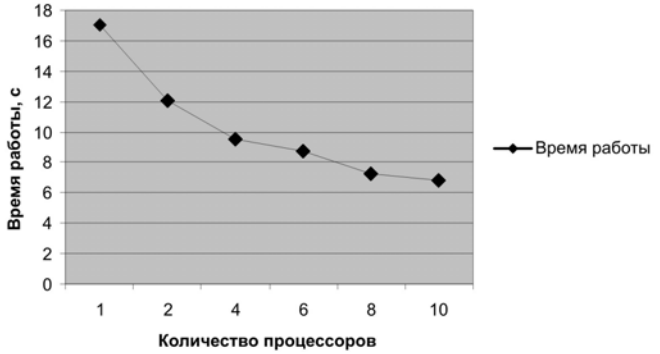



Рис. 2. График зависимости времени работы
от количества процессоров

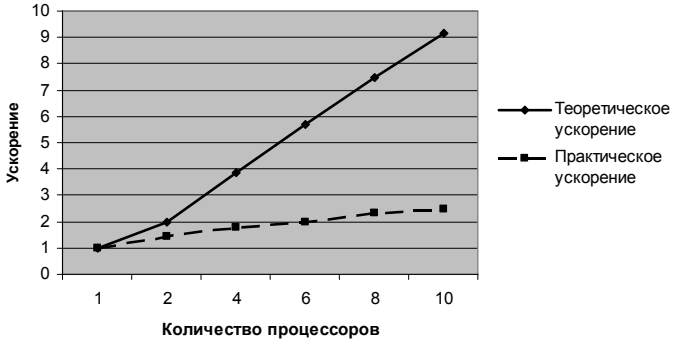



Рис. 3. График зависимости теоретического и практического
ускорения от количества процессоров

Зависимость времени работы программы от количества процессоров представлена на рис. 2.

Для наглядного представления построим график зависимости теоретической оценки ускорения по закону Амдала и ускорения работы программы, рассчитанной после тестирования от количества процессоров, который представлен на рис. 3.

Итак, в статье была рассмотрена идея сведения кооперативных игр к множеству биматричных для игроков с большим числом стратегий и предложен способ задания и решения коо-

ративных игр. После анализа алгоритма была предложена схема решения поставленной задачи с использованием высокопроизводительных вычислений, в частности, был разработан и программно реализован параллельный алгоритм. После запуска программы на кластере ОГУ был проведен анализ полученных результатов.

Как видно из графика, приведенного на рис. 3, наблюдается тенденция ускорения работы программы с увеличением количества работающих процессоров, хотя максимальное теоретическое ускорение и не была достигнуто.

Несоответствие теоретического и практического ускорения может быть связано с тем, что теряется время на пересылку данных между процессами и/или с тем, что нулевой процесс простаивает, ожидая завершения работы остальных процессов для продолжения решения задачи.

В дальнейшем, планируется рассмотреть другие способы решения кооперативных игр и модификации алгоритмов для других типов параллельных архитектур.

## Список литературы

1. Algorithmic Game Theory / ed. by Noam Nisan, Cambridge University Press, 2007. – 754 p.

2. Протасов И.Д. Теория игр и исследование операций. – М.: Гелиос АРВ, 2006. – 368 с.

3. Thomas S. Ferguson GAME THEORY Class notes for Math 167, Fall, 2000.

4. Widger J. and Grosu D. Computing Equilibria in Bimatrix Games by Parallel Support Enumeration. In Proceedings of the 2008 international Symposium on Parallel and Distributed Computing (July 01–05, 2008). ISPDC. IEEE Computer Society. – Washington, DC, 2008. – P. 250–256.

5. Widger J. and Grosu D. Parallel Computation of Nash Equilibria in N-Player Games, Computational Science and Engineering, IEEE International Conference, 2009. – P. 209–215.

6. Колемаев В.А., Калинина В.Н. Теория вероятностей и математическая статистика: учеб. для вузов. – 2-е изд., перераб. и доп. – М.: Юнити, 2003. – 352 с.

7. Данциг Дж. Линейное программирование, его применения и обобщения. – М.: Прогресс, 1966. – 600 с.

**Е.А. Козинов**

Нижегородский государственный университет им. Н.И. Лобачевского

## РАЗЛИЧНЫЕ ПОДХОДЫ К ВИЗУАЛЬНОМУ ПРЕДСТАВЛЕНИЮ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

В настоящей работе рассматривается одна из актуальных проблем написания параллельных программ – снижение сложности и времени разработки. В настоящее время большая часть процессоров в различных вычислительных устройствах имеют более одного ядра. Во всех крупных компаниях используются многопроцессорные вычислительные системы. Как следствие, эффективная программа должна уметь утилизировать данные ресурсы, т.е. быть параллельной. На создание параллельной программы, как правило, уходит существенно больше времени, чем на разработку последовательной программы. Связано это с тем, что в параллельных программах могут возникать «параллельные ошибки», сложность выявления и устранения которых гораздо выше «обычных ошибок». Для снижения сложности написания параллельных программ может быть применено визуальное программирование. Визуальный язык может помочь программисту построить правильный «каркас» параллельной программы. Сгенерированный «каркас» в данном случае берет на себя всю сложность организации параллельных вычислений, оставив на программиста последовательные участки кода, отвечающие за программируемый алгоритм. Такая схема

создания параллельных программ, как правило, по производительности будет проигрывать коду, написанному полностью вручную, но позволяет сократить время написания и отладки программ. Следовательно, работы в данном направлении являются актуальными.

**Краткий обзор визуальных языков.** Направление создания визуальных языков параллельного программирования активно и давно развивается. Часть языков обладает общими чертами, с помощью которых языки можно условно разделить на группы. Наиболее интересными представляются следующие группы языков:

- группа, отображающая программы в виде функциональных зависимостей;
- группа, отображающая программы в виде асинхронно выполняемых последовательностей задач;
- группа, отображающая программы через представление их потоков данных.

Визуальные языки, относящиеся к первой группе, содержат два типа элементов: функции и их аргументы. Результат выполнения функции – аргумент, являющийся либо целью алгоритма, либо входом для другой функции. При графическом изображении функциональных языков чаще всего используют древовидные структуры. Корнем дерева является финальная цель алгоритма. Листьями дерева являются входные данные алгоритма и функции. Все остальные узлы соответствуют алгоритму решения задачи. Ярким примером функционального языка является язык из системы «Пифагор» [1]. На рис. 1 представлен параллельный алгоритм суммирования элементов вектора.

Визуальные языки, отображающие асинхронно выполняемые задачи, представляют параллельную программу в виде направленного графа. В качестве вершин графа могут быть использованы один или несколько типов запускаемых задач. Ребра в графе отображают последовательность, в которой задачи должны быть выполнены. В данных языках, как правило, принято, что для задачи запускается столько копий, сколько на нее указывает ребер. Яркими представителями данной группы визуальных языков являются «CODE» [2] и «ГРАФ ПЛЮС» [3].

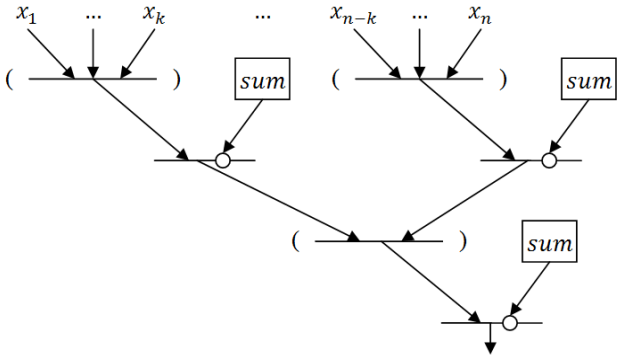



Рис. 1. Визуальная схема параллельного суммирования вектора с точки зрения функционального языка

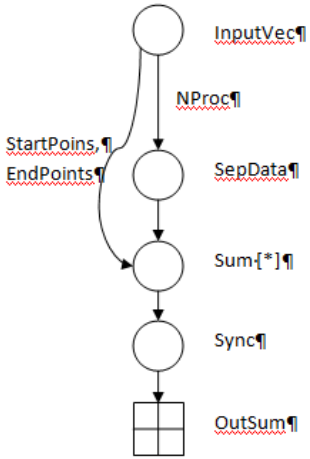



Рис. 2. Визуальная схема параллельного суммирования вектора в виде асинхронно выполняемых последовательностей задач

В зависимости от визуального языка граф может быть дополнен дополнительными графическими элементами, как это сделано в языке «HeNCE» [4]. Отличительной его особенностью является то, что в языке есть специальные графические элементы для представления графа параллельной программы в более компактном виде. К этим графическим элементам относятся обозначение условного исполнения задач, а также репликации цик-

лического и конвейерного исполнения задач в графе. Параллельное суммирование вектора элементов на графическом языке «CODE» [2] может выглядеть, как изображено на рис. 2.

Визуальные языки, отображающие потоки данных, также представляют параллельную программу в виде направленного графа, но смысл вершин и ребер меняется. В данных визуальных языках вершины графа – либо задачи, которые нужно исполнить, либо процессы, которые умеют выполнять заданный алгоритм. Сами задачи могут обмениваться данными. Для отображения передачи данных в языке используются ребра. Как правило, в данных языках считается, что задачи запускаются либо в момент старта программы, либо по готовности всех входящих данных для задачи. К данной группе языков относятся языки «TREPPER»[5] и «GRAPNEL» [6]. В обоих языках вершины графа описывают процессы. Ребра отображают каналы передачи данных. Для каждого процесса есть возможность отдельно описать выполняемую работу. Данные в этих языках могут передаваться через одни и те же каналы, причем передачу данных требуется указывать явным образом. Другие два языка «PGRAPH» [7] и «ЯГСПП» [8] явным образом не привязаны к понятиям процессов или потоков. Под задачей понимается вычислительный элемент, использующий входные данные для создания выходных. Данные при этом передаются автоматически по завершении задачи. Параллельное суммирование вектора элементов на графическом языке, подобном «PGRAPH» [7] и «ЯГСПП» [8], может выглядеть, как изображено на рис. 3.

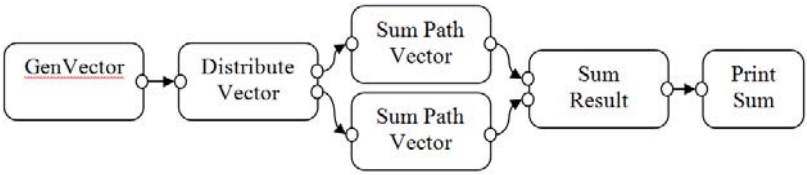



Рис. 3. Визуальная схема параллельного суммирования, отображающая потоки данных

**Разрабатываемый визуальный язык.** Разрабатываемая визуальная система параллельного программирования относится к последней группе и оперирует тремя основными типами визуальных элементов:

1. **«Задача»** – данный компонент содержит часть логики разрабатываемого алгоритма.

2. **«Порт»** и **«каналы»** – визуальные элементы, отображающие данные, передаваемые между задачами.

3. **«Компонент разделения»** и **«компонент слияния»** данных – необходимые визуальные элементы в параллельном программировании для отображения геометрической декомпозиции данных.

Алгоритм суммирования элементов вектора на разрабатываемом языке представлен на рис. 4.

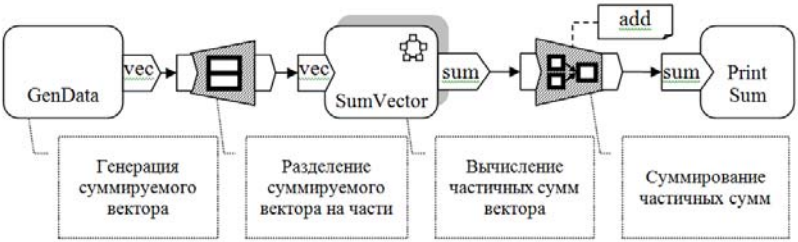



Рис. 4. Визуальная схема параллельного суммирования вектора в виде асинхронно выполняемых последовательностей задач

Одна из основополагающих идей разрабатываемого языка – за счет специальных визуальных элементов добиться компактности, прозрачности передачи данных и масштабируемости получаемого представления параллельных программ.

Рассмотрим более подробно пример визуальной схемы суммирования элементов вектора представленного на рис. 4. Первая задача «GenData» последовательная и существует для того, чтобы сгенерировать суммируемый вектор. Далее идет специальный визуальный элемент – компонент разделения данных. Методов разделения данных в визуальном языке преду-

смотрено несколько видов. Каждый из видов имеет свое графическое представление. Вид разделения данных визуально отображен на компоненте разделения данных. После разделения данных идет задача специального вида «SumVector». Можно обратить внимание на то, что задача «SumVector» отличается от остальных тем, что у нее есть тень. Тень означает, что задача имеет несколько копий. Предполагается, что количество копий может быть задано в момент исполнения, при этом компонент разделения данных сам понимает, как необходимо распределить данные между копиями. Важно заметить, что код задачи «SumVector» остается последовательным, но пишется для части полученных данных. Данный подход дает схемам масштабируемость и компактность представления. После подсчета частичных сумм с помощью компонента слияния данных вычисляется сумма всех элементов вектора. Как и в случае компонента разделения данных, метод слияния визуально отображен на компоненте слияния данных. В финале с помощью последовательной задачи «PrintSum» результат суммирования может быть отображен пользователю. Также следует заметить, что во всех визуальных элементах видно, какие и куда передаются данные, поскольку все данные предаются через порты.

В данной работе на простом примере рассмотрены различные подходы к визуальному представлению параллельных программ. Также коротко описан новый разрабатываемый визуальный язык, позволяющий описывать сложные параллельные алгоритмы в компактном виде. Компактность графического представления обусловлена наличием ряда введенных компонентов, таких как задачи с несколькими копиями, компоненты разделения и слияния данных. Задачи с несколькими копиями позволяют на схемах не рисовать повторяющиеся задачи и при этом получать более масштабируемые параллельные программы. Масштабируемость достигается за счет того, что часть введенных компонентов параметризировано. За счет изменения параметров можно изменять степень параллельности получаемых программ.

Целью дальнейших исследований является поиск расширений визуального языка, которые позволили бы:

- контролировать данные, с которыми работает пользователь системы;
- создавать и повторно использовать базу визуальных схем параллельных программ;
- используя базу визуальных схем, автоматически выстраивать схемы алгоритмов решения задач, поставленных пользователем.

## Список литературы


1. Функциональная модель параллельных вычислений и язык программирования «Пифагор» / А.И. Легалов [и др.] – URL: http://www.softcraft.ru/parallel/fpp/fppcontent.shtml (дата обращения: 27.06.2010)

2. Experiences with CODE and HeNCE in Visual Programming for Parallel Computing / J.C. Browne [et al.] // IEEE Parallel and Distributed Technology. B. – 1994. – Vol. 3, No. 1. – P. 75–83.

3. Востокин С.В. Графический метод проектирования параллельных программ с использованием асинхронной событийной модели вычислений // Вестн. Самар. гос. техн. ун-та. Серия «Физико-математические науки». – 2004. – № 30. – С. 178–183.

4. Graphical Development Tools for Network-Based Concurremt Supercomputing / A. Beguelin [et al.] // Proc. Supercomputing ‘91, Albuquerque, NM. B, 1994. P.435-444.

5. Scheidler C., Schäfers L. TRAPPER:A Graphical Programming Environment for Industrial High-Performance Applications // PARLE '93 Parallel Architectures and Languages Europe. Lecture Notes in Computer Science B. 1993 Vol.694. – P. 403–413.

6. A graphical development and debugging environment for parallel programs / P. Kacsuk [et al.] // Parallel Computing, February, 1997. – Vol. 22. – P. 1747–1770

7. Жидченко В.В. Программный комплекс моделирования и анализа алгоритмов параллельных вычислений: дис. … к.т.н. – Самара, 2007. – 189 с.

8. Программные средства поддержки выполнения граф-схемных программ для кластерных систем / Д.В. Котляров [и др.] // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород, 2006. – С. 156–159.

**Е.А. Козинов, М.В. Кутлаев, Д.В. Осокин**

Нижегородский государственный университет им. Н.И. Лобачевского

# СОЗДАНИЕ УЧЕБНОЙ БИБЛИОТЕКИ ПАРАЛЛЕЛЬНЫХ МЕТОДОВ PARLIB

Для решения ряда задач необходимы огромные вычислительные мощности. К таким задачам относятся задачи построения прогноза погоды, расшифровки генома человека, создания лекарств и многие другие. Необходимые вычислительные ресурсы могут предоставить многоядерные и многопроцессорные вычислительные системы. Для того чтобы извлечь их потенциал, необходимо обладать специальными знаниями и уметь использовать различные технологии параллельного программирования. Программисты должны уметь разрабатывать параллельные алгоритмы, а затем переносить их в программный код.

Чтобы овладеть знаниями в области параллельного программирования, необходимо разрабатывать учебные курсы и практикумы. В работах [1–4] на часто применяемых в вычислениях алгоритмах показаны базовые подходы, используемые при разработке параллельных программ. Для улучшения восприятия материала необходимы практические занятия с заранее разработанным правильно работающим программным кодом. При наличии такого кода обучаемые имеют возможность сравнить свои разработанные параллельные программы с эталоном, оценить их правильность и увидеть возможный потенциал улучшений. Таким образом, создание библиотеки, использующей различные технологии и реализующей алгоритмы из работ [1–4], является важным и актуальным направлением работы.

**Постановка задачи.** Для создания учебной библиотеки параллельных методов **ParLib** в первой версии необходимо использовать две технологии OpenMP и MPI. Данная библиотека предназначена для поддержки проведения практических занятий. Основная цель библиотеки – не только достижение максимального ускорения реализуемых алгоритмов, но и демонстрация пользователям возможных провалов производительности

различных параллельных схем. Для обучения предполагается весь код библиотеки предоставлять в открытом виде.

В первой версии библиотеки необходимо создать по нескольку реализаций параллельных алгоритмов для следующих разделов математики:

- перемножение матрицы на вектор;
- перемножение матриц;
- решение систем линейных однородных уравнений;
- алгоритмы оптимизации на графах;
- сортировки данных;
- решение дифференциальных уравнений в частных производных.

**Описание архитектуры библиотеки.** Архитектура библиотеки разрабатывается с учетом ее учебного применения. Реализуемые методы могут поддаваться модификации, причем это не должно затрагивать структуры всего проекта (рисунок).

Проект ParLib состоит из нескольких типов библиотек:

1. Библиотеки алгоритмов параллельных методов. Для каждого раздела математики разрабатывается две группы параллельных методов – с использованием технологий OpenMP и MPI. Группа алгоритмов оформляется в виде динамически подключаемой библиотеки.

2. Обобщающие библиотеки. С помощью них пользователь получает доступ к методам из библиотек алгоритмов. Обобщающие библиотеки содержат функции, предоставляющие возможность выбора необходимого алгоритма из группы за счет параметра – идентификатора метода.


Рис. Схема архитектуры библиотеки

Механизм динамического связывания библиотек параллельных методов позволяет вносить изменения только в необходимую библиотеку, не изменяя всего проекта.

На текущий момент создана первая версия библиотеки ParLib. Библиотека реализует параллельные алгоритмы из различных областей математики. В реализации используются широко распространенные технологии параллельного программирования OpenMP и MPI. Созданная библиотека обладает гибкой архитектурой. Для каждого раздела математики и каждой технологии код оформлен в виде динамической библиотеки. Если пользователь ParLib в учебных целях захочет попробовать изменить часть реализованного кода, то для интеграции с библиотекой достаточно заменить только одну динамическую библиотеку, а не компилировать весь код ParLib заново. Кроме того, для каждой динамической библиотеки, реализующей алгоритмы из выбранного раздела математики, существует набор тестов. Благодаря тестам пользователь может контролировать правильность своих изменений. Для тех же, кто планирует использовать библиотеку ParLib для сравнения различных способов распараллеливания, предусмотрена обобщающая динамическая библиотека. Данная библиотека позволяет через интерфейсную функцию посредством одного лишь параметра выбирать метод распараллеливания, решать поставленную задачу и засекать время ее работы.

Целью дальнейшей разработки библиотеки ParLib является создание графического интерфейса, позволяющего запускать и сравнивать производительность существующих в библиотеке базовых параллельных реализаций алгоритмов. Также предполагается, что в дальнейшем визуальный интерфейс будет позволять пользователям из базовых алгоритмов конструировать более сложные. Для более сложных алгоритмов пользователь сможет исследовать влияние выбора методов распараллеливания базовых алгоритмов на решение общей сложной задачи.

## Список литературы

1. Гергель В.П. Теория и практика параллельных вычислений. – М.: БИНОМ. Лаборатория знаний. 2007. – 424 с.

2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

3. Grama, A., Gupta, A., Kumar V. Introduction to Parallel Computing. – Harlow, England: Addison-Wesley, 2003, 2nd edn. – 656 p.

4. Quinn M.J Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill, 2004. – 480 p.

**А.В. Козлова, В.П. Муленков, Ю.В. Соколкин, Д.В. Зимин, В.Я. Модорский**

Пермский государственный технический университет

## ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ ДЛЯ МОДЕЛИРОВАНИЯ ПРОЦЕССОВ ГИДРОАБРАЗИВНОГО ИЗНОСА В ЦИРКУЛЯЦИОННОМ КАРМАНЕ ФЛОТАЦИОННОЙ МАШИНЫ

**Постановка задачи.** Технологическое оборудование для переработки калийных солей работает в контакте с пульпой и постоянно подвержено воздействию химической агрессивной среды и гидромеханической смеси, которая, в свою очередь, вызывает абразивный износ участков поверхности [1, 2, 3]. При длительной эксплуатации такое оборудование изнашивается и требует замены. Для решения таких задач необходимо произвести обоснованный выбор материалов элементов оборудования, рассмотреть возможности прогнозирования скорости его износа, а также создания оптимальных конструкторско-технологиических решений.

В качестве важного элемента технологической цепочки для переработки калийных солей применяется флотационный циркуляционный карман (рис.1). Он устанавливается в ванну и полностью погружается в раствор с пульпой. Импеллер всасывает пульпу через насадок (поток идет от щелевой части к цилиндрическому фланцу насадка) и выбрасывает раствор обратно в ванну. Вместе с потоком пульпы импеллер всасывает поток воздуха для обеспечения процесса флотации.

Пульпа состоит из водного раствора и твердых частиц KCl и NaCl. Карман имеет входную щель, кромки которой подвержены гидроабразивному износу. Карман выполнен из листовой стали толщиной 8 мм. Однако под воздействием потока через 6–8 месяцев эксплуатации кромки щели значительно изнашиваются, и конструкцию приходится менять.

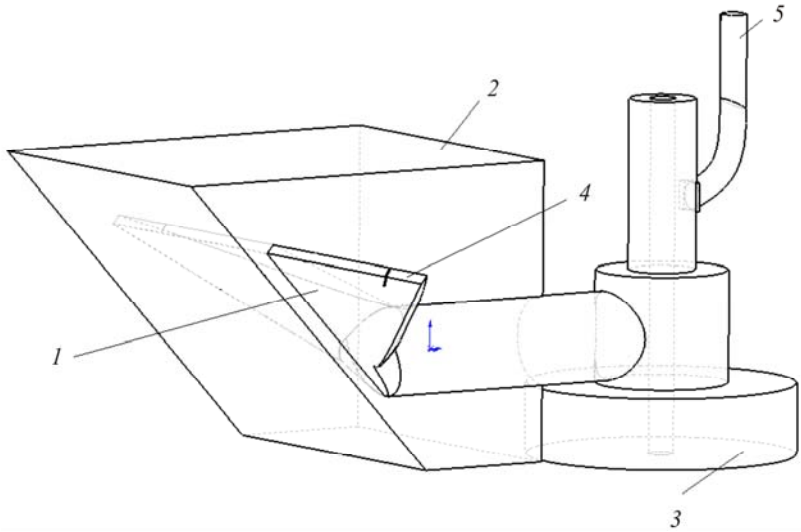



Рис. 1. Схема флотационной машины
*1* – карман; *2* – ванна; *3* – импеллер; *4* – входная щель кармана; *5* – подача воздуха

Для исследования процессов гидроабразивного износа в циркуляционном кармане флотационной машины необходимо расчетным путем определить параметры потока пульпы и характеристики нагрузки, действующей в насадке флотационной машины. В качестве перспективного материала насадка применен стеклопластик. Вычислительные эксперименты проведены с использованием современных систем инженерного анализа FlowVision-HPC и ABAQUS, технических ресурсов Регионального центра технической компетенции «AMD-ПГТУ». Программа расчета газогидродинамических процессов подготовлена к использованию на высокопроизводительном вычислительном

комплексе ПГТУ, имеющем следующие технические характеристики: 64 вычислительных узла; 128 четырехядерных процессоров Barcelona-3 (всего 512 ядер); пиковая производительность 4,096 Тфлоп; производительность в тестовом пакете Linpack 78 %; объем системы хранения информации 12 ТБ; объем оперативной памяти 512 Гб (8 Гб/узел).

Для гидродинамического расчета процессов в стеклопластиковом насадке флотационной машины применялся многопроцессорный программный комплекс FlowVision-HPC [4]. FlowVision обладает характеристиками, которые являются важными при выборе системы инженерного анализа: реализация технологии параллельных вычислений, высокая точность выполняемых расчетов и приемлемые требования к вычислительным ресурсам; широкий выбор математических моделей; автоматическая генерация адаптивной сетки для численных расчетов; удобный русскоязычный графический интерфейс и приемлемый по времени и достаточный по возможностям анализ результатов; адаптация к требованиям заказчика и поддержка российского разработчика, а также низкие, сравнительно с конкурентными системами инженерного анализа, инвестиции в организацию рабочего места.

Сформулирована следующая физическая модель:

– конструкция полагается трехмерной;

– рабочие тела: несущая фаза 1 (пульпа), несущая фаза 2 (воздух) и несомая фаза (частицы соли);

– камера постепенно заполняется частицами соли.

Сформирована следующая математическая модель. Данная модель описывает течения вязкой жидкости/газа при малых числах Маха (M<0,3). Допускаются малые изменения плотности, что позволяет естественным образом учесть подъёмную силу.

Математическая модель включает в себя следующие соотношения:

1. Уравнение сохранения массы несущей фазы 1

$$\frac{\partial}{\partial t}\left(\rho_1\right)+\bar{\nabla}\left(\rho_1 V_1\right)=Q_{\text{м}}^{\text{ч}}.$$

2. Уравнение сохранения импульса несущей фазы 1

$$\frac{\partial}{\partial t}\left(\rho_1 \boldsymbol{V}_1\right)+\bar{\nabla}\left(\rho_1 \boldsymbol{V}_1 \otimes \boldsymbol{V}_1\right)=-\bar{\nabla} P+\bar{\nabla}\cdot\tau_1+\rho_1 \boldsymbol{g}+Q_{\text{и}}^{\text{ч}}.$$

3. Уравнение сохранения массы несущей фазы 2

$$\frac{\partial}{\partial t}\left(\rho_1 Y_2\right)+\bar{\nabla}\left(\rho_1 \boldsymbol{V}_1 Y_2\right)=\bar{\nabla}\left(\left(\rho_1 D+\frac{\mu_{\text{т}}}{\text{Sc}_{\text{т}}}\right)\bar{\nabla} Y_2\right)+Q_{\text{м}}^{\text{ч}}.$$

4. Уравнение состояния

$$P=\frac{\rho_1 R_0 T_1}{M}.$$

5. Уравнение динамики движения частиц

$$\frac{d\boldsymbol{V}_{\text{ч}}}{dt}=\frac{\pi d_{\text{ч}}^2}{8 m_{\text{ч}}} C_D \rho_1 \left|\boldsymbol{V}_1-\boldsymbol{V}_{\text{ч}}\right|\left(\boldsymbol{V}_1-\boldsymbol{V}_{\text{ч}}\right)+\boldsymbol{g}\left(1-\frac{\rho_1}{\rho_{\text{ч}}}\right).$$

6. Уравнение сохранения массы частицы

$$\frac{\partial m_{\text{ч}}}{\partial t}=-m^{*}\pi d_{\text{ч}}^2.$$

7. Уравнение турбулентной энергии

$$\frac{\partial}{\partial t}\left(\rho_1 K\right)+\bar{\nabla}\left(\rho_1 \mathbf{V}_1 K\right)=\bar{\nabla}\left(\left(\mu+\frac{\mu_{\text{т}}}{\sigma_k}\right)\bar{\nabla} K\right)+\mu_{\text{т}} G-\rho_1 \varepsilon.$$

8. Уравнение скорости диссипации турбулентной энергии

$$\frac{\partial}{\partial t}\left(\rho_1 \varepsilon\right)+\bar{\nabla}\left(\rho_1 \mathbf{V}_1 \varepsilon\right)=\bar{\nabla}\left(\left(\mu+\frac{\mu_{\text{т}}}{\sigma_\varepsilon}\right)\bar{\nabla}\varepsilon\right)+C_1\frac{\varepsilon}{K}\mu_{\text{т}} G-C_2 f_1 \rho_1 \frac{\varepsilon^2}{K}.$$

Задавались следующие граничные условия. На входе, в вертикальном сечении ванны вдали от щелевой части насадка, задавалась скорость подачи пульпы и частиц. Выход моделировался в выходном сечении импеллера. На входе подачи воздуха задавалась скорость его подачи. Для несущей фазы на стенке задано условие «логарифмический закон» изменения скорости, а для несомой фазы (частиц) условие упругого отскока. Для несущей фазы на границе «симметрия» задано условие проскальзывания, а для несомой фазы (частиц) условие упругого отскока.

В качестве метода решения исходной системы дифференциальных уравнений в частных производных применяется численный подход – метод конечных объемов. В этом методе дифференциальные уравнения переноса интегрируются по объему каждой *i*-й ячейки расчетной сетки и по временному шагу *dt*.

Метод конечных объемов является вариантом метода конечных разностей и сохраняет его преимущества – простые и быстрые алгоритмы расчета. При этом метод конечных объемов лишен недостатков, присущих методу конечных разностей, так как разбивка производится не на «кирпичики», а на многогранники с произвольными углами и размерами. Это позволяет достаточно точно описать реальную геометрию.

На первом этапе вычислительный эксперимент производится в рамках модели, не учитывающей наличие твердых частиц в потоке. На втором этапе – с учетом движения частиц. По результатам проведенных вычислительных экспериментов можно провести оценку гидродинамических процессов, возникающих при работе флотационной машины, определить поля скоростей и давлений потока в области входной щели кармана, степень наполненности установки пульпой и частицами соли, распределение частиц соли в объеме камеры в зависимости от их размера, определить силовые нагрузки на конструкцию кармана.

Результаты, полученные в ходе гидродинамического расчета применимы для дальнейшего расчета напряженно-деформированного состояния и оценки на прочность стеклопластикового насадка флотационной машины. Для такого расчета применялся известный в области конечно-элементных прочностных расчетов программный комплекс ABAQUS [5]. С его помощью можно получать точные и достоверные решения сложных линейных и нелинейных инженерных задач.

Результаты проведенных вычислительных экспериментов по оценке напряженно-деформированного состояния позволили сделать прогноз гидроабразивного износа защитного покрытия циркуляционного кармана. Проведены расчеты для различных конструкторских оформлений насадка. На основании получен-

ных результатов, в качестве конструкторского решения, предложены наружные ребра жесткости, расположенные вдоль щели. Это позволило увеличить проходное сечение насадка в 6 раз по сравнению с базовым вариантом, так как зазор увеличивается с 3 до 20 мм.

### Список литературы

1. Пермяков Р.С. Добыча и переработка калийных солей Верхнекамского месторождения. – Пермь: Пермское книжное изд-во, 1976. – 144с.

2. Абрамов А.А. Флотационные методы обогащения. – М.: Недра, 1993. – 45 с.

3. Исследование процессов гидроабразивного износа в циркуляционном кармане флотационной машины: отчет о НИР / Перм. гос. техн. ун-т; рук. Ю.В. Соколкин; исполн.: В.Я. Модорский [и др.]. – Пермь, 2007. – 140 с.

4. FlowVision 3.07.02. Руководство пользователя [Электронный ресурс] / ООО «ТЕСИС». – М., 2010. – URL: http://www.flowvision.ru/ (дата обращения 21.05.2010).

5. ABAQUS. Руководство пользователя [Электронный ресурс] / ООО «ТЕСИС». – М., 2010. – URL: http://www.tesis.com.ru/software/abaqus/applian.php (дата обращения 31.05.2010).

**А.В. Козлова, В.Я. Модорский**

Пермский государственный технический университет

## ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ ДЛЯ МОДЕЛИРОВАНИЯ НЕСТАЦИОНАРНЫХ ПРОЦЕССОВ В ГАЗОХОДАХ

Наиболее достоверные результаты могут быть получены в результате проведения натурных экспериментов. Однако высокая стоимость их проведения накладывает жесткие ограничения на объем опытных испытаний. Кроме того, чрезвычайно затруднено и физическое моделирование. При этом необходимо

соблюдение как геометрического, так и физического подобия модельной и натурной конструкции.

Строгое аналитическое решение оказывается очень приблизительным. В свою очередь, вычислительный эксперимент, базирующийся на численных подходах, оперирует более полными и точными описаниями натурного изделия.

Вычислительный эксперимент в отличие от других методов исследования характеризуется такими свойствами, как оперативность, экономность, наглядность, многомодельность, универсальность и безопасность. Таким образом, представляется обоснованным выбор вычислительного эксперимента для изучения нестационарных газогидродинамических процессов в качестве эффективного современного метода исследований сложных нелинейных задач.

Отработка вычислительного эксперимента реализована с использованием технических ресурсов Регионального центра технической компетенции «AMD-ПГТУ». Программа расчета газогидродинамических процессов подготовлена к использованию на высокопроизводительном вычислительном комплексе ПГТУ, имеющем следующие технические характеристики: 64 вычислительных узла; 128 четырехядерных процессоров Barcelona-3 (всего 512 ядер); пиковая производительность 4,096 Тфлоп; производительность в тестовом пакете Linpack 78 %; объем системы хранения информации 12 ТБ; объем оперативной памяти 512 Гбайт (8 Гбайт/узел).

**Постановка задачи.** В технике широко используются газоходы для отвода и охлаждения высокотемпературных и высокоскоростных потоков истекающих газов от энергетических установок. На рисунке представлен газоход переменного сечения. Горячие газы поступают в газоход и продвигаются по каналу. Для их охлаждения, снижения скорости и удержания вредных компонентов через пояса форсунок малого диаметра в газоход подается вода. Порции воды, подаваемые у истока горячей струи, отбирают от нее энергию и способствуют турбулизации потока, что вовлекает дополнительно воздух. В средней части

газожидкостного эжектора (ГЖЭ) происходит дальнейшее охлаждение и торможение потока. Вода, взаимодействуя с горячим газом, сначала нагревается до температуры кипения, а затем превращается в пар. На выходе образуется парогазовая смесь.

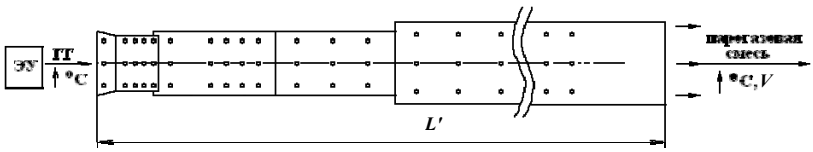



Рис. Газоход переменного сечения

При проведении работ имели место случаи нестационарной непрогнозируемой работы газохода, которые сопровождались возникновением нештатных ситуаций. Поэтому необходимо исследовать газодинамические процессы при работе ГЖЭ и выявить области его допустимой работы. При этом для удобства при использовании необходима программа для ЭВМ, которая представляла бы собой базу данных и являлась универсальным инструментом для анализа результатов.

Полная оценка газогидродинамических параметров потока в ГЖЭ и построение многопараметрической области его допустимой работы возможны при проведении серии вычислительных экспериментов, рассматриваются даже экстремальные режимы. Таким образом, необходимо разработать методику проведения таких вычислительных экспериментов для оценки нестационарных газогидродинамических параметров в канале газохода.

Сформулирована следующая физическая модель: процессы рассматриваются в трехмерной динамической постановке; поток многофазный: несущая фаза – вязкий сжимаемый газ, несомая фаза – несжимаемая жидкость; учитываются процессы движения, нагревания и испарения капель воды; учитывается взаимное влияние фаз. Для расчета нестационарных газогидродинамических процессов в проточном тракте газохода выбрана

стандартная k-ε модель турбулентности, ввиду ее устойчивости и разумной точности вычислений. Кроме того, данная модель турбулентности является высокорейнольдсовой и сверхдиффузионной.

При формировании физической модели были приняты следующие допущения:

1. Несомая фаза представлена монодисперсными каплями воды (частицы). Это объясняется тем, что нет точных данных по геометрическим характеристикам капель воды, поступающих в газоход.

2. Стенки канала непроницаемые, нетеплопроводные (продолжительность работы установки в вычислительном эксперименте намного меньше, чем время прогрева стенки) и гладкие (принимается, что все выступы элементов шероховатости лежат внутри вязкого слоя).

3. Не учитываются процессы дробления и коагуляции капель, так как принимается, что скорости этих (противоположных) процессов равны.

4. Не учитывается конденсация пара на поверхности капель ввиду того, что газогидродинамические процессы рассчитываются в пределах газохода, а процессы конденсации в основном протекают за пределами газохода.

5. Не учитывается осаждение капель воды на стенку. Предполагается упругий отскок.

6. Не учитывается процесс окисления горячих газов в эжектируемом воздухе. Результаты вычислительных экспериментов хорошо согласуются с результатами натурных экспериментов, поэтому нет необходимости учитывать химические реакции при взаимодействии горячих газов и воздуха.

7. Расчеты проводились без учета гравитации, так как предварительные вычислительные эксперименты показали, что результаты расчетов с учетом и без учета гравитации близки.

В соответствии с принятой физической моделью разработана математическая модель, которая базируется на законах со-

хранения массы, импульса, энергии и замыкается уравнениями состояния, динамики частиц и уравнениями турбулентности.

Математическое описание процесса газожидкостной эжекции в указанной постановке включает в себя следующие соотношения:

Уравнение сохранения массы газа

$$\frac{\partial}{\partial t}\left(\rho_{\text{г}}\right)+\bar{\nabla}\left(\rho_{\text{г}}\boldsymbol{V}_{\text{г}}\right)=Q_{\text{м}}^{\text{ч}}.$$

Уравнение сохранения импульса газа

$$\frac{\partial}{\partial t}\left(\rho_{\text{г}}\boldsymbol{V}_{\text{г}}\right)+\bar{\nabla}\left(\rho_{\text{г}}\boldsymbol{V}_{\text{г}}\otimes\boldsymbol{V}_{\text{г}}\right)=-\bar{\nabla}P+\bar{\nabla}\tau_{\text{г}}+\rho_{\text{г}}\boldsymbol{g}+Q_{\text{и}}^{\text{ч}}.$$

Уравнение сохранения энергии газа

$$\frac{\partial}{\partial t}\left(\rho_{\text{г}}H_{\text{г}}\right)+\bar{\nabla}\left(\rho_{\text{г}}\boldsymbol{V}_{\text{г}}H_{\text{г}}\right)=\frac{\partial P}{\partial t}+\bar{\nabla}\left(\left(\frac{\lambda_{\text{г}}}{c_{p\text{г}}}+\mu_{\text{т}}\right)\bar{\nabla}H_{\text{г}}\right)+Q_{\text{э}}^{\text{ч}}.$$

Уравнение сохранения массы пара

$$\frac{\partial}{\partial t}\left(\rho_{\text{г}}Y_{\text{п}}\right)+\bar{\nabla}\left(\rho_{\text{г}}\boldsymbol{V}_{\text{г}}Y_{\text{п}}\right)=\bar{\nabla}\left(\left(\rho_{\text{г}}D+\frac{\mu_{\text{т}}}{\text{S}c_{\text{т}}}\right)\bar{\nabla}Y_{\text{п}}\right)+Q_{\text{м}}^{\text{ч}}.$$

Уравнение состояния

$$P=\frac{\rho_{\text{г}}R_0T_{\text{г}}}{M}.$$

Уравнение динамики движения частиц

$$\frac{d\boldsymbol{V}_{\text{ч}}}{dt}=\frac{\pi d_{\text{ч}}^2}{8m_{\text{ч}}}C_D\rho_{\text{г}}\left|\boldsymbol{V}_{\text{г}}-\boldsymbol{V}_{\text{ч}}\right|\left(\boldsymbol{V}_{\text{г}}-\boldsymbol{V}_{\text{ч}}\right)+\boldsymbol{g}\left(1-\frac{\rho_{\text{г}}}{\rho_{\text{ч}}}\right).$$

Уравнение сохранения массы частицы

$$\frac{\partial m_{\text{ч}}}{\partial t}=-m^*\pi d_{\text{ч}}^2.$$

Уравнение сохранение энергии частицы

$$\frac{\partial T_{\text{ч}}}{\partial t}=\left(\left[\text{Nu}\frac{\lambda_{\text{г}}}{d_{\text{ч}}}\left(T_{\text{г}}-T_{\text{ч}}\right)\right]-m^*q\left(T_{\text{ч}}\right)\right)\frac{6}{d_{\text{ч}}\rho_{\text{ч}}c_{p_{\text{ч}}}}.$$

$$\frac{\partial}{\partial t}\left(\rho_{г} K\right)+\bar{\nabla}\left(\rho_{г}\mathbf{V}_{г} K\right)=\bar{\nabla}\left(\left(\mu+\frac{\mu_{т}}{\sigma_{k}}\right)\bar{\nabla} K\right)+\mu_{т} G-\rho_{г}\varepsilon\,.$$

Уравнение скорости диссипации турбулентной энергии

$$\frac{\partial}{\partial t}\left(\rho_{г}\varepsilon\right)+\bar{\nabla}\left(\rho_{г}\mathbf{V}_{г}\varepsilon\right)=\bar{\nabla}\left(\left(\mu+\frac{\mu_{т}}{\sigma_{\varepsilon}}\right)\bar{\nabla}\varepsilon\right)+C_{1}\frac{\varepsilon}{K}\mu_{т} G-C_{2} f_{1}\rho_{г}\frac{\varepsilon^{2}}{K}\,.$$

Задавались следующие граничные условия. Горячий газ поступал в канал газохода при заданных постоянных значениях температуры и массовой скорости подачи. Массовая скорость подачи горячего газа варьировалась от 78 до 178 кг/с·м$^2$. Воздух поступал в газоход из окружающей среды при нормальных условиях. Вода через форсунки подавалась в установку с заданным расходом. На выходе образовывался дозвуковой поток, задавалось условие «свободного истечения» (значение давления на границе приравнивается начальному, а нормальные производные параметров скорости и температуры на границе равны нулю). На границах «симметрия» и «стенка» задавались условия «проскальзывания» для несущей фазы и условие «непротекания» для несомой фазы. Для сокращения времени счета задачи рассматривается сектор, представляющий собой ¼ часть объема модельной конструкции, образованный взаимноперпендикулярными плоскостями, проходящими через ось симметрии установки.

В качестве метода решения исходной системы дифференциальных уравнений в частных производных рассматривается численный метод конечных объемов [1, 2, 3]. В этом методе дифференциальные уравнения переноса интегрируются по объему каждой *i*-й ячейки расчетной сетки и по временному шагу *dt*.

Метод конечных объемов является вариантом метода конечных разностей и сохраняет его преимущества – простые и быстрые алгоритмы расчета. При этом метод конечных объемов лишен недостатков присущих методу конечных разностей, так как разбивка производится не на «кирпичики», а на многогранники с произвольными углами и размерами. Это позволяет достаточно точно описать реальную геометрию.

Для реализации вычислительных экспериментов по расчету гидрогазодинамических нестационарных рабочих процессов в канале переменного сечения газожидкостного эжектора был выбран программный комплекс FlowVision-HPC [4].

FlowVision-HPC обладает характеристиками, которые являются важными при выборе системы инженерного анализа: реализация технологии параллельных вычислений, высокая точность выполняемых расчетов и приемлемые требования к вычислительным ресурсам; широкий выбор математических моделей; автоматическая генерация адаптивной сетки для численных расчетов; удобный русскоязычный графический интерфейс и приемлемый по времени и достаточный по возможностям анализ результатов; адаптация под требования заказчика и поддержка российского разработчика, а также низкие, сравнительно с конкурентными системами инженерного анализа, инвестиции в организацию рабочего места.

**Методика проведения вычислительных экспериментов.** В соответствии с методологией реализации вычислительного эксперимента [5] сформулирована инженерная методика его проведения, применительно к расчету нестационарных газогидродинамических процессов в газоходах. Выделим основные этапы.

1-й этап – постановка задачи. Она включает в себя формулирование физической модели, качественное описание объекта моделирования и построение его геометрии.

2-й этап – разработка математической модели. Математическая модель включает в себя 10 уравнений и дополняется соответствующими граничными и начальными условиями.

На 3-м этапе осуществляется выбор метода решения. Выбран метод конечных объемов.

4-й этап характеризуется выбором программного обеспечения. Для проведения вычислительных экспериментов выбрана система инженерного анализа FlowVision.

5-й этап предусматривает отладку программного обеспечения. Полученное в результате вычислительного эксперимента численное решение является приближенным. Для его количест-

венной оценки погрешности проводится решение тестовых задач: расчет газового эжектора, решение задачи о движении поршня в трубе, тепловой расчет газового и газожидкостного эжектора. Далее проводится сравнение результатов аналитических решений и численных экспериментов.

На 6-м этапе проводят вычислительные эксперименты. Выбирается диапазон варьирования массовой скорости и температуры горячего газа. Диапазон варьирования коэффициента соотношения расхода воды и горячего газа. Осуществляется выбор критериев, в соответствии с которыми будет осуществляться анализ полученных результатов.

7-й этап – обработка результатов. При проведении вычислительных экспериментов получается огромный числовой материал. Для его обработки необходимо осуществить визуализацию результатов в виде:

а) построения полей. Это позволит обнаружить области повышенных значений температуры и давления и оценить скорости потока;

б) построения графиков по длине газохода. Это позволит провести анализ волновых процессов, оценить температурное воздействие на стенки;

в) построение графиков по времени на входе и выходе из газохода. По полученным данным можно судить о динамике характеристик потока и выходе решения на стационарный режим.

По полученным результатам строятся области и номограммы допустимой работы газохода в соответствии с выбранными критериями.

6, 7-й этапы повторяются столько раз, сколько необходимо провести вычислительных экспериментов.

На 8-м этапе проводится анализ результатов вычислительных экспериментов. Он позволит судить об адекватности математической модели объекту исследования, надежности программной реализации.

**Разработка программы для ЭВМ.** По результатам анализа серии вычислительных экспериментов разработана про-

грамма для ЭВМ «Поток 2» (Свидетельство об официальной регистрации программ для ЭВМ №2010614281 от 01.07.2010 г.), которая зарегистрирована Роспатентом. При задании исходных данных программа выводит на монитор графические зависимости или распределение по выбранному сечению давления, температуры или скорости потока. Предусмотрена возможность анализа динамических процессов с помощью анимации.

## Список литературы

1. Роуч П. Вычислительная гидродинамика. – М.: Мир, 1980. – 616 с.

2. Патанкар С.В. Численные методы решения задач теплообмена и динамики жидкости. – М. : Энергоатомиздат, 1984. – 152 с.

3. Crowe C., Sommerfeld M. and Tsuji Yu. Multiphase Flows with Droplets and Particles // CRC Press LLC. – 1998. – 471 p.

4. FlowVision 3.07.02. Руководство пользователя [Электронный ресурс] / ООО «ТЕСИС». – М., 2010. – URL: http://www.flowvision.ru/ (дата обращения 21.05.2010).

5. Самарский, А.А. Вычислительный эксперимент в задачах технологии // Вестн. АН СССР. – 1984. – № 11. – С. 17–29.

**А.Г. Колчанова**

ОАО «ПермьЭнерго» филиал «МРСК Урала»
Пермские городские электрические сети

## РЕШЕНИЕ ЗАДАЧ УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ С ИСПОЛЬЗОВАНИЕМ КЛАСТЕРНЫХ ТЕХНОЛОГИЙ И ERP-СИСТЕМЫ «КАПИТАЛ CSE»

Сегодня практически во все отрасли экономики, науки и техники стали внедряться кластерные системы. Создаются кластеры и в такой важной отрасли промышленности, как энергетика (с целью взаимовыгодного объединения предприятий данной отрасли в рамках одного региона).

С 1999 года была автоматизирована деятельность в аппарате управления ОАО «Пермэнерго», а также во всех его филиалах с помощью единой системы управления предприятием (ERP-системы) «Капитал CSE». Система «Капитал CSE» включает в себя более 50 бизнес-приложений, сгруппированных в 16 прикладных контуров.

Кроме того, в систему могут входить дополнительные компоненты:

- редактор таблиц OleReport;
- растройка основных элементов приложения;
- специальные диалоговые окна;
- настройка параметров соединения;
- панель запуска приложений.

Программный комплекс «Капитал CSE» (разработчик ЗАО «Геликон Про») – это система управления предприятием, предназначенная для информационного обеспечения процессов управления материальными, финансовыми и кадровыми ресурсами.

Одним из важнейших свойств подобных систем является способность адаптироваться к бизнес-процессам конкретного предприятия. С этой точки зрения, одним из наиболее важных контуров комплекса является контур «Средства настройки Капитал CSE», предназначенный для выполнения задач, связанных с адаптацией базовой функциональности продукта.

Функциональные элементы данного контура:

- объектно ориентированный язык GOAL, поддерживающий язык SQL, управление транзакциями и выполнение хранимых процедур, структурную обработку исключительных ситуаций, работу с OLE-серверами и т.д.;
- редактор диалогов, предназначенный для визуального проектирования и программирования интерфейса пользователя;
- проблемно ориентированная библиотека объектов GBO(GELICON BUSINESS OBJECT).

Внедрение и функционирование системы «Капитал CSE» на предприятии призвано обеспечить:

- переход на новую модель управления в рамках единого информационного пространства;

- эффективное решение оперативно-управленческих и бухгалтерско-учетных задач;
- приведение системы отчетности в соответствие с российскими и международными стандартами;
- непрерывный учет и контроль ресурсов предприятия и взаимоотношений с контрагентами;
- оперативную подготовку аналитических документов, отчетов и справок, а также прогнозов и планов работ предприятия;
- повышение эффективности и скорости принятия управленческих решений;
- организацию сквозного учета и своевременного контроля ключевых участков финансово-хозяйственной деятельности в режиме реального времени;
- качественное повышение эффективности всего процесса договорной деятельности за счет систематизации учета договоров, дополнительных соглашений и актов;
- разграничение прав доступа и назначения ответственных за конкретные обязательства;
- ведение архива документов;
- экономический эффект за счет контроля соблюдения сроков исполнения обязательств;
- контроль соблюдения исполнения бюджетов;
- сокращение финансовых издержек в виде штрафов;
- увеличение оперативности деятельности за счет создания единой базы договоров;
- предоставление актуальной информации о состоянии договоров;
- электронные средства поиска и анализа информации.

На первом этапе на каждом из предприятий ОАО «Пермэнерго» была развернута своя база данных и инсталляция системы.

В течение 2008 года был осуществлен переход всех филиалов и аппарата управления на новую версию системы «Капитал CSE» 4.1. При этом вся информация из баз данных предприятий(филиалов) была объединена в одной базе данных, эксплуатируемой на сервере ОАО «Пермэнерго». Возникла необходимость использования системных средств для распределения

ресурсов между всеми пользователями, которых к тому времени насчитывалось около 1600. Кроме того, требовались дополнительные средства администрирования для работы с единой базой, объединяющей информацию по всем вышеуказанным задачам и по всем предприятиям ОАО «Пермэнерго».

Переход был успешно выполнен благодаря тому, что были приняты корректные решения задач доступа к информации единой базы данных, к ресурсам системы и распределению прав доступа отдельных пользователей и групп пользователей к ресурсам.

С 2009 года работа удаленных пользователей (Березниковские электрические сети, Очерские электрические сети, Северные электрические сети, Кунгурские электрические сети, Чайковские электрические сети, Чусовские электрические сети, Центральные электрические сети, Пермские городские электрические сети ) осуществляется в режиме реального времени посредством терминального доступа.

В качестве терминальной системы как раз и был использован кластер компьютеров под управлением MS Windows Server 2003. Встала задача распределения вычислительных ресурсов, памяти и др. В качестве сервера баз данных использовалась СУБД Oracle на платформе Linux.

Именно на такой платформе появилась возможность решить задачу централизации информации и распределения ресурсов между достаточно большим количеством пользователей.

В связи с тем что, по определению, кластер объединяет два или более серверов, действующих совместно для обеспечения безотказной работы набора приложений или служб и воспринимаемых клиентом как единый элемент, эта технология как нельзя более кстати была использована для решения задачи удаленного доступа к единой базе данных ОАО «Пермэнерго». Узлы кластера объединяются между собой с помощью аппаратных сетевых средств, совместно используемых разделяемых ресурсов и серверного программного обеспечения.

Microsoft Windows 2000/2003 поддерживает две технологии кластеризации: кластеры с балансировкой нагрузки (Network Load Balancing) и кластеры серверов.

Кластер серверов распределяет свою нагрузку среди серверов кластера, причем каждый сервер несет свою собственную нагрузку. Если происходит отказ узла в кластере, то приложения и службы, настроенные на работу в кластере, прозрачным образом перезапускаются на любом из свободных узлов. Кластеры серверов используют разделяемые диски для обмена данными внутри кластера и для обеспечения прозрачного доступа к приложениям и службам кластера. Для них требуется специальное оборудование, но данная технология обеспечивает очень высокий уровень надежности, поскольку сам кластер не имеет какой-либо единственной точки отказа. Этот режим конфигурации кластера также называется active-passive режимом. Приложение в кластере работает на одном узле с общими данными, расположенными на внешнем хранилище.

Поскольку задачи управления персоналом, бухгалтерского учета должны выполняться в режиме реального времени, любой сбой в работе системы может повлечь за собой серьезные последствия, поэтому отказоустойчивость кластера помогает оперативно решать возникшие во время работы системы проблемы. Именно эта технология и используется в ОАО «Пермэнерго».

Кластерный подход к организации внутренней сети дает следующие преимущества:

1. Высокий уровень готовности. Если происходит сбой службы или приложения на каком-то узле кластера, настроенного на совместную работу, кластерное программное обеспечение позволяет перезапустить это приложение на другом узле. Пользователи при этом ощутят кратковременную задержку при проведении какой-то операции либо вообще не заметят серверного сбоя.

2. Масштабируемость. Для приложений, работающих в кластере, добавление серверов к кластеру означает увеличение возможностей: отказоустойчивости, распределение нагрузки и т.д.

3. Управляемость. Администраторы, используя единый интерфейс, могут управлять приложениями и службами, устанавливать реакцию на сбой в узле кластера, распределять нагрузку среди его узлов и снимать нагрузку с них для проведения профилактических работ.

Рассмотрим работу кластера на реальном примере – системе управления предприятием (в данном случае речь пойдет об ОАО «Пермэнерго»).

Сфера деятельности. В настоящее время ОАО «Пермэнерго» работает в статусе филиала ОАО «Межрегиональная распределительная сетевая компания Урала», осуществляет передачу электрической энергии по распределительным сетям и подключение потребителей к электросетевой инфраструктуре. При автоматизации деятельности системы управления предприятием при помощи системы «Капитал CSE» задействованы следующие бизнес-процессы, автоматизированные с использованием данной системы:

1) бухгалтерский учет;

2) учет движения основных средств;

3)учет движения товарно-материальных ценностей;

4) расчет заработной платы;

5) управление персоналом;

6) управление договорной деятельностью;

7) управление снабжением;

8)управление финансами.

Администрирование терминального сервера дает возможность распределения прав доступа пользователей не только к информации базы данных, но и к ресурсам на самом терминале. Лишь часть пользователей имеет возможность пересылки почты со своего компьютера на терминал, это ограничение было вызвано необходимостью защиты терминала от компьютерных вирусов, лишь часть пользователей имеет возможность пересылки сформированных таблиц на свой компьютер, этот доступ дается лишь при возникновении производственной необходимости.

Использование экономических, технических кластеров – новый и наиболее корректный подход для решения современных задач, требующих интеграции информационных процессов, обработки больших объемов информации в режиме реального времени при наличии технических ресурсов предприятий и вузов[1].

[1] К информации, изложенной выше, есть доступ в сети Интернет на сайте компании «Геликон Про» http://www.gelicon.biz.

**С. Кукаева**

Нижегородский государственный университет им. Н.И. Лобачевского

# ТЕСТИРОВАНИЕ РЕАЛИЗАЦИИ МОАГП С ИСПОЛЬЗОВАНИЕМ РАЗВЕРТОК ПЕАНО НА GPU

**Постановка задачи и используемые подходы.** Рассматривается конечномерная задача оптимизации (задача нелинейного программирования)

$$f(y) \to inf, y \in Y \subseteq R^N$$

$$D = \left\{ y \in R^N : y_i \in [a_i, b_i], 1 \le i \le N \right\},$$

т.е. задача отыскания экстремальных значений целевой (минимизируемой) функции $f(y)$ в области $Y$, задаваемой координатными ограничениями [1, 4].

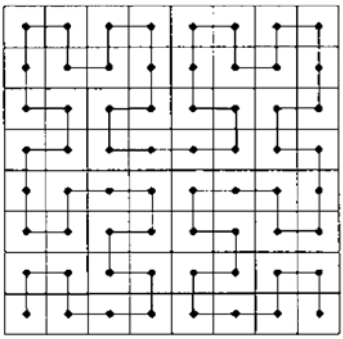


Рис. 1. Образ отрезка [0, 1] при отображении пеаноподобной разверткой третьего порядка

Не уменьшая общности, далее будем считать, что поставлена задача нахождения минимума. Одним из подходов к решению многомерной задачи оптимизации является сведение многомерной задачи к одномерной. Это может сделано разными способами. Наиболее эффективными являются многошаговая схема оптимизации и редукция размерности на основе кривых Пеано, о которой далее и пойдет речь. Данная схема основана на известном фундаментальном факте, согласно которому $N$-мерный гиперпараллелепипед и отрезок [0, 1] вещественной оси являются равномощными множествами, и отрезок [0, 1] может быть однозначно и непрерывно отображен на гиперпараллелепипед. Такие отображения и называются кривыми (или развертками) Пеано [1, 2].

На рис. 1 изображен образ отрезка [0, 1] при отображении пеаноподобной разверткой при разбиении третьего порядка (отметим, что в данной работе используется 8-й порядок разбиения).

В качестве основного алгоритма решения была выбрана модификация информационно-статистического алгоритма глобального поиска (АГП) – многомерный обобщенный алгоритм глобального поиска (МОАГП), в котором в качестве характеристики интервала используется величина

$$R(i) = m^N \sqrt{|x_i - x_{i-1}|} + \frac{(z_i - z_{i-1})^2}{m^N \sqrt{|x_i - x_{i-1}|}} - 2(z_i + z_{i-1}),$$

где

$$m = f(x) = \begin{cases} rM, M > 0 \\ 1, M = 0 \end{cases} (r-1),$$

$$M = \max_{1 \le i \le \tau} \frac{(z_i - z_{i-1})}{\sqrt[N]{|x_i - x_{i-1}|}},$$

$$z_j = f\left(y(x_j)\right), 1 \le j \le \tau,$$

$y(x_j)$ – точка $N$-мерного гиперпараллелепипеда, являющаяся образом точки $x_j$ отрезка [0, 1].

В данной работе используется одна развертка на всю область поиска минимума.

Искомый глобальный минимум будем искать согласно следующему алгоритму:

1. Запускаем $K$ потоков на GPU, пронумерованных 0, 1, …, $K$–1 соответственно.

2. Отрезок [0, 1] также разбиваем на $K$ отрезков, пронумерованных 0, 1, …, $K$–1.

3. Потоку $i$ соответствует отрезок с тем же номером. Каждый поток ищет минимум на своем отрезке. Важно, что потоки не обмениваются информацией относительно найденных точек, таким образом, каждый поток находит свой минимум на своем участке. Данные о найденных минимумах передаются на CPU.

4. Находим минимальное из тех значений, что найдены в каждом потоке. Эта часть является последовательно исполняемой на CPU.

В данной версии реализация поиска минимума отдельным потоком абсолютно идентична реализации поиска минимума на CPU.

**Результаты вычислительных экспериментов.** В наших численных экспериментах было использовано два класса (простой и сложный) 2-мерных тестовых ND-функций, порожденных GKLS-генератором [3]. Этот генератор обладает несколькими преимуществами, которые позволяют использовать его в качестве хорошего инструмента для сравнения численных алгоритмов.

Он генерирует классы 100 тестовых с одинаковым числом локальных минимумов и предоставляет полную информацию о каждой функции: ее размерность, значения всех локальных минимумов, их координаты, области притяжения и т.д. Существует возможность с легкостью генерировать более сложные или более простые тестовые классы. Только 5 параметров должны быть определены пользователем, а остальные генерируются случайным образом. Важная особенность генератора состоит в полной повторяемости экспериментов: если вы используете те же самые 5 параметров, то при каждом запуске генератор будет порождать такой же класс функций [3].

Важно, что вычисления значений функций производится не на CPU, а на GPU. Параметры, необходимые для вычисления значения функции, заносятся в специальную структуру, которая передается в глобальную память GPU. Используя эти данные, на GPU каждый поток вычисляет значение в необходимой точке.

Таблица 1

**Параметры тестовых классов функций**

| Класс $n$ | $N$ | $m$ | $f^*$ | $d$ | $r_g$ |
|---|---|---|---|---|---|
| Простой | 2 | 10 | –1,0 | 0,66 | 0,33 |
| Сложный | 2 | 10 | –1,0 | 0,90 | 0,20 |

В табл. 1 представлены параметры для каждого из используемых классов функций, где $f^*$ – значение функции в глобальном минимуме, $d$ – максимальное расстояние от точки глобального минимума до центра параболоида, $r_g$ – радиус области притяжения глобального минимума функции. Глобальный оптимум считается найденным, если точка найденного оптимума попала в некоторую окрестность ρ настоящего минимума, которая в нашем случае равна 0,02.

Параметр надежности алгоритма для «простого» и «сложного» классов функций $r$ выбирался равным 2,1. Это значение было выбрано экспериментально, поскольку при нем удается получить решение всех задач тестового набора при относительно небольшом числе испытаний. Вторым важным параметром параллельной реализации алгоритма являлось число K начальных отрезков, на которые разбивается исходный отрезок. Следует напомнить, что в каждой подобласти независимо от других запускается параллельный поток, в котором выполняется алгоритм поиска глобального минимума. В каждом потоке разрешается выполнить по $M$ испытаний. Тем самым общее число выполненных испытаний составит $K{\cdot}M$.

Данные, представленные на рис. 2 и 3, отражают тот факт, что за счет выбранного принципа параллельной реализации алгоритма глобального поиска в его независимо выполняемых частях не происходит обменов информацией и неизбежно возникает избыточность по числу измерений.

Верхняя кривая соответствует однопоточной реализации, которая обладает наибольшей надежностью при заданном общем количестве выполняемых измерений целевой функции. Увеличение числа параллельных потоков, каждый из которых связывался со своим интервалом первоначального разбиения области поиска $[0;1]$, приводило к росту количества избыточных измерений.

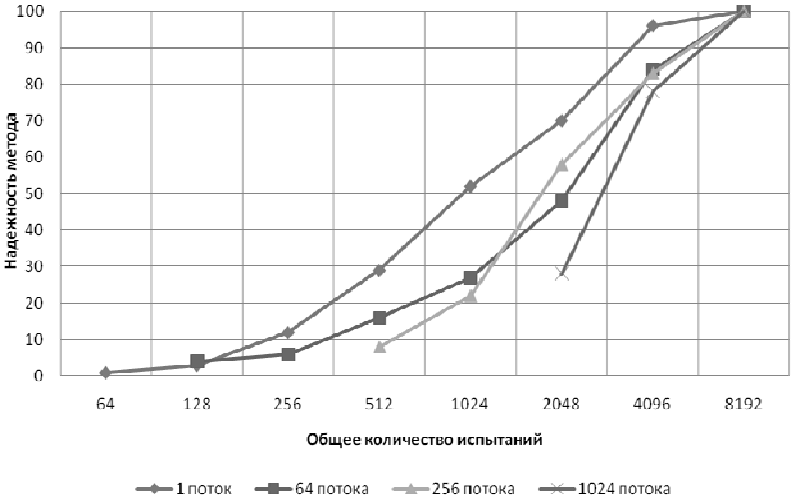



Рис. 2. Простой класс функций

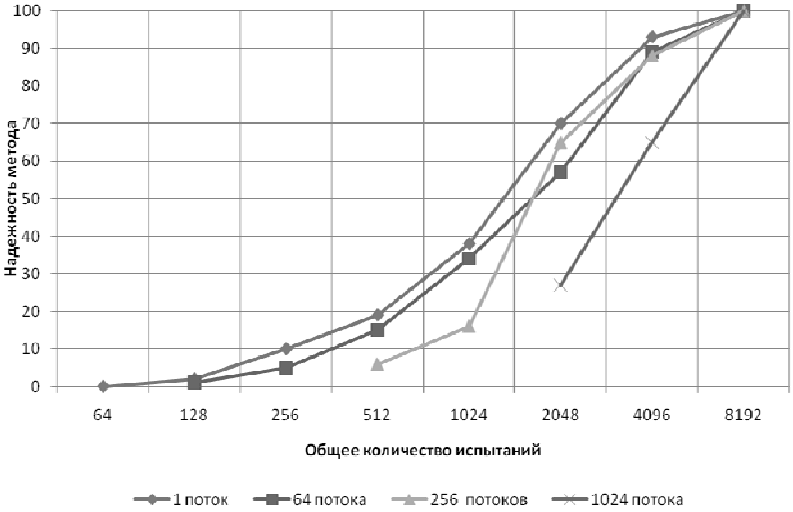



Рис. 3. Сложный класс функций

Оценим коэффициент избыточности, сравнивая затраты по числу измерений целевой функции в многопоточном и однопоточном вариантах реализации на GPU при уровне надежности

0,88 (для «простого» класса функций) и 0,89 (для «сложного» класса функций). По этой методике для значения числа потоков $M = 512$ получим коэффициент избыточности равный 1,1 (для «простого» класса функций) и 1,22 (для «сложного» класса функций). Наибольший коэффициент избыточности за весь период работы алгоритмов наблюдался при уровне надежности 0,5–0,6 и был примерно равен 2 (для сложного класса функций) и ≈4 (для простого класса функций).

Следует учитывать, что при параллельном исполнении программного кода в потоках при выполнении очередной итерации в одном потоке (приводящей к вычислению значения целевой функции) в среднем одновременно выполняются аналогичные итерации и в остальных потоках. Таким образом, за период одной итерации в среднем выполняется *M* вычислений значений функции (по количеству выполняемых потоков). Следовательно, с точки зрения обобщенно понимаемого времени ожидания результата решения следует рассматривать операционные характеристики иного, чем на рис. 2, 3 вида, а именно, необходимо перейти к характеристикам типа «надежность – число итераций», представленным в табл. 2, 3. В этом случае становится очевидным преимущество параллельных реализаций алгоритма.

Таблица 2

**Простой класс функций**

| Количество потоков | Количество испытаний в одном потоке | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| 32 | 3 | 5 | 11 | 18 | 36 | 56 | 83 |
| 64 | 4 | 6 | 16 | 27 | 48 | 84 | 100 |
| 128 | 6 | 17 | 36 | 65 | 85 | 99 | 100 |
| 256 | 8 | 22 | 58 | 83 | 100 | 100 | 100 |
| 512 | 16 | 36 | 88 | 100 | 100 | 100 | 100 |
| 1024 | 28 | 78 | 100 | 100 | 100 | 100 | 100 |

Таблица 3

**Сложный класс функций**

| Количество потоков | Количество испытаний в одном потоке | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| 32 | 0 | 3 | 5 | 14 | 31 | 63 | 88 |
| 64 | 1 | 5 | 15 | 34 | 57 | 89 | 100 |
| 128 | 3 | 8 | 25 | 56 | 89 | 100 | 100 |
| 256 | 6 | 16 | 65 | 88 | 100 | 100 | 100 |
| 512 | 20 | 40 | 89 | 99 | 100 | 100 | 100 |
| 1024 | 27 | 65 | 100 | 100 | 100 | 100 | 100 |

Если, например, проанализировать ускорение, вытекающее из табл. 2, 3, то при уровне надежности 1,0 и увеличении количества потоков с 1 до 128, мы получим сокращение числа итераций примерно с 6000 до 128, что показывает ускорение по количеству итераций в 48 раз.

При проведении исследований дополнительно были выполнены оценки ускорения непосредственно по времени. При этом замерялось среднее время до момента определения с заданной точностью $\rho = 0,02$ глобально-оптимального решения (или до исчерпания установленного максимального числа итераций, если решение не определялось). Усреднение проводилось по всем 100 функциям GKLS-генератора. Максимальное число итераций на один поток выбиралось так, чтобы обеспечивался уровень надежности 1,0 (правильное решение 100 задач из 100). В качестве тестового стенда при замерах времени использовалась: GPU- ATI Radeon 4890, CPU – Intel® Core™ 2 Duo, E6559, 2,33 GHz, 4.00 Gb RAM, 64 bit OS. В данной реализации в качестве основного инструмента выбрана технология OpenCL.

Усредненные значения времени были получены для нескольких значений числа $M$ используемых параллельных потоков на GPU. Коэффициент ускорения вычислялся как отношение среднего времени решения на CPU (1 поток) к среднему времени при $M > 1$. Как видно из графика, ускорение составляет 55–60 раз по сравнению с последовательной версией.

Как видно из результатов, использование графических процессоров для вычислений общего назначения в который раз доказало свою эффективность. Конечно, мы получаем определенную избыточность в вычислениях, однако ускорение по времени с ростом числа потоков неуклонно растет. Приведенные результаты показывают, что использование графических карт в данной области оправданно (ускорение работы до 55–60 раз), однако требуется более детально рассмотреть вопросы, связанные с точностью вычислений, эффективным использованием ресурсов самой видеокарты. В связи с этим интересно протестировать некоторые модификации данного алгоритма. К примеру, разбить исходный гипериинтервал на подгипериинтервалы и в каждом из них реализовать отдельную развертку. Также необходимо протестировать работу для больших размерностей (к примеру, 3–5).

Работа выполнена при финансовой поддержке Федерального агентства по науке и инновациям (ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.5018).

## Список литературы

1. Городецкий С.Ю., Гришагин В.А. Модели и методы конечномерной оптимизации: учеб. курс. Ч. 2. Нелинейное программирование и многоэкстремальная оптимизация. – Н. Новгород, 2003.

2. Lera D., Sergeyev Ya.D. (2010) Lipschitz and Holder global optimization using space-filling curves // Applied Numerical Mathematics, 60(1–2), 115–129.

3. Software for generation of classes of test functions with known local and global minima for global optimization / M. Gaviano [et al.] // ACM TOMS 29 (4) (2003) 469–480.

4. Гришагин В.А. Параллельные алгоритмы многоэкстремальной оптимизации в областях с вычислимой границей. Высокопроизводительные параллельные вычисления на кластерных системах. – Н. Новгород, 2007.

**Р.Ф. Кулеев**

Казанский федеральный университет

# ОБ ЭФФЕКТИВНОСТИ РАСПАРАЛЛЕЛИВАНИЯ НЕКОТОРЫХ АЛГОРИТМОВ ДЕШИФРИРОВАНИЯ ИЗОБРАЖЕНИЙ

Для решения задач дешифрирования изображений в первую очередь необходимы теоретические основы, с помощью которых можно строить алгоритмы. В работах [1, 2, 3] были предложены математическая модель сцены, а также некоторые алгоритмы для поиска объектов.

Современные алгоритмы дешифрирования нуждаются в значительных вычислительных ресурсах. Это становится особенно актуальным, когда задачи дешифрирования требуется решать в режиме реального времени. Сегодня все большее распространение получают высокопроизводительные системы, состоящие из большого количества вычислительных элементов. Для повышения производительности системы до требуемого уровня достаточно добавить в нее новые вычислительные элементы или выбрать систему с необходимым количеством вычислительных узлов. Зачастую это не представляет большой сложности, являясь сугубо технической задачей.

В таком случае возникает задача распараллелить алгоритмы дешифрирования для их выполнения на параллельных вычислительных системах. Очевидно, что с увеличением количества вычислительных элементов системы ее производительность, а следовательно, и скорость выполнения алгоритма должны возрастать. Чем быстрее растет ускорение алгоритма при добавлении вычислительных элементов, тем эффективнее будем считать степень его распараллеливания. В данной работе рассматривается эффективность распараллеливания алгоритмов сглаживания, поиска зон интереса и сегментации.

**Вычислительная система и программное обеспечение.** Для исследования эффективности распараллеливания алгоритмов проводились эксперименты на кластере HP BLc3000 Twr CTO Enclosure. Кластер объединяет 4 узла с общей памятью.

Узел содержит 2 процессора Intel Xeon 5410 по 4 ядра каждый. Таким образом, общее количество ядер кластера равно 32. Ядра в процессоре попарно динамически разделяют кэш 2-го уровня общим объемом 12 Мб (по 6 Мб на пару ядер). Также каждое ядро имеет кэш 1-го уровня размером 64 Кб. Два ядра узла, в зависимости от взаимного расположения, могут обмениваться информацией либо через кэш 2-го уровня, либо через шину. Все 8 ядер имеют доступ к общей для узла оперативной памяти объемом 8 Гб. Вычислительные узлы соединены сетью Gigabit Ethernet. Кластер работает под управлением операционной системы Linux.

Программное обеспечение разрабатывалось на языке программирования C#. Для распараллеливания использовалась библиотека MPI.NET, которая была сконфигурирована для вызова функций Open MPI. Для запуска программного обеспечения на кластере, работающего под ОС Linux, была использована платформа Mono.

Очевидно, что время работы параллельной программы состоит из «чистого» времени выполнения алгоритма, а также времени, требуемого на подготовку и пересылку данных. Время выполнения программы на $N$ вычислительных элементах без учета времени на пересылку будем обозначать $t_N$, а совокупное время работы программы – $t'_N$. Одним из наиболее распространенных критериев эффективности распараллеливания алгоритма является его ускорение $S_N = \frac{t_1}{t_N}$.

Рассмотренный кластер позволяет смоделировать архитектуры вычислительных систем с общей и локальной памятью. В данной работе для алгоритмов экспериментальным путем выбиралась оптимальная архитектура вычислительной системы, а также анализировалось ускорение системы при добавлении новых вычислительных элементов. Для этого программное обеспечение выполнялось сначала на архитектуре с локальной памятью. Далее те же программы запускались лишь на одном

узле кластера, на архитектуре с общей памятью. При этом для распараллеливания использовался механизм MPI. И, наконец, на архитектуре с общей памятью выполнялись программы, распараллеливаемые с помощью потоков.

**Сглаживание.** Сглаживание является одной из наиболее распространенных операций, используемых при дешифрировании изображений. В разработанной программе для сглаживания с радиусом $r$ использовалась окрестность квадратной формы со стороной $2r + 1$. В проведенных экспериментах сглаживание с радиусом 4 проводилось одновременно по 9 квадратным изображениям со стороной 2048 пикселей. Для распараллеливания исходное изображение разделялось на полосы, каждая из которых обрабатывалась отдельным вычислительным элементом. В табл. 1 приведены результаты распараллеливания алгоритма сглаживания на архитектуре с локальной памятью (I), на архитектуре с общей памятью с помощью MPI (II), а также на архитектуре с общей памятью с помощью потоков (III). Сглаживание зачастую выполняется внутри других алгоритмов дешифрирования. Поэтому важным критерием эффективности распараллеливания данной операции является $S_N$.

Таблица 1

**Сглаживание**

| Архитектура памяти | $N$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| I | $t'_N$, мс | 25732 | 15967 | 12617 | 10712 |
| | $t_N$, мс | 21122 | 10519 | 7029 | 5275 |
| | $S_N$ | 1,00 | 2,01 | 3,00 | 4,00 |
| II | $t'_N$, мс | 25579 | 15623 | 12053 | 10264 |
| | $t_N$, мс | 21294 | 10578 | 7035 | 5254 |
| | $S_N$ | 1,00 | 2,01 | 3,03 | 4,05 |
| III | $t'_N$, мс | 27468 | 16269 | 12349 | 10289 |
| | $t_N$, мс | 23212 | 11976 | 8058 | 5999 |
| | $S_N$ | 1,00 | 1,94 | 2,88 | 3,87 |

Результаты показывают высокую эффективность распараллеливания алгоритма сглаживания: во всех экспериментах ускорение изменяется практически пропорционально росту числа вычислительных элементов. Сравнение MPI и потоков на архитектуре с общей памятью отдает предпочтение механизму MPI. Видимо, причиной этого является отсутствие привязки потоков к конкретным ядрам процессора, вследствие чего возникают затраты ресурсов на переключение задачи между ядрами. Кроме того, при использовании MPI лучшей оказывается локализация данных в памяти. При использовании MPI на архитектуре с общей памятью наблюдается суперлинейное ускорение, вызванное тем, что с увеличением числа ядер все большая часть данных умещается в кэш-памяти.

Из-за необходимости пересылки данных между узлами кластера $t'_N$ на архитектуре с локальной памятью оказалось больше $t'_N$, полученного на архитектуре с общей памятью. Однако время выполнения операции сглаживания $t_N$ в обоих случаях практически совпадает. Было сделано предположение, что это обеспечивается за счет эффективной работы кэш-памяти. Поэтому для исследования влияния кэш-памяти на результаты работы программы были проведены эксперименты, в которых максимально снижалась эффективность работы кэш-памяти второго уровня. Обход сглаживаемых пикселей проводился не по порядку, а случайным образом. В табл. 2 приведены результаты экспериментов, аналогичных рассмотренным ранее.

Таблица 2

**Сглаживание без кэш-памяти**

| Архитектура памяти | $N$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| I | $t_N$, мс | 42365 | 17497 | 10880 | 7398 |
| | $S_N$ | 1,00 | 2,42 | 3,89 | 5,73 |
| II | $t_N$, мс | 42689 | 17937 | 12497 | 7788 |
| | $S_N$ | 1,00 | 2,38 | 3,42 | 5,48 |

Случайный порядок обхода сглаживаемых пикселей увеличил время работы программы, так как необходимые для процессора данные, как правило, не оказываются в кэш-памяти. Суперлинейность ускорения стала еще более выраженной. Также незначительно увеличилась разница во времени выполнения программы на вычислительных системах с различной архитектурой.

**Поиск зон интереса.** Для дешифрирования объектов по геометрическим признакам необходимо уметь находить их проекции. Для этого существует операция сегментации. Операцию сегментации можно значительно упростить, если проводить ее в зоне интереса – квадрате, содержащем только один объект, не имеющий общих точек с границей [2]. При поиске зон интереса строится множество квадратов, гарантированно содержащих проекции заданных объектов. Далее среди них выбираются те, которые наиболее вероятно содержат объект. В качестве решающего правила для классификации квадратов предлагается использовать признак пятна [3]. Для распараллеливания операции поиска зон интереса исходное изображение разделялось на полосы с частичным перекрытием, чтобы проекция каждого объекта сцены полностью попадала в одну из полос. Каждая из полос обрабатывалась отдельным вычислительным элементом.

Для исследования эффективности распараллеливания операции поиска зон интереса были проведены эксперименты, аналогичные экспериментам с операцией сглаживания. Поиск зон интереса со стороной $l = 55$ пикселей выполнялся одновременно по трем изображениям размером 1024 на 1024 пикселя. Наибольшая эффективность распараллеливания была достигнута при использовании MPI на архитектуре с общей памятью. Результаты экспериментов приведены в табл. 3.

Таблица 3

**Поиск зон интереса (общая память, MPI)**

| $N$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $t_N$, мс | 22102 | 11143 | 7000 | 5207 |
| $S_N$ | 1,00 | 1,98 | 3,16 | 4,24 |

**Сегментация.** Исследование эффективности распараллеливания операции сегментации было проведено на примере обобщенного метода пятна [3]. Для распараллеливания исходная зона интереса разделялась на полосы. Сегментация зоны интереса со стороной $l = 55$ пикселей проводилась одновременно по трем изображениям, сглаженным с радиусом $r = 1$. Наибольшая эффективность распараллеливания была достигнута при использовании механизма MPI на вычислительной системе с общей памятью (табл. 4).

Таблица 4

**Сегментация**

| $N$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $t'_N$, мс | 62 | 218 | 221 | 225 |
| $t_N$, мс | 4,90 | 3,49 | 2,85 | 2,80 |
| $S_N$ | 1,00 | 1,40 | 1,72 | 1,75 |

В отличие от результатов сглаживания и поиска зон интереса, в проведенных экспериментах наблюдается очень медленный рост ускорения $S_N$. Это связано с латентностью используемой системы. Так, только вызов метода и выполнение сегментации одного пикселя на одном вычислительном элементе занимает около 0,4 мс. Это оказывает существенное влияние на время выполнения сегментации всей зоны интереса и, следовательно, на ускорение. Если выполнять сегментацию зоны интереса со стороной 1024 пикселя, то влияние латентности системы становится незначительным, и ускорение $S_N$ изменяется практически пропорционально с ростом $N$.

Если известно минимальное расстояние $d_{\min}$ между проекциями объектов сцены, то появляется возможность ускорить операцию поиска зон интереса. Для этого в найденной зоне интереса необходимо провести сегментацию и по полученной проекции принять решение о наличии объекта. Если наличие объекта подтверждается, то поиск других зон интереса можно выполнять на расстоянии $d_{\min}$ от найденной. В таком случае операция сегментации становится составной частью поиска зон интереса.

Возникает вопрос о целесообразности распараллеливания сегментации или выполнения последовательного варианта алгоритма. Очевидно, что для зоны интереса с $l$ = 55 на используемой вычислительной системе алгоритм сегментации распараллеливать нецелесообразно, так как $t_1 < t_N^{'} \forall N = \overline{1,4}$. Для определения стороны зоны интереса $l$ распараллеливание сегментации которой целесообразно, были проведены эксперименты с различным $l$ и приближенно найдены зависимости $t_N$ и $t'_N$ от $l$. Оказалось, что при $l > 1207$ находятся $N$, при которых $t_1 > t_N^{'}$. Следовательно, в таком случае целесообразнее проводить распараллеливание сегментации.

Сравнение различных архитектур параллельных вычислительных систем и подходов к распараллеливанию показало, что для рассмотренных алгоритмов дешифрирования наибольшая эффективность распараллеливания достигается при использовании механизма MPI на вычислительной системе с общей памятью. Исследовано влияние латентности на ускорение алгоритма сегментации, а также даны рекомендации о целесообразности распараллеливания алгоритма.

## Список литературы

1. Фофанов В.Б. Формализация сцены в задаче дешифрирования многозональных изображений // Оптический журнал. – 2007. – Т. 74, № 3. – С. 51–54.

2. Фофанов В.Б., Демченко А.В., Кулеев Р.Ф. Дешифрирование многозональных изображений: методы и результаты // Оптический журнал. – 2007. – Т. 74, № 3. – С. 55–59.

3. Burylin S.A., Fofanov V.B. Generalized Spot Criterion as Applied for Image Deciphering // Pattern Recognition and Image Analysis. – 2004. – Vol. 14, No. 2. – P. 243–248.

**А.Г. Кучумов**

Пермский государственный технический университет

# МНОГОУРОВНЕВЫЙ ПОДХОД В ЗАДАЧАХ БИОМЕХАНИКИ. ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ КЛАСТЕРНЫХ СИСТЕМ ДЛЯ РЕШЕНИЯ ДАННЫХ ЗАДАЧ

Традиционно в механике и биомеханике объект исследования рассматривается как макрообъект. На данном уровне общепринятым является применение аппарата механики сплошной среды, который позволяет рассчитывать макроскопические характеристики (относительное удлинение, пределы пластичности и текучести и т.д.). Однако многие материалы, в том числе биологические ткани и органы, которые на макроуровне являются однородными, на мезо- микро- и наноуровнях обладают гетерогенной структурой, которая определяет их макроскопическое поведение. Поэтому для тотального изучения того или иного биологического объекта необходимо рассматривать его поведение на макро-, мезо-, микро- и наноуровнях (т.е. использовать многоуровневый подход).

Многоуровневое моделирование заключается в том, что сначала строится модель поведения материала на самом «мелком» уровне, а затем осуществляется переход на более «крупный» уровень, сохраняя информацию о свойствах и механическом отклике, относящуюся к предыдущему уровню (рис. 1). В конечном счете получается макромодель исследуемого объекта.

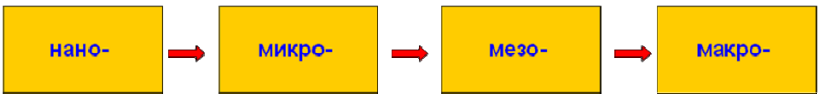



Рис. 1. Многоуровневый подход «снизу – вверх»

Первоначально идея многоуровневого моделирования была выдвинута в работах по моделированию материалов с внутренними структурными неоднородностями (включениями,

трещинами и т.д.). В дальнейшем данный подход стал использоваться в задачах биомеханики, таких как:

- изучение ростовых и регенерационных процессов;
- исследование кости с учетом нанобиокомпозитной структуры;
- анализ микроструктуры мягких тканей;
- гемодинамика сердечно-сосудистой сети;
- биотрибология;
- исследования в области онкологических заболеваний.

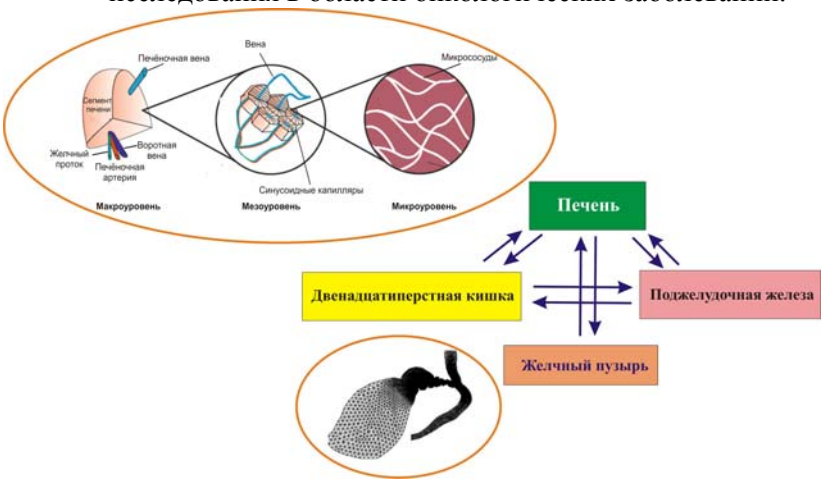



Рис. 2. Билиарная система

Использование многоуровневого подхода в исследовании функционирования различных систем в норме и при патологиях является одним из перспективных направлений биомеханики. Для моделирования функционирования биологических систем в норме и при патологиях использование упрощённых аналитических моделей практически невозможно, так как, во-первых, они строятся при большом количестве допущений, во-вторых, они будут громоздкими при учёте различных эффектов, которые присущи сложным биологическим объектам, и, в-третьих, они не всегда учитывают влияние микроструктуры, которая играет

важную роль. Учёт влияния микроструктуры на поведение макрообъекта влечёт за собой необходимость использования численных методов и средств, реализующихся на вычислительных кластерах. В качестве примера в данной работе рассматривается модель билиарной системы (рис. 2), включающей в себя такие элементы, как печень, желчный пузырь, желчные протоки и поджелудочная железа. Рассматривается актуальность изучения её элементов, начиная со сложной микроструктуры с переходом в дальнейшем на более высокие уровни. Постулируется класс задач [связанные задачи о взаимодействии жидкости и твердого тела (fluid–solid interaction), контактные задачи и т.д.] и методы, которые необходимо использовать при моделировании поведения данной системы в норме и при патологиях.

**А.А. Лабутина**

Нижегородский государственный университет им. Н.И. Лобачевского

## ОПТИМИЗАЦИЯ ТЕСТОВ HPC CHALLENGE BENCHMARK SUITE ДЛЯ ГЕТЕРОГЕННЫХ КЛАСТЕРОВ

В этом году студенческая команда Нижегородского государственного университета им. Н.И. Лобачевского примет участие в соревновании Student Cluster Competition, которое будет проходить в рамках международной конференции Supercomputing–2010 в Новом Орлеане, США. Команда ННГУ выступает при поддержке компаний Microsoft, IBM и NVidia.

При подготовке к соревнованию каждая команда в сотрудничестве с консультантами и спонсорами разрабатывает архитектуру своего передового высокопроизводительного вычислительного кластера. Кластер команды ННГУ предоставлен компанией IBM, крупнейшим в мире производителем серверов. Кластер состоит из шести серверов IBM iDataPlex dx360 M3 с центральными 6-ядерными процессорами Intel Xeon серии 5600, укомплектованных двумя вычислительными модулями Nvidia Tesla M2050. Модуль общего назначения Tesla M2050, включающий 448 процес-

сорных ядер и оснащенный 3 Гб выделенной памяти GDDR5, предлагает скорость работы до 515 GFlops для вычислений с двойной точностью (1030 GFlops для вычислений с одинарной точностью). По оценке IBM, по сравнению с серверами предыдущих поколений серверы iDataPlex dx360 M3 обеспечивают на 40 % улучшенное потребление энергии и до 5 раз увеличенную производительность.

Основная задача команды на соревновании – за отведенное время правильно выполнить максимальное число запусков заранее определенных параллельных приложений на тех входных данных, которые предоставляются организаторами. Для вычислительной системы действуют ограничения на максимальную потребляемую мощность.

В качестве предварительного этапа соревнования командам предстоит выполнить запуск набора тестов HPC Challenge Benchmark Suite (HPCC) и получить максимально возможный результат. Команде, которой удастся добиться наибольшей производительности, начисляются дополнительные очки.

- HPCC представляет собой средство оценки и исследования производительности высокопроизводительных систем, реализующее более сложные модели доступа к оперативной памяти, нежели тест High Performance Linpack. В состав HPCC входят тесты *HPL*, *DGEMM*, *STREAM*, *PTRANS*, *Random Access*, *FFT* и *Communication bandwidth and latency*.
- *HPL* оценивает производительность при решении системы линейных уравнений.
- *DGEMM* оценивает производительность при выполнении матричного умножения.
- *STREAM* оценивает пропускную способность оперативной памяти на основе четырёх простейших векторных операций с векторами большой длины (COPY, SCALE, ADD, TRIAD).
- *PTRANS* оценивает пропускную способность сети и памяти на задаче параллельного транспонирования матриц.
- *Random Access* оценивает скорость случайного доступа к оперативной памяти.

• *FFT* оценивает производительность при выполнении одномерного быстрого преобразования Фурье.

• *Communication bandwidth and latency* оценивает пропускную способность сети и величину латентности в трёх моделях коммуникации между процессами – ping-pong, ring, random ring.

Многие из этих тестов представляют собой уже давно зарекомендовавшие себя вычислительные ядра (computation kernels), распространённые в среде специалистов по высокопроизводительным вычислениям. Однако с целью более точного и тонкого тестирования авторами были внесены коррективы, проводящие верификацию данных и вывод извлечённых характеристик в наглядной форме.

Таким образом, чтобы увеличить свои шансы на победу, команде ННГУ необходимо произвести перенос части вычислений, которые выполняются в тестах HPCC, на GPU.

В данной статье мы представим подход, который используется для увеличения производительности теста Linpack при помощи технологии CUDA. В состав пакета для разработчика CUDA Toolkit входят профилировщик, отладчик и широко используемые библиотеки для высокопроизводительных вычислений:

• библиотека CUBLAS – реализация библиотеки операций линейной алгебры BLAS;

• библиотека CUFFT – библиотека функций для выполнения быстрого преобразования Фурье.

Подход, представленный в данной статье, предполагает использование библиотеки CUBLAS и не требует программирования дополнительных CUDA-функций.

Тест Linpack решает случайно заданную систему линейных уравнений методом LU-разложения с выбором ведущего элемента столбца, используя 64-битную арифметику с плавающей запятой, и позволяет судить о производительности системы при решении такой задачи. Наибольшее время при выполнении теста Linpack затрачивается на выполнение операции матричного умножения (DGEMM), причем с ростом размерности матрицы время, необходимое на выполнение данной операции, также

растет. Таким образом, оптимизация матричного умножения – основной шаг к увеличению производительности теста Linpack.

Матричное умножение – одна из основных операций при реализации многих алгоритмов. Интерфейс BLAS предусматривает наличие нескольких функций для выполнения этой операции: для случая, если умножение выполняется над матрицами с элементами типа double, предусмотрена функция DGEMM.

Если *A[M, K]*, *B[K, N]* и *C[M, N]* – входные данные, то вызов функции DGEMM выполнит операцию $C = \alpha AB + \beta C$. Интерфейс BLAS предусматривает возможность задания размеров ведущих измерений (*leading dimension*) матриц *A*, *B* и *C* (*lda*, *ldb* и *ldc*) отличными от *M*, *K* и *N*. Для выполнения представленной задачи умножения матриц необходимо вызвать функцию DGEMM со следующими аргументами:

DGEMM('N', 'N', M, N, K, alpha, A, lda, B, ldb, beta, C, ldc).

Выполнение матричного умножения – одна из немногих задач, где процессор может продемонстрировать производительность, близкую к пиковой. Для функции DGEMM существует несколько высокопроизводительных реализаций, например, реализация в рамках библиотеки GotoBLAS, разработанной в техасском университете, а также реализации от производителей процессоров, такие как Intel MKL и AMD ACML.

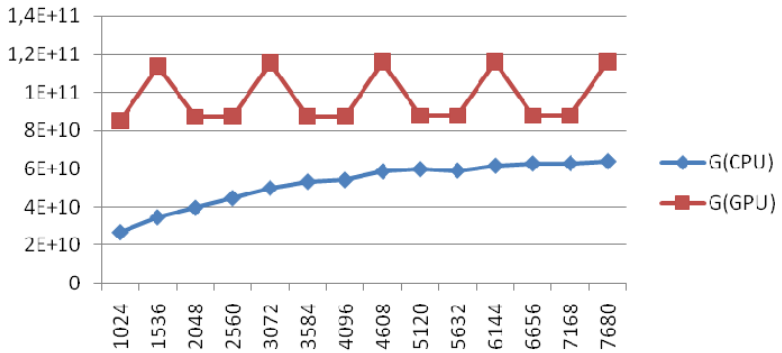



Рис. 1. Выполнение операции DGEMM на CPU и GPU

На тестовом сервере с двумя процессорами Intel Xeon серии 5600 и двумя графическими модулями nVidia Tesla M2050 были проведены эксперименты для определения производительности CPU и GPU при выполнении функции DGEMM. Для выполнения умножения матриц на центральном процессоре использовалась функция cbas_Dgemm, реализованная в библиотеке IntelMKL. Для выполнения умножения матриц на графическом процессоре использовалась реализация из библиотеки CUBLAS. Результаты экспериментов представлены на рис. 1.

Идея, которая использована в данной работе для ускорения теста Linpack, очень проста: для того чтобы ускорить выполнение операции матричного умножения, исходные матрицы C и B представляются как объединение двух подматриц $C=C_1UC_2$ и $B=B_1UB_2$. В этом случае операция DGEMM

$$C = \alpha AB + \beta C$$

может быть представлена следующим образом:

$$C = \alpha(AB_1+AB_2)+\beta(C_1+C_2),$$

а вызов функции разделяется на два последовательных вызова (рис. 2):

DGEMM('N', 'N', M, $N_1$, K, alpha, A, lda, $B_1$, ldb, beta, $C_1$, ldc)
DGEMM('N', 'N', M, $N_2$, K, alpha, A, lda, $B_2$, ldb, beta, $C_2$, ldc).

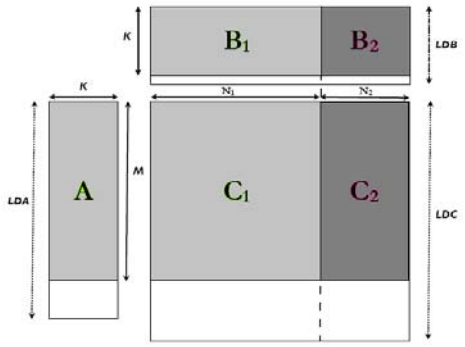



Рис. 2. Разделение вычислительной нагрузки между CPU и GPU при выполнении операции умножения матриц

Поскольку эти две операции независимы, мы можем выполнить первую на GPU а вторую – на CPU. Как только необходимые данные переданы в память графического процессора, мы можем параллельно выполнять операции DGEMM, как это показано на рис. 3. Кроме того, следует отметить, что при выполнении разделения вычислительной нагрузки можно подобрать размеры частей матрицы таким образом, чтобы производительность графического процессора достигала наибольшего возможного значения.

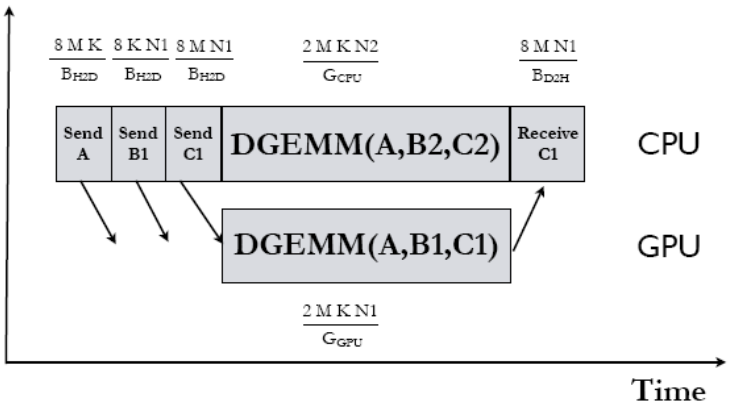



Рис. 3. Схема параллельного выполнения операции матричного умножения на CPU и GPU

Используя специализированные функции библиотеки CUBLAS для перемещения данных между оперативной памятью и памятью графического процессора, реализовать подобный подход не представляет труда:

```
// Copy A from CPU memory to GPU memory devA
status = cublasSetMatrix (m, k , sizeof(A[0]), A,
lda, devA, m_gpu);
// Copy B1 from CPU memory to GPU memory devB
status = cublasSetMatrix (k ,n_gpu, sizeof(B[0]),
B, ldb, devB, k_gpu);
// Copy C1 from CPU memory to GPU memory devC
```

```
status = cublasSetMatrix (m, n_gpu, sizeof(C[0]),
C, ldc, devC, m_gpu);
// Perform DGEMM(devA,devB,devC) on GPU
// Control immediately return to CPU
cublasDgemm('n', 'n', m, n_gpu, k, alpha, devA,
m,devB, k, beta, devC, m);
// Perform DGEMM(A,B2,C2) on CPU
dgemm_cpu('n','n',m,n_cpu,k,        alpha,        A,
lda,B+ldb*n_gpu, ldb,
  beta,C+ldc*n_gpu, ldc);
// Copy devC from GPU memory to CPU memory C1
status = cublasGetMatrix (m, n, sizeof(C[0]), devC,
m, C, *ldc);
```

Для того чтобы максимизировать производительность, мы должны определить наилучшее разделение объема вычислений между CPU и GPU. Необходимо учитывать, что дополнительное время будет затрачено на перемещение данных между памятью центрального процессора и графического процессора. Требуется измерить значения следующих величин:

- $B_{H2D}$ – величина пропускной способности канала при передаче данных из оперативной памяти в память графического процессора (GB/s);
- $G_{GPU}$ – производительность операции DGEMM на графическом процессоре (GFlops);
- $G_{CPU}$ – производительность операции DGEMM на центральном процессоре (GFlops);
- $B_{D2H}$ – величина пропускной способности канала при передаче данных из памяти графического процессора в оперативную память (GB/s);

Вызов функции DGEMM на CPU выполняет 2KMN операций, и, следовательно, если центральный процессор может выполнять эти операции со скоростью GCPU, то время, необходимое для вычислений

$$T_{CPU}(M,K,N) = 2\frac{MKN}{G_{CPU}}.$$

Общее время, необходимое на выполнение части операции матричного умножения на GPU, состоит из времени ввo-

да/вывода, которое тратится на передачу данных в память графического процессора и обратно, и времени вычислений:

$$T_{GPU}(M,K,N) = 8\frac{(MK + KN + MN)}{B_{H2D}} + 2\frac{MKN}{G_{GPU}} + 8\frac{MN}{B_{D2H}}.$$

Оптимальное разделение вычислительной нагрузки между CPU и GPU достигается в случае

$$T_{CPU}(M,K,N) = T_{GPU}(M,K,N_1),$$

где

$$N = N_1 + N_2.$$

Для упрощения полученных оценок величины оптимального разделения $\mu = N_1/N$ мы можем принебречь временем передачи данных $O(N^2)$ в сравнении со временем вычислений $O(N^3)$:

$$\mu = \frac{G_{GPU}}{G_{GPU} + G_{CPU}}.$$

Для тестовой системы, где центральный шестиядерный процессор Intel Xeon имеет производительность на операции DGEMM 60 GFlops, а графический процессор – 115 GFlops, данная формула позволяет определить оптимальное разделение $\mu = 0{,}65$.

Подобный механизм разделения вычислений необходимо реализовать также для выполнения операции DTRSM.

Одной из важных характеристик представленного подхода является то, что для его реализации не нужно вносить каких-либо изменений в исходный код тестов HPCC. Достаточно реализовать библиотеку-обертку, которая перехватывает вызов функции DGEMM и DTRSM и выполняет распределение вычислений между CPU и GPU.

Выполнение некоторых других тестов из набора HPCC также может быть оптимизировано с использованием технологии CUDA. Планируется разработать оптимизированную версию тестов DGEMM и STREAM.

### Список литературы

1. http://www.top500.org

2. HPL – a portable implementation of the high performance Linpack benchmark for distributed memory computers, version 2.0 / Petitet A. [et al.]. – URL: http://www.netlib.org/benchmark/hpl.

3. NVIDIA CUDA Compute Unified Device Architecture Programming Guide.

4. The HPC Challenge (HPCC) Benchmark Suite / P. Luszczek [et al.]. – URL: http://icl.cs.utk.edu/projectsfiles/hpcc/pubs/sc06_hpcc.pdf.

5. Fatica M. Accelerating Linpack with CUDA on heterogenous clusters // ACM International Conference Proceeding Series. Vol. 383.

6. Dongarra J.J. Performance of various computers using standard linear equations software. Technical report, 2008. – URL: http://www.netlib.org/benchmark/performance.pdf.



**А.А. Лабутина, А.В. Линев**

Нижегородский государственный университет им. Н.И. Лобачевского


## ОПТИМИЗАЦИЯ ПРИКЛАДНЫХ ПАКЕТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КЛАСТЕРА КОМПОЗИТНОЙ АРХИТЕКТУРЫ В РАМКАХ ПОДГОТОВКИ К СТУДЕНЧЕСКОМУ СОРЕВНОВАНИЮ STUDENT CLUSTER COMPETITION–2010

В ноябре 2010 года будет проходить SC10 (SuperComputing–2010) – международная конференция, посвященная высокопроизводительным вычислениям, сетям, системам хранения и исследованиям [1]. Одним из пунктов программы конференции является Student Cluster Competition (SSC) – соревнование, в котором участвуют команды, состоящие из шести студентов, проходящее в режиме реального времени на территории выставки [2]. В ходе состязания команды должны продемонстрировать использование для решения предоставленного организаторами набора задач ряда распространенных прикладных пакетов с открытым кодом, установ-

ленных на небольшом коммерчески доступном кластере с ограниченным электропотреблением (26 А, 120 В).

В 2010 году соревнование проходит в четвертый раз, и на нем при поддержке Microsoft выступит команда Нижегородского государственного университета им. Н.И. Лобачевского (ННГУ).

**Условия соревнования.** Команда из шести студентов, имеющая в распоряжении небольшой кластер, в течение ограниченного времени (48 часов) должна выполнить корректные расчеты для как можно большего числа наборов входных данных, предоставленных организаторами. В ходе подготовки к соревнованию команда совместно со своим наставником и спонсорами должна построить из коммерчески доступных компонентов кластер, потребляющий ток силой не более 26 ампер (120 В), и добиться на нем максимальной производительности четырех заранее определенных приложений и теста производительности HPCC.

Команда ННГУ выступает при поддержке компаний Microsoft и IBM. IBM предоставил команде ННГУ кластер, состоящий из 6 серверов IBM System x iDataPlex dx360 M3 (2 Intel Xeon L5640 2.26 ГГц, 12MB кэш, 1333FSB, 6x4 GB PC3L-10600 DDR3 DIMMS, 2 Nvidia Tesla M2050 GPU, Mellanox ConnectX2 VPI dual port QDR IB, 250 Гб 7200 3.5” SATA HDD), объединенных посредством сети Infiniband. Основная вычислительная мощность данного кластера заключена в графических процессорах NVidia Tesla M2050 (теоретическая производительность одной карты при использовании одинарной точности – 1.03 Tflops, двойной точности – 515 Gflops)[3].

В ходе соревнования командам будут предлагаться задачи, решаемые посредством следующих пакетов:

- FLASH – моделирование термоядерных вспышек в астрофизике [4];
- WRF (The Weather Research and Forecasting) – расчет прогноза погоды [5];
- NAMD (NAnoscale Molecular Dynamics) – параллельный пакет молекулярной динамики, разработанный для высокопроизводительной симуляции больших биомолекулярных систем [6];

Password Recovery – перебор паролей, зашифрованных алгоритмами MD5 и Blowfish; участникам предоставляется право самостоятельно выбрать реализацию либо разработать собственную.

Отдельно будут произведены испытания кластеров на производительность посредством HPCC Benchmark.

Таким образом, требуется обеспечить высокую пропускную способность при решении перечисленных задач на кластере композитной архитектуры. В качестве дополнительного условия для команды ННГУ выступает необходимость использования операционной системы семейства MS Windows.

**Подготовка пакетов.** Основной задачей команды на период подготовки является проведение исследований, направленных на обеспечение высокопроизводительной работы заданного набора приложений на предполагаемом кластере. Относительно пакетов FLASH, WRF и NAMD подготовка включает следующие этапы.

- Изучение предметной области проекта и назначения пакета, определение используемых математических моделей и численных методов.
- Определение архитектуры и состава пакета, порядка его использования, используемых алгоритмов и структур данных, а также параметров, влияющих на время обработки входных данных.
- Осуществление сборки пакета под операционную систему, для которой он разрабатывался (для всех перечисленных пакетов это Linux).
- Выполнение профилирования пакета.
- Выполнение сборки пакетов для операционной системы семейства MS Windows (используется Microsoft Windows Server 2008).
- Использование существующих или разработка собственных реализаций наиболее трудоемких участков кода, выявленных на этапе профилирования, с использованием технологии CUDA для выполнения на GPU [8].

- Использование других техник оптимизации (ручная оптимизация кода, компиляторная оптимизация и т.д.)

Для задачи Password Recovery выполняется собственная реализация, использующая технологию CUDA.

**Результаты исследований.** Была успешно выполнена сборка всех пакетов под операционную систему Microsoft Windows Server 2008 (с использованием cygwin и компилятора gcc, либо cygwin и компиляторов PGI, либо посредством Visual Studio), проведено профилирование пакетов, частично использована сторонняя CUDA-реализация в пакете NAMD, выполнена собственная реализация алгоритмов шифрования паролей (реализация MD5 использует графический процессор, реализация blowfish – нет).

До начала соревнований (15 ноября 2010 г.) планируется обеспечить использование графических процессоров пакетами NAMD и WRF, а также полностью выполнить CUDA-реализацию задачи восстановления паролей.

Работа выполнена в лаборатории Intel-ННГУ «Информационные технологии» (ITLab).

## Список литературы

1. SC10. – URL: http://sc10.supercomputing.org.

2. SC10 – Student Cluster Competition. – URL: http://sc10.supercomputing.org/?pg=studentcluster.html.

3. Supercomputing at 1/10-th the Cost. – URL: http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_M2050_M2070_Apr10_LowRes.pdf, 2010.

4. Evan Scannapieco, Marcus Brüggen. Simulating Supersonic Turbulence in Galaxy Outflows. – URL: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzYzEwc2NjfGd4OjMyOTdmYjY4MjRhMjgyMTQ, 2009.

5. The Weather Research and Forecasting Model Website. – URL: http://www.wrf-model.org/index.php]

6. NAMD – Scalable Molecular Dynamics. – URL: http://www.ks.uiuc.edu/Research/namd.

7. «The HPC Challenge (HPCC) Benchmark Suite» / P. Luszczek [et al.] SC06 Conference Tutorial, IEEE, Tampa, Florida, November 12, 2006.

8. CUDA Zone. – URL: http://www.nvidia.ru/object/cuda_home_new_ru.html.

**[1]И.И. Левин, [2]А.И. Дордопуло, [1]В.А. Гудков**

[1]Научно-исследовательский институт многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета, г. Таганрог

[2]Южный научный центр РАН, г. Ростов-на-Дону

## ПРИНЦИПЫ РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ COLAMO

Реконфигурируемые вычислительные системы демонстрируют высокую реальную производительность при решении расчетоемких сильносвязанных задач различных классов. Вместе с тем сдерживающим фактором для широкого распространения реконфигурируемых вычислительных систем (РВС) являются сложность и трудоемкость их программирования, а также отсутствие удобных высокоуровневых средств разработки приложений.

Для программирования РВС наибольшее распространение получили языки HDL-группы, позволяющие описать структуру и функционирование вычислительной системы, а также моделировать обмен данными между блоками системы. Наиболее популярным языком этой группы является язык VHDL, обладающий высоким функциональным уровнем абстракции. Однако громоздкое текстовое описание устройств на VHDL обусловило популярность и широкое использование систем автоматизированного проектирования в виде специализированных графических редакторов, таких как Xilinx ISE, Altera MaxPlus или OrCad. Такой подход к программированию РВС требует участия в программировании системы как специалиста-схемотехника,

создающего конфигурацию вычислительной системы с учетом особенностей ее архитектуры и элементной базы, так и прикладного программиста, создающего параллельную программу, описывающую потоки данных в созданной схемотехником виртуальной вычислительной структуре.

Естественное желание программировать РВС на языке высокого уровня привело к созданию ряда процедурных языков, таких как ImpulseC, Mitrion-C, Catapult C, Handel-C и др. Для языков этой группы характерен привычный для большинства программистов персональных ЭВМ синтаксис языка С, а различия проявляются в семантических особенностях вызова и использования специализированных параллельных операторов. При этом для описания параллельных процессов в этих языках используется изначально последовательный язык С, семантика которого ориентирована на взаимодействие последовательных процессов, что не позволяет в полной мере использовать все возможности РВС. Это приводит к семантическому разрыву между исходным информационным графом алгоритма задачи, его описанием на языке высокого уровня и созданной транслятором схемотехнической реализацией, что выражается в существенном снижении эффективности прикладной программы. Как правило, созданные с помощью указанных языков приложения имеют в 3–5 раз более низкую производительность по сравнению с их аналогами, разработанными на более низких уровнях абстракции.

Поэтому разработка новых предложений в области средств программирования РВС, сочетающих удобство описания информационных графов задач на языке высокого уровня с эффективностью и возможностями языка VHDL, представляется актуальной научной задачей, решение которой позволит расширить традиционные области применения РВС.

**Принципы программирования на языке высокого уровня COLAMO.** В Научно-исследовательском институте многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета более 20 лет

развивается оригинальное научное направление по созданию реконфигурируемых вычислительных систем с динамически программируемой архитектурой различной конфигурации и их программного обеспечения. Программирование созданных образцов осуществляется на языке программирования высокого уровня COLAMO [1].

Язык COLAMO предназначен для описания реализации и создания в архитектуре РВС специализированной вычислительной структуры на основе принципов структурно-процедурной организации вычислений [2], которая предполагает последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов. Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации, в каждой структуре соответствующих потоков данных. Для представления такой организации вычислений в языке используется понятие «кадр» [1,2], являющийся неразрывной совокупностью вычислительной структуры фрагмента задачи и множества операций чтения-записи входных и результирующих потоков данных.

Более строго, кадром является программно-неделимый компонент, представляющий собой совокупность операторов, которые реализуются в виде арифметико-логических команд и команд чтения/записи, выполняемых на различных функциональных устройствах, соединенных между собой в соответствии с информационной структурой алгоритма.

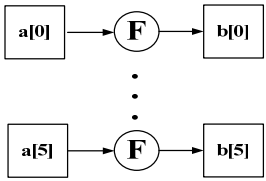
В языке COLAMO отсутствуют явные формы описания параллелизма, а распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов, что характерно для языков потока данных. Для обращения к данным используются два основных метода доступа: параллельный доступ (задаваемый типом Vector) и последо-

вательный доступ (задаваемый типом Stream). Степень параллелизма определяется по минимальному значению параметра распараллеливания. Для типа доступа Stream степень параллелизма равна 1, а для типа доступа Vector определяется наименьшим значением векторной составляющей каждого массива, участвующего в вычислениях. Для параллельного типа доступа возможна одновременная обработка всех размерностей массивов, заданных типом Vector, при этом повышается аппаратный ресурс на обработку, но снижается время обработки.

Многомерные массивы данных могут иметь множество измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно. Если вычисления связывают между собой массивы с различным типом доступа, то заданная пользователем вычислительная структура может оказаться несбалансированной, что приведет к затрате дополнительного оборудования и снизит скорость обработки потоков данных.

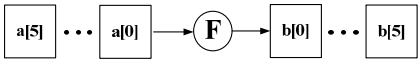
На рисунке представлены примеры простейших программ на языке COLAMO, отличающихся только типом доступа к массиву, и графы синтезируемых вычислительных структур. Данный пример показывает, что одна и та же программа в зависимости от типа доступа к данным при описании массива может быть реализована как параллельно, так и последовательно, причем синтез соответствующей описанию вычислительной структуры осуществляет транслятор.

```
Var a,b: Array Integer [10 : Vector] Mem;
Var i : Number;
Cadr FVector;
   For i := 0 to 5 do
      b[i] :=F(a[i]);
EndCadr;
```




*а*

```
Var a,b : Array Integer [10 : Stream] Mem;
Var i : Number;
Cadr FStream;
   For i := 0 to 5 do
      b[i] :=F(a[i]);
EndCadr;
```




*б*

Рис. Параллельная (*а*) и последовательная (*б*) обработка массивов

Изменение типа доступа позволяет достаточно просто управлять как степенью распараллеливания вычислений на уровне описания структур данных, так и скоростью обработки и занимаемым ресурсом, что позволяет программисту описывать различные виды параллелизма в достаточно сжатом виде.

Помимо типа доступа для переменной в языке COLAMO определен также и тип ее хранения: мемориальный (Mem), регистровый (Reg) и коммутационный (Com).

Мемориальной переменной называется величина, хранящаяся в ячейке распределенной памяти и, следовательно, сохраняющая свое значение до очередного переприсваивания. Для мемориальной переменной возможно одновременное выполнение только одного процесса. Поэтому в семантике языка COLAMO для мемориальной переменной в теле кадра действуют правило однократного присваивания и правило единственной подстановки. Правило однократного присваивания указывает на то, что мемориальная переменная в кадре изменяет свое значение только один раз. Правило единственной подстановки определяет, что переменная в выражении может использоваться только для одного процесса: чтения или записи.

Для описания связей между элементами информационного графа задачи в языке COLAMO предназначена коммутационная переменная. Поскольку коммутационная переменная описывает информационные связи, она не требует никакого вычислительного аппаратного ресурса для своей реализации. Доступ к значению коммутационной переменной после выполнения кадра невозможен. Коммутационная переменная необходима транслятору для указания информационных зависимостей при построении вычислительной структуры задачи. На коммутационную переменную, как и на мемориальную переменную, действует правило однократного присваивания, но не действует правило единственной подстановки. Использование коммутационных переменных позволяет легко разветвлять и дублировать потоки данных, но не позволяет реализовать рекурсию.

Для организации рекурсии в языке COLAMO используется регистровая переменная, которая представляет собой регистр

на аппаратном уровне и используется для хранения промежуточных данных, полученных в процессе вычислений. Единственным ограничением для регистровой переменной в теле кадра является правило однократного присваивания.

Для управления порядком выполнения операций и реализации ветвлений в языке COLAMO предусмотрен оператор условного перехода IF, который может быть внутренним или внешним по отношению к кадру. Внутренний по отношению к кадру оператор условного перехода, расположенный в теле кадра, реализуется на структурном уровне: обе ветви условия выполняются параллельно, а для выбора результата используется мультиплексор. Реализации внешнего оператора условного перехода осуществляются процедурной составляющей путем вызова того или иного кадра в зависимости от условия.

**Расширения языка высокого уровня COLAMO.** Программирование РВС на уровне логических ячеек ПЛИС (LUT), когда программист использует специализированные аппаратно-реализованные блоки, соответствующие арифметико-логическим операциям программы, и создает связи между ними в соответствии с информационным графом задачи, позволяет создать более компактные аппаратные решения для конкретной задачи, учитывающие специфику проблемной области, необходимый формат представления информации, а также требования к производительности задачи.

Для объявления постоянной вычислительной структуры при программировании на уровне ячеек ПЛИС в стандарте языка COLAMO используется конструкция LET, которая описывает функционально законченный неизменяемый подграф в информационном графе задачи. При этом программа может содержать единственное описание вычислительной структуры LET, вызов которой обязательно (по умолчанию) осуществляется в теле каждого кадра программы. В этом случае кадры отличаются друг от друга информационными потоками, подключенными на входы и выходы конструкции LET.

Переключение информационных потоков при вызове конструкции LET в кадре параллельной программы может осуществ-

ляться двумя основными способами: сдвоенным контроллером распределенной памяти (КРП), для автоматического формирования которого транслятором необходимо выполнение ряда условий, или с помощью дополнительных коммутаторов, реализация которых требует как затрат оборудования, так и предварительной настройки коммутаторов перед выполнением каждого кадра.

Современная архитектура ПЛИС позволяет использовать внутреннюю память ПЛИС, которая обладает достаточной емкостью, высокой скоростью доступа, а также возможностью организации одновременного доступа к памяти нескольких независимых процессов чтения и записи по разным портам. Поддержка работы с внутренней памятью ПЛИС в языке COLAMO осуществляется с помощью нового типа хранения переменных – InterMem **<N>**, где N – количество доступных портов при работе с внутренней памятью ПЛИС.

Для решения задач символьной обработки в языке COLAMO предусмотрена поддержка битовых переменных и соответствующего им набора операций и функций. Обработка битовых переменных, которые объявляются ключевым словом Bit, в языке осуществляется вычислительными операциями, выполняющими логические функции (and, or, xor, not), и логическими поразрядными операциями (операции сдвига влево и вправо, операции циклического сдвига влево и вправо и др.). Языковые средства позволяют получить доступ к любому биту в слове высокоуровневыми конструкциями и адресовать биты операторами индексного доступа к элементам массива. Также в языке поддерживаются групповые операции («срезы») для произвольного измерения массива любого типа, позволяющие сократить запись.

Для сокращения объема программы и повышения ее читабельности предназначены макросы обработки битовых переменных, позволяющие выполнять битовые операции не на уровне элементов битовых массивов, а на уровне 32-разрядных или 64-разрядных слов.

Язык программирования реконфигурируемых вычислительных систем COLAMO с учетом описанных расширений позволяет рационально использовать ресурсы ПЛИС при про-

граммировании РВС на уровне логических ячеек и обеспечивает пользователя набором необходимых средств для быстрой разработки эффективных параллельных программ.

## Список литературы


1. Каляев А.В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.

2. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А. Каляев [и др.]. – изд. 2-е, перераб., доп. / под общ. ред. И.А. Каляева. – Ростов н/Д: Изд-во ЮНЦ РАН, 2009. – 344 с.

3. Левин И.И. Многопроцессорная система с программированием архитектуры на нескольких уровнях // Методы и средства обработки информации: тр. Первой Всерос. науч. конф. – М.: Изд-во МГУ, 2003. – С. 111–118.

4. Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой / А.И. Дордопуло [и др.] // Высокопроизводительные вычислительные системы: материалы IV Междунар. науч. молодежн. школы. – Таганрог: Изд-во ТТИ ЮФУ, 2007. – С. 68–74.

5. Левин И.И. Ресурсонезависимое параллельное программирование // Искусственный интеллект. – Донецк: Наука і освіта, 2002. – № 3. – С. 277–285.



**С.Ю. Лесовой, А.В. Панюков**

Южно-Уральский государственый университет, г. Челябинск


## ПРИМЕНЕНИЕ МАССИВНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ РЕАЛИЗАЦИИ ОСНОВНЫХ ОПЕРАЦИЙ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ

Существенным недостатком вычислений с помощью аппаратных типов данных с плавающей точкой является погрешность. Данную проблему позволяет решить использование производных дробно-рациональных типов данных, в которых числа

представляются точно в виде дроби из векторов переменной длины [1, 2]. Однако операции над такими типами данных на порядки увеличивают вычислительную нагрузку и, следовательно, оттесняют круг требовательных к ресурсам задач в ранг невыполнимых (в отношении времени вычислений).

Один из возможных способов расширения возможностей заключается в использовании мощности графического ядра обычного компьютера. Данная технология разрабатывается ведущим производителем видеокарт Nvidia и на сегодняшний день любая современная видеокарта этой фирмы поддерживает технологию CUDA, позволяющую средствами языка C++ задействовать мощность графического ядра. Стоит отметить, что количество параллельных потоков, выполняемых на графическом ядре, достигает 128 и более в топовых моделях, в то время как средний компьютер имеет всего 2 ядра CPU, а топовый – 4

При реализации параллельных алгоритмов простейших арифметических операций над длинными числами узким местом оказывается ограничение скорости пересылки данных между оперативной памятью и памятью видеокарты, которая определяется пропускной способностью шины видеокарты. Следовательно, необходимо свести к минимуму такие пересылки и при решении практических задач исходные данные помещать в память видеокарты, а в оперативную память пересылать решение задачи.

В основе реализации машинной арифметики лежат операции сложения и умножения натуральных чисел. Рассмотрим их подробнее.

**Алгоритм А.** (Сложение неотрицательных целых чисел).

По заданным неотрицательным целым числам $(a_{n-1},...,a_0)_r$ и $(b_{m-1},...,b_0)_r$ по основанию $r$ алгоритм формирует их сумму $(c_n,...,c_0)_r$, здесь $c_n$ – разряд переноса, всегда равный 0 или 1. Далее будем считать $n \geq m$, что не уменьшает общности изложения

**А1** [Начальная установка]. Выделить *n* процессов (тредов).

**А2.** Каждому треду $i$ =1,2,…,*n* выполнить следующе ниже шаги.

**А3. С**читать *i*-й разряд из числа *a* во временную переменную $t[i] = a_i$.

```
        mov   t[i], a_i
```

**A4** [Суммирование]. К временной переменной *t* добавить соответствующий разряд числа *b* и полученную сумму занести в результирующую переменную $c[i] = t[i] + b_i$. Флаг переноса при этом занести в переменную *f*, экземпляр которой создается для каждого треда. Значение этой переменной принимает значения 0 либо 1.

```
        mov   f[i], 0
        add   c[i], b_i
        adc   f[i], 0
```

**A5.** [Распространение переносов].

```
loop:   inc   i
        mov   f[i], 0
        add   c[i], f[i-1]
        adc   f[i],0
        jc    loop
```

Конец описания **Алгоритма А.**

Оценим трудоемкость **Алгоритма А** и возможный выигрыш от его использования. Как известно, числа представляются в компьютерной памяти в двоичном виде, поэтому разрядом длинного числа разумно считать $2^r$, где *r* – количество используемых двоичных разрядов. Для большинства современных компьютеров оптимальными являются 32-битные числа, но уже постепенно распространяются 64-битные системы.

На шаге **A3** происходит присвоение значений разрядов числа соответствующим переменным *t* каждого из *n* тредов. Теоретически можно обойтись без этой временной переменной и выполнить сразу же сложение, однако на практике большинство процессоров не поддерживают операции типа память–память, а только регистр–память.

Шаг **A4** выполняется полностью параллельно, а значит, время его выполнения будет минимальным (время выполнения элементарного сложения обозначим как *A*) при условии наличия достаточного количества тредов.

На шаге **A5** все зависит от значений исходных слагаемых, время выполнения будет варьироваться в пределах $[0, nA]$.

Таким образом, время выполнения алгоритма будет зависеть от исходных слагаемых и находиться в пределах $[A,(n+1)A]$. Предположим, что цифры входных числа являются случайными и распределены по равномерному закону.

Введем обозначения $p$ – вероятность возникновения переполнения при суммировании в одном из разрядов, $q$ – вероятность получения после суммирования числа равного значению $2^r-1$. Тогда

$$p=\frac{1}{2^r}\cdot\frac{1}{2^r}\left(0+1+2+...+(2^r-1)\right)=\frac{1}{2^{2r}}\cdot\frac{(2^r-1)2^r}{2}=\frac{2^r-1}{2\cdot 2^r}=\frac{1}{2}-\frac{1}{2\cdot 2^r};$$

$$q=\frac{1}{2^r}.$$

Рассчитаем теперь вероятности выполнения за различное время $P_i$.

$$P_1=\left(1-\frac{1}{2}+\frac{1}{2\cdot 2^r}\right)^m=(1-p)^m;$$

$$P_2=1-P_1-\sum_{i=3}^{m+1}P_i=1-P_1-mpq=1-(1-p)^m-mpq;$$

$$P_3=m\left(\frac{1}{2}-\frac{1}{2\cdot 2^r}\right)\frac{1}{2^r}-\sum_{i=4}^{m+1}P_i=mpq-\sum_{i=4}^{m+1}P_i=mpq-(m-1)pq^2;$$

$$P_4=(m-1)\left(\frac{1}{2}-\frac{1}{2\cdot 2^r}\right)\left(\frac{1}{2^r}\right)^2-\sum_{i=5}^{m+1}P_i=(m-1)pq^2-\sum_{i=5}^{m+1}P_i=(m-1)pq^2-(m-2)pq^3;$$

$$P_m=2\left(\frac{1}{2}-\frac{1}{2\cdot 2^r}\right)\left(\frac{1}{2^r}\right)^{m-1}-P_{m+1}=2pq^{m-1}-P_{m+1}=2pq^{m-1}-pq^m;$$

$$P_{m+1}=\left(\frac{1}{2}-\frac{1}{2\cdot 2^r}\right)\left(\frac{1}{2^r}\right)^m=pq^m.$$

Несложно заметить, что уже при $r=32, p\approx\frac{1}{2}, q\approx 0$, поэтому вероятности $P_3,P_4...,P_{m+1}$ можно считать нулевыми. Вероятность $P_1$ будет зависеть от длины меньшего слагаемого $m$ и уже при $m=10, P_1\approx 0$. Таким образом, вероятность $P_2\to 1$ при увеличении $m$, $r$. Следовательно, время выполнения алгоритма будет

составлять примерно $2A$. Если сравнивать это с последовательным выполнением суммирования, то производительность алгоритма выше в $m$ раз, то есть не зависит от длины складываемых чисел, а только от их значений и ограничивается лишь мощностью графического ядра (количество физических процессоров).

Операция вычитания осуществляется по тем же принципам, что и сложение и за такое же время, поэтому не будет здесь рассматриваться.

**Алгоритм М.** По заданному целому числу по основанию $r$ $(a_{n-1},...,a_0)_r$ и одноразрядному числу $u$ алгоритм находит их произведение $(c_n,...,c_0)_r$.

**М1** [Начальная установка]. Выделить *n* процессов(тредов).

**М2.** Каждому треду *i* =1,2,…,*n* выполнить следующе ниже шаги

**М3** [Начальная установка] Выделить $n$ тредов. В каждом потоке считать соответствующий разряд из числа *a* в временную переменную $t[i] = a_i$.

```
mov   t[i], a_i
```

**М4** [Умножение]. Умножить временную переменную *t* на число $u$ и полученную сумму занести в определенный разряд дополнительной переменной *d*. Идея состоит в том, чтобы сформировать 2 независимых слагаемых $d_1, d_2$, которые представляют собой результаты произведений четных и нечетных разрядов соответственно и находятся в двух частях переменной *d*.

$$d\left[(i\%2)\left(\frac{n}{2}\right)+i\right] = t[i]\cdot u,$$

результат этого произведения будет занимать уже 2 разряда. После выполнения этого шага будет сформирован массив *d* длиной *2n* разрядов.

```
mov   eax, u
mul   t[i]
mov   d[(i%2)*(n/2)+i], eax
mov   d[(i%2)*(n/2)+i+1], edx
```

**M5** [Формирование результата]. Для получения результата необходимо сложить обе части промежуточной переменной *d*, что можно выполнить с помощью алгоритма А, разобранного ранее.

Конец описания **Алгоритма М.**

Оценим производительность данной операции. Шаг **M3**, как и в случае алгоритма сложения, будет зависеть от времени выполнения операции присвоения (инициализации) и выполняется за пренебрежительно малое время.

Шаг **M4** выполняется полностью параллельно, время его выполнения будет как у операции элементарного умножения (число на число), обозначим его *M*.

Шаг **M5,** как и сам алгоритм сложения, выполняется за время, зависящее от входных данных, которое лежит в пределах $[A,(n+1)A]$. На основе анализа алгоритма сложения следует вывод, что среднее время выполнения этого шага будет *2A*.

Таким образом, умножение длинного числа на одноразрядное значение можно выполнить за время, сопоставимое с $M+2A$.

**Алгоритм ММ** (Умножение натуральных чисел). По заданным числам по основанию $r$ $(a_{n-1},...,a_0)_r$ и $(b_{m-1},...,b_0)_r$ этот алгоритм формирует их произведение $(c_{2n-1},...,c_0)_r$, считаем при этом, что $n \geq m$.

**ММ1** [Распределение вычислений]. Для параллелизации процесса воспользуемся предыдущим алгоритмом умножения длинного числа на одноразрядное. Разбить операцию умножения на независимые блоки, которые могут выполняться параллельно. Каждый такой блок будет представлять собой произведение числа $(a_{n-1},...,a_0)_r$ на соответствующий разряд $b_i$ и его можно выполнить, используя предыдущий алгоритм. При этом будет сформирован массив промежуточных значений $z_0,...,z_{m-1}$, представляющих собой результат умножения на одноразрядное число.

**ММ2** [Общее суммирование]. Для быстрого суммирования можно воспользоваться алгоритмом парного суммирования. Для этого понадобится $\frac{m}{2}$ блоков, которые осуществят

суммирование по алгоритму А промежуточных значений $z_0 + z_1, z_2 + z_3, ..., z_{m-1} + z_m$. В случае нечетности $m$ последний блок будет просто копировать значение $z_m$.

**ММ3** [Проверка окончания]. Шаг ММ2, представляющий собой операцию парного суммирования, повторять до тех пор, пока результатом не станет одно результирующее число, которое и будет ответом. Несложно заметить, что количество повторов шага М2 будет составлять $\log_2 m$.

Конец описания **Алгоритма ММ.**

Оценим производительность **Алгоритма ММ**. Шаг ММ1 выполняется параллельно на разных блоках и при условии достаточного количества блоков время выполнения будет такое же, как и у алгоритма умножения длинного числа на одноразрядное $M + 2A$.

Шаг ММ2 и ММ3 связаны логически, и время выполнения будет определяться количеством повторов шага М2 $\log_2 m$. Сам шаг ММ2 представляет собой сложение длинных чисел и выполняется по алгоритму А. Следовательно, время выполнения составит $2A \cdot \log_2 m$.

Таким образом, умножение длинных чисел можно выполнить за время порядка $M + 2A + 2A \cdot \log_2 m$.

Операцию деления можно реализовать через уже разобранную операцию умножения с помощью модифицированного алгоритма Ньютона [3, с. 304–425].

## Список литературы

1. А.В. Панюков, М.И. Германенко, В.В. Горбик. Библиотека классов «Exact Computational» / Свидетельство о государственной регистрации программы для ЭВМ № 2009612777 от 29 мая 2009г. // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Офиц. бюл. Российского агентства по патентам и товарным знакам № 3 2009. С. 251.

2. GNU Multiple Precision Arithmetic Library. – URL: http://gmplib.org/, 2010.

3. Кнут Д. Искусство программирования. Т. 2. Получисленные методы. – М.: Вильямс, 2000. – 832 с.

**И.Н. Лозгачев, А.В. Сенин**

Нижегородский государственный университет им. Н.И. Лобачевского

## ИСПОЛЬЗОВАНИЕ СИМУЛЯТОРА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ GSSIM В ЗАДАЧЕ ОПРЕДЕЛЕНИЯ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ПЛАНИРОВАНИЯ

Высокопроизводительные вычисления на системах кластерного тема – важный инструмент для решения широкого круга задач в науке и промышленности. Согласно списку Top 500 самых производительных суперкомпьютеров [1] с каждым годом наблюдается рост числа узлов и процессорных элементов в системах – лидерах списка. Пропорционально числу вычислительных элементов и числу пользовательских задач растет сложность управления такими системами. Ключевая роль отводится здесь алгоритмам планирования, распределяющим задания по вычислительным ресурсам. Можно показать, что правильно выбранный алгоритм планирования может существенным образом повысить эффективность использования кластера [2].

Задача планирования на кластерных системах относится к числу NP –трудных, что делает невозможным нахождение точного решения. На практике используются те или иные приближенные алгоритмы, для многих из которых неизвестны теоретические обоснования их эффективности. Поэтому в большинстве современных работ, посвященным алгоритмам планирования на кластерах, исследователи сравнивают новые алгоритмы с существующими, проводя демонстрационные запуски на выбранных конфигурациях и потоках задач. Вполне естественно, что использовать для этих целей реальные высокопроизводительные системы обычно не представляется

возможным, так как доступ к крупнейшим кластерам сильно ограничен. Возможная альтернатива – использовать специальные симуляторы высокопроизводительных систем, позволяющих проверить интересующие аспекты работы алгоритма на произвольных системах без использования дорогостоящего оборудования.

В данной работе рассматривается вопрос интеграции системы управления вычислительной инфраструктурой «Метакластер» [3], разрабатываемой в ННГУ им Н.И. Лобачевского, с симулятором Grid для исследования реализованных алгоритмов планирования.

**Требования к симулятору.** Алгоритмы планирования системы управления «Метакластер» позволяют распределять вычисления на многокластерной системе, учитывая при этом как неоднородности между отдельными кластерами, так и возможную неоднородность внутри каждого кластера. Для проведения адекватных исследований необходимо, чтобы инфраструктура симулятора позволяла сымитировать максимально широкое число сценариев использования системы. Это позволяет выдвинуть следующие требования к симулятору:

- возможность гибкого задания вычислительных ресурсов кластера с возможной неоднородностью по таким параметрам, как: скорость процессора, объем оперативной памяти, число вычислительных элементов (ядер);
- возможность задания параметров сети передачи данных: пропускная способность, латентность, топология;
- возможность симуляции нескольких кластеров одновременно.

Сравнение эффективности работы алгоритмов планирования проводится на основании метрик, подсчитанных по результатам планирования. Существуют различные метрики, усредненные значения которых используются для оценки эффективности компонентов планирования. К наиболее распространенным критериям относятся:

- время ожидания запуска (waiting time);

• время отклика (response time) – сумма времени ожидания запуска ( $T_w$ ) и времени работы ( $T_r$ ) задачи $T_w + T_r$ ;

• время замедления (slowdown) – отношение времени отклика ко времени работы задачи $\frac{T_w + T_r}{T_r}$ ;

• предельное время замедления (bounded slowdown) – метрика, значение которой определяется следующей формулой $\max\left(\frac{T_w + T_r}{\max(T_r, \tau)}, 1\right)$, где $\tau$ – некоторая граничная константа времени работы задачи (как правило, устанавливается равной 10 с).

К требованиям, предъявляемым к симулятору, также относятся возможность подсчета указанных метрик и предоставление средств визуализации расписания (например, диаграмма Ганта).

**Выбор симулятора.** На данный момент существует ряд проектов по созданию симуляторов высокопроизводительных вычислений: GridSim, OptorSim, Monarch, ChicSim, SimGrid, MicroGrid, GSSIM. Большинство из них ориентированы на Grid-симуляции и созданы для запуска на операционных системах семейства Unix. Ниже приведена сравнительная характеристика наиболее популярных симуляторов:

| Характеристики симулятора | Симуляторы | | | | | | |
|---|---|---|---|---|---|---|---|
| | Grid-Sim[4] | OptorSim [5] | Monarc [6] | ChicSim [7] | SimGrid [8] | Micro-Grid [9] | GSSIM [10] |
| Репликация данных | + | + | + | + | - | - | + |
| Накладные расходы на операции дискового ввода/вывода | + | - | + | - | - | + | + |
| Планирование задач пользователя | + | - | + | + | + | + | + |
| Резервация ресурсов | + | - | - | - | - | - | + |

Окончание табл.

| Характеристики симулятора | Симуляторы | | | | | | |
|---|---|---|---|---|---|---|---|
| | Grid-Sim[4] | OptorSim [5] | Monarc [6] | ChicSim [7] | SimGrid [8] | Micro-Grid [9] | GSSIM [10] |
| Симуляция на основе Workload архива | + | - | - | + | - | - | + |
| Поддержка различных моделей QoS | + | - | - | - | - | - | + |
| Генерация побочного траффика | + | + | - | - | + | + | + |
| Язык программирования | C | Java | Java | Parsec | C | Java | Java |

Для тестирования разрабатываемых алгоритмов командой разработчиков системы управления «Метакластер» был выбран симулятор GSSIM. В основе сделанного выбора лежит ряд преимуществ симулятора GSSIM: легкость переноса на различные операционные системы, поддержка потоков задач в формате Workload Parallel Archive, резервация ресурсов и др. Ключевым при выборе также стал тот факт, что в отличие от других симуляторов GSSIM изначально проектировался для тестирования именно алгоритмов планирования и содержит ряд специфичных функций, отсутствующих в симуляторах общего назначения: генерация потока задач с заданным распределением параметров, возможности визуализации расписания и другие.

**Особенности Grid-симулятора GSSIM.** В основе любого вычислительного эксперимента лежат параметры вычислительной системы, исследуемый алгоритм планирования и набор задач пользователя. Для описания этой конфигурации GSSIM использует отдельные модули:

- описание вычислительной системы – содержит набор кластеров с характеристиками производительности, параметры коммуникационной среды (учитываются топология, латентность и пропускная способность), набор хранилищ данных, список очередей задач, а также генератор критических ситуаций;

- описание эксперимента – включает в себя набор задач пользователя (содержит требования к ресурсам, время старта и окончания счета, приоритет), планировщик уровня Grid, планировщик уровня кластера, плагин подсчета времени, необходимого для решения конкретной задачи.

В качестве дополнительной возможности реализована поддержка Workload Parallel Archive – базы данных, содержащей историю выполнения вычислительных задач с реальных кластеров и Grid по всему миру. Данная функциональность очень удобна и важна, поскольку решает проблему генерации вычислительных задач и позволяет посмотреть, как будет себя вести исследуемый алгоритм планирования на реальных системах.

После проведения вычислительного эксперимента GSSIM генерирует отчет, содержащий следующую информацию:

- график распределения CPU по задачам пользователя;
- график средней загрузки CPU по каждому кластеру отдельно;
- график выполнения задач;
- график выполненных резерваций ресурсов;
- статистика загрузки сети.

Данная информация может быть использована в дальнейшем для заключения выводов об эффективности того или иного алгоритма планирования.

**Интеграция системы управления «Метакластер» и симулятора GSSIM.** Симулятор GSSIM предоставляет удобную среду для исследования алгоритмов планирования. Однако для использования этих возможностей с алгоритмами «Метакластера» необходимо портировать алгоритмы в среду GSSIM. Существует два решения этой задачи:

- переписывание алгоритмов планирования на язык Java (в настоящее время алгоритмы написаны на C#) и внедрение созданных модулей в симулятор;
- создание некоторого связующего звена между планировщиком и симулятором.

Первое решение неудобно тем, что при модификации алгоритма в системе управления придется переписывать и модули,

внедряемые в симулятор. Поэтому используется второй вариант, а в качестве связующего звена выступает веб-сервис, перенаправляющий запросы от симулятора к планировщику. Выбор веб-сервисов в качестве интерфейса взаимодействия обусловлен в основном соображениями ускорения разработки: C# и Java имеют хорошо развитые средства работы с веб-сервисами.

Основные функции веб-сервиса:

- добавление/удаление вычислительных узлов (AddNode, DeleteNode);
- добавление/удаление задач (AddTask, DeleteTask);
- обработка критических событий с перепланировкой очередей исполнения (ответ на возникшее событие определенного типа).

Для взаимодействия с веб-сервисом реализованы GridSchedulerPlugin и LocalSchedulerPlugin:

- планировщик уровня Grid просто перенаправляет входящие задачи на уровень ниже;
- планировщик уровня кластера подает запросу вебсервису, вызывая методы AddTask или DeleteTask, и получает на выходе очередь задач. Затем ставит их на исполнение.

Для вычисления приблизительного времени работы программы на системе с конкретным набором ресурсов взят стандартный плагин, входящий в состав GSSIM.

Общение между симулятором и веб-планировщиком осуществляется по протоколу SOAP с использованием автоматически сгенерированного WSDL. Такие сущности, как Task или Node, имеют некоторые различия в наборе характеристик у системы управления «Метакластер» и симулятора GSSIM. Для решения этой проблемы написан код по преобразованию типов.

Работа по интеграции алгоритмов планирования системы управления «Метакластер» с симулятором GSSIM еще не закончена. К настоящему времени проработана схема интеграции, написаны основные функции на стороне веб-сервиса и частично на стороне симулятора. Планируется закончить интеграцию и провести тестирование реализованных в «Метакластере» алгоритмов на различных вычислительных системах и потоках задач.

Работа выполняется в лаборатории Intel-ННГУ «Информационные технологии» (ITLab).

**Список литературы**

1. TOP500 Supercomputing Sites [Электронный ресурс]. – URL: http://www.top500.org.

2. Гергель В.П., Кустикова В.Д., Сенин А.В. Интеграция системы управления интегрированной средой высокопроизводительных вычислений «Метакластер» с подсистемой планирования Maui // Вестн. ННГУ. Информационные технологии. – 2010.

3. Гергель В.П., Сенин А.В. Разработка системы управления интегрированной средой высокопроизводительных вычислений «Метакластер» // Вестн. ННГУ. Информационные технологии. – 2010.

4. Официальная страница симулятора GridSim [Электронный ресурс]. – URL: http://www.buyya.com/gridsim.

5. Официальная страница симулятора OptorSim [Электронный ресурс]. – URL: http://edg-wp2.web.cern.ch/edg-wp2/ optimization/optorsim.html.

6. Dobre C., Stratan C. Monarc simulation framework // Politehnica University of Bucharest, 2004.

7. Ranganathan K., Foster I. Decoupling computation and data scheduling in distributed data-intensive applications // University of Chicago.

8. Официальная страница симулятора SimGrid [Электронный ресурс]. – URL: http://simgrid.gforge.inria.fr.

9. The MicroGrid: a scientific tool for modeling computational gridsr / Song H.J. [et al.]. – IEEE. 2000.

10. Официальная страница симулятора GSSIM [Электронный ресурс]. – URL: http://www.gssim.org.

**С.А. Лопаткин, Е.Н. Шелухин**

Томский политехнический университет

# ИСПОЛЬЗОВАНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ КЛАСТЕРНЫХ СИСТЕМ ДЛЯ МОДЕЛИРОВАНИЯ ЭЛЕКТРОВЗРЫВА В ОБЪЕМЕ МАТЕРИАЛА

Известно, что взрыв – крайне быстрое выделение энергии, связанное с внезапным изменением состояния вещества, сопровождаемое деформированием окружающей среды, в которой при этом образуются особого рода возмущения – ударные или взрывные волны, имеет место переход начальной энергии в энергию движения вещества. В том случае, когда исходным видом энергии является электрическая, говорят об электровзрыве [1].

Разряд емкостного накопителя через проводник приводит к мгновенному разрушению последнего с образованием разрядного плазменного канала. Ввод энергии в канал приводит к его расширению, деформации окружающего материала и образованию ударных волн.

Электровзрыв позволяет в достаточно широких пределах регулировать энерговыделение в разрядном канале и тем самым влиять на поле напряжений вокруг канала, определяя картину деформации материала. В связи с этим основной задачей повышения эффективности применения электроразрядных технологий является согласование режимов энерговыделения при разряде с характеристиками генерируемой волны. При этом крайне актуальным является математическое моделирование, позволяющее, не прибегая к экспериментам, проследить динамику развития процесса [2].

Анализ модели электровзрыва сводится к решению значительного числа уравнений для большого количества взаимосвязанных ячеек среды. Использование классических методов решения задач подобного рода, как правило, связано с большими временными затратами. В данном случае очевидным представляется использование распределенных систем вычисления, позволяющих значительно ускорить решение поставленной задачи.

**Математическая модель и программная реализация.** Компьютерная модель электровзрыва реализовывалась с использованием технологии распределенных вычислений MPI [3] (Message Passing Interface) для последующих расчетов на суперкомпьютерном кластере СКИФ ТПУ. Алгоритм программы представляет собой цикл параллельных вычислений с одним главным потоком-мастером.

В основу математической модели электровзрыва положена схема преобразования энергии накопителя в канальной стадии разряда (рис. 1). Энергия накопителя при разряде конденсатора частично теряется в активном сопротивлении разрядного контура и при взрыве проводника, а основная ее часть выделяется в расширяющемся разрядном канале. Последняя составляющая расходуется на образование и нагрев плазмы, а также на работу по ее расширению, т.е. на энергию ударной волны. Энергия волны трансформируется во внутреннюю энергию материала и энергию его движения [2].

Модель электровзрыва включает в себя: уравнения Кирхгофа для разрядного контура и уравнения для взрыва проводника; уравнение энергобаланса плазменного канала и систему уравнений, описывающих импульсное деформирование среды вокруг канала и динамику ударной волны в материале.

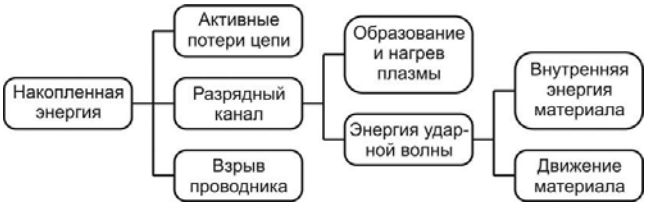



Рис. 1. Схема распределения энергии генератора

Проводник и образующийся на его месте разрядный плазменный канал приближенно могут быть представлены в виде идеальных цилиндров (рис. 2, *а*). На небольших расстояниях от разрядного канала ударная волна также имеет цилиндрическую

форму, а следовательно, ее развитие может быть рассмотрено с применением цилиндрической сетки (рис. 2, *б*), ячейки которой имеют одинаковые начальные параметры (давление, плотность и т.д.).

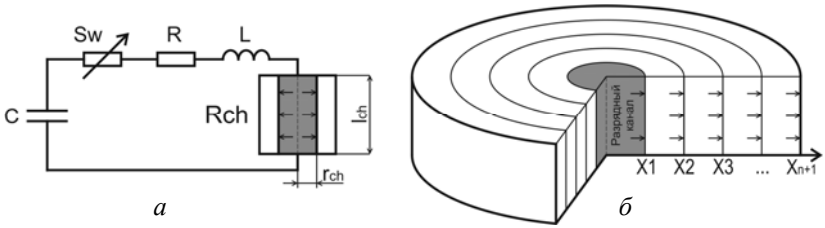



Рис. 2. Разрядный контур (*а*), среда распространения ударных волн (*б*)

На начальной стадии электроразрядного процесса поток-мастер анализирует взрыв проводника в соответствии с известной теорией [4] и задает параметры для последующего моделирования.

В начале каждой итерации основного цикла программы поток-мастер задает шаг моделирования по времени *dt*. Шаг выбирается таким образом, чтобы ударная волна проходила первую ячейку сетки за 5 или более итераций, тем самым обеспечивается стабильность модели:

$$dt = \frac{\Delta X}{hCw},\ h = 5, 6, \ldots . \tag{1}$$

Далее путем решения уравнений Кирхгофа производится расчет переходных процессов в разрядном контуре за время, равное *dt*, и определяется введенная в разрядный канал энергия, а также оцениваются активные потери энергии в контуре:

$$L\frac{di}{dt} + (R + R_{ch})i = U_C,\ \frac{dU_C}{dt} = -\frac{i}{C}; \tag{2}$$

$$W_{ch}(t) = \int_0^t i^2(t) R_{ch}(t)\, dt,\ W_R(t) = \int_0^t i^2(t) R dt. \tag{3}$$

Сопротивление разрядного канала, образующегося после взрыва проводника, находится через интеграл действия тока по соотношению Ромпе–Вайцеля:

$$R_{ch}(t) = Al_{ch} \Big/ \sqrt{\int_0^t i^2(t)\,dt}\,, \qquad (4)$$

где A – искровая постоянная материала.

Связь электротехнической части процесса с волновой динамикой в среде устанавливается с помощью уравнения энергобаланса

$$dW_{ch} = P_{ch} dV_{ch} + \frac{d(P_{ch} V_{ch})}{(\gamma - 1)}\,. \qquad (5)$$

Первое слагаемое описывает приращение работы, совершаемой расширяющимся каналом при изменении его объема $V_{ch}$ под действием давления внутри канала $P_{ch}$. Второе – энергия плазмы, расширяющейся в адиабатическом приближении с показателем $\gamma$ [2].

На данном этапе поток-мастер обрабатывает имеющийся массив данных и разбивает его на части. После этого происходит рассылка по потокам [5]. Поскольку каждая ячейка сетки неразрывно связана с соседними, используется ленточное разделение массива (рис. 3).

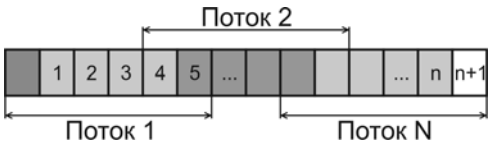



Рис. 3. Ленточное разбиение массива

После получения данных каждый поток начинает цикл расчета деформаций среды. Этот этап включает в себя решение уравнений законов сохранения импульса, массы и энергии в лагранжевых координатах для цилиндрической симметрии [6]:

$$\frac{dU}{dt} = -\frac{\Delta P}{\rho \Delta X}\,,\ U = U + dU\,,\ \frac{dX}{dt} = U\,,\ X = X + dX\,; \qquad (6)$$

$$\rho = \rho_0 \left( \frac{\Delta X_0}{\Delta X} \right), \; P = B \left( \left( \frac{\rho}{\rho_0} \right)^n - 1 \right). \qquad (7)$$

При определении скорости движения стенок ячеек для стабилизации модели при росте плотности вводится искусственная вязкость в соответствии с зависимостью

$$Q_n^t = 2a\Delta U^2 \Big/ \left( \frac{1}{\rho_{t-1}} + \frac{1}{\rho_{t-2}} \right). \qquad (8)$$

По завершении расчетов поток-мастер собирает все части массива воедино и производит его анализ. После этого основной цикл переходит на новую итерацию.

В процессе роста разрядного канала близлежащие к нему ячейки сильно сжимаются, при этом модель становится крайне нестабильной, а также уменьшается шаг по времени, что приводит к сильному возрастанию времени моделирования. Для решения этой проблемы при сжатии ячеек до критически малого размера соседние ячейки попарно объединяются.

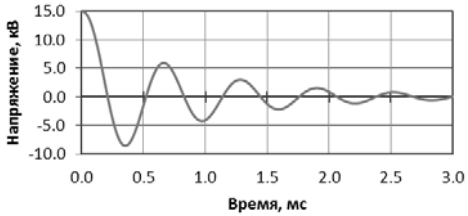
Энергия волны $W_w$, состоящая из суммы внутренней $W_p$ и кинетической $W_k$ энергий, рассчитывается по соотношениям:

$$W_w = W_k + W_p; \qquad (9)$$

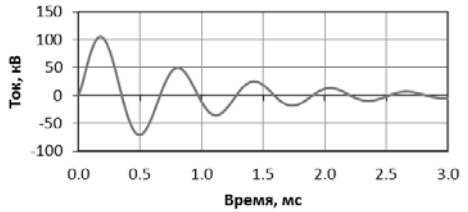
$$W_k = 2\pi l_{ch} \int_{r_{ch}}^{Xm} e(X)\rho(X)\,dX, \; W_p = \pi l_{ch} \int_{r_{ch}}^{Xw} u^2(X)\rho(X)\,dX. \qquad (10)$$

Для проверки адекватности модели было проведено моделирование разрядного процесса в воде, в качестве накопителя использовался генератор импульсных токов (15 кВ, 96 мкФ). Полученные результаты представлены на рис. 4 и хорошо совпадают с данными оценочного расчета [1].

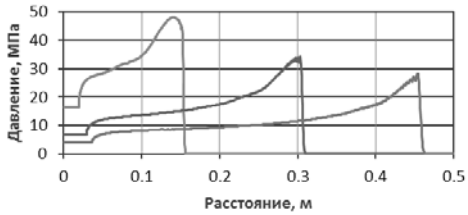
**Оптимизация расчетного процесса.** В связи с возникшей потребностью моделирования перед разработчиками встал ряд проблем, не имеющих прямого аналитического решения, в частности необходимость выбора оптимальных параметров установки. Возникшие проблемы могут быть достаточно просто решены путем проведения многократного анализа исходной модели электрического взрыва при различных входных параметрах, т.е. решения большого числа задач.



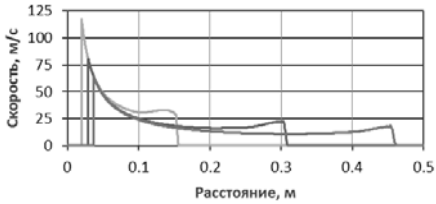

*1*




*2*




*3*




*4*

Рис. 4. Результаты моделирования: *1*, *2* – осциллограммы напряжения и тока; *3*, *4* – давление и скорость ударной волны в воде через 0,1 мс, 0,2 мс, 0,3 мс

Наилучшие условия для решения одной задачи ($2^{16}$ ячеек) определялись путем автоматического подбора. Первоначально задача решалась с использованием 40 процессов, затем их число изменялось на 5 и сравнивалось время, затраченное на решение

задачи, для каждого из вариантов. В конечном счете было установлено, что максимальное число процессов, дающих прирост производительности для решаемой задачи, – 52.

Известно, что с увеличением количества задействованных в расчете процессов эффективность использования аппаратных ресурсов снижается [3]. Очевидно, что наиболее разумным решением данной проблемы является сокращение числа использованных процессов для одной задачи с одновременным запуском нескольких параллельных задач (рис. 5).

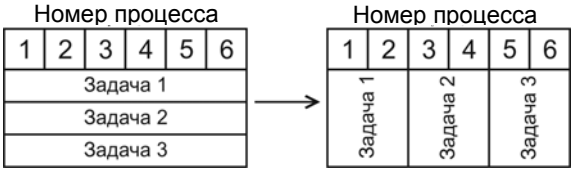



Рис. 5. Переход от последовательных задач к параллельным

Для оценки прироста производительности измерялось время обработки задачи для 50 процессов и той же задачи для 10. Время расчета составило 150.31 с и 416.19 с соответственно. Очевидно, производительность при решении 5 задач параллельно возросла в 1,8 раза.

$$5t_{50} / t_{10} = 751{,}55 / 416{,}19 \approx 1{,}8. \qquad (11)$$

В процессе исследований была разработана и протестирована модель электровзрыва в объеме материала. Программная реализация модели адаптирована для расчетов на высокопроизводительных кластерных системах, использующих технологию распределенных вычислений MPI.

В процессе выполнения работы был определено максимальное число процессов, дающих прирост производительности для решения одной задачи, а также обоснована оптимальная схема распределения аппаратных ресурсов, заключающаяся в решении нескольких задач одновременно.

Результаты моделирования имеют хорошее совпадение с экспериментальными. Таким образом, вычислительные возможности кластерных систем позволяют решать сложные ре-

сурсоемкие задачи, что дает исследователю возможность четко представлять характер поведения ударной волны в материале при различных режимах и условиях.

### Список литературы

1. Усов А.Ф., Семкин Б.В., Зоновьев Н.Т. Переходные процессы в установках электроимпульсной технологии. – Л.: Наука, 1987.

2. Буркин В.В., Кузнецова Н.С., Лопатин В.В. Волновая динамика электровзрыва в твердых диэлектриках. // Журнал технической физики. – 2009. – Т. 79. Вып. 5. – С. 42–48.

3. Антонов А.С. Параллельное программирование с использованием технологии MPI: учеб. пособие. – М.: Изд-во МГУ, 2004.

4. Бурцев В.А., Калинин Н.В., Лучинский А.В. Электрический взрыв проводников и его применение в электрофизических установках. – М.: Энергоатомиздат, 1990.

5. Воеводин В.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ, 2006.

6. Уилкинс М.Л. Расчет упруго-пластических течений. Вычислительные методы в гидродинамике. – М.: Мир, 1967.

**Е.В. Лысь, В.В. Лисица, Г.В. Решетова, В.А. Чеверда**

Институт нефтегазовой геологии и геофизики
им. А.А. Трофимука СО РАН, Новосибирск

## ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ КОНЕЧНО-РАЗНОСТНОГО МОДЕЛИРОВАНИЯ ДАННЫХ АКУСТИЧЕСКОГО КАРОТАЖА

Проведение акустического каротажа в районах со сложным геологическим строением, повсеместное использование наклонных и горизонтальных скважин ведёт к необходимости существенного углубления понимания процессов формирования и распространения сейсмоакустических полей в трёхмерно-неоднородных средах с учётом таких особенностей горных пород, как анизотропия и поглощение. К сожалению, аналитическое описание такие волновые

поля допускают только в простейших постановках, весьма далёких от реальности. В связи с этим возникает необходимость в алгоритмах, позволяющих выполнять их полномасштабное математическое моделирование, что стало возможным с появлением высокопроизводительных вычислительных систем с параллельной архитектурой.

В работе представлен метод конечно-разностного моделирования акустического каротажа для системы уравнений упругости, где тензор упругих модулей не имеет квазидиагональной структуры, что позволяет моделировать задачи с произвольной анизотропией и TTI. При таком виде тензора упругих модулей появляется необходимость использовать специальные схемы (схема на повернутых сетках, схема Лебедева), поскольку сдвинутая сетка Верьё [9] не позволяет аппроксимировать решение такой системы. Математическая постановка задачи упругости рассматривалась в цилиндрической системе координат, поскольку в этом случае удаётся избежать проблемы аппроксимации на криволинейных границах скважины и окружающей её формации.


Рис. 1. Элементарная ячейка схемы Лебедева в цилиндрической системе координат. Компоненты тензора напряжений определены в узлах, отмеченных сферами, а компоненты скоростей смещений в узлах, помеченных звёздочками

В качестве граничных условий был выбран идеально согласованный слой PML (perfectly match layer).

**Постановка задачи.** Рассматривалась система уравнений упругости

$$\rho \frac{\partial u}{\partial t} = \nabla \sigma, \quad \frac{\partial \varepsilon}{\partial t} = \frac{1}{2}(\nabla u + \nabla u^T), \quad \sigma = C_\varepsilon,$$

где ρ – плотность, $u$ – скорость смещений, σ – тензор напряжений, ε – тензор деформаций, $C$ – тензор упругих постоянных.

В выбранной системе координат тензор $C$ представляется квадратной симметричной (6×6) матрицей. Для изотропного случая тензор $C$ имеет простой квазидиагональный вид. В этом случае для конечно-разностной аппроксимации системы возможно использовать схему Вирьё [9]. В анизотропных средах тензор $C$ теряет квазидиагональный вид, и использование схемы Вирьё приводит к необходимости интерполяции переменных и, как следствие, потере точности конечно-разностного решения. В работе [1] была предложена схема Лебедева для конечно-разностной аппроксимации системы уравнений упругости с произвольным видом тензора $C$, позволяющая аппроксимировать задачу без каких-либо пространственных интерполяций. Схема Лебедева является обобщением схемы Вирьё для более широкого класса задач и вследствие этого обладает сходными дисперсионными и спектральными свойствами. На рис. 1 представлена элементарная ячейка схемы Лебедева в цилиндрической системе координат.

Рис. 2. Типичное скоростное строение среды при акустическом каротаже

**Геометрия расчетной области и строение сетки.** Как видно из рис. 2, наиболее контрастные границы типичного скоростного строения среды при такого рода исследованиях – это границы между зондом и флюидом в скважине и стенка скважины (граница флюид-формация). Эти интерфейсы имеют цилиндрическую геометрию. Чтобы избежать численного рассеяния на этих интерфейсах, все рассмотрения проводятся

в цилиндрической системе координат, координатные линии которой совпадают с наиболее контрастными границами среды.

При фиксированных шагах сетки объем элементарной ячейки будет увеличиваться при удалении от оси $r = 0$, т.е. ухудшаются дисперсионные свойства схемы и налагается более строгое ограничение на шаг по времени. Чтобы устранить эти явления, проводится измельчение сетки в азимутальном направлении (рис. 3), в результате которого разница объемов любых двух ячеек сетки (покрывающей некоторый геометрический объем) не превосходит некоторого наперед заданного числа. На интерфейсах между сетками (неизмельченной и измельченной) возникает необходимость проводить интерполяцию переменных задачи в точках, в которых определены операторы конечно-разностной схемы. Как было показано В.И. Костиным с соавторами [5], интерполяция, основанная на преобразовании Фурье, имеет экспоненциальный порядок сходимости благодаря $2\pi$-периодичности интерполируемых функций.

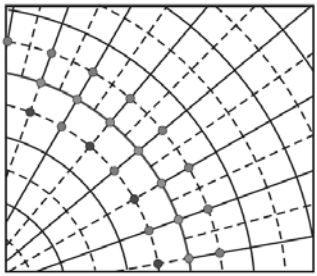


Рис. 3. Схема распределения узлов (переменных) на интерфейсе между двумя сетками

**Декомпозиция области вычислений.** Программная реализация данного алгоритма была выполнена на языке C++ с использованием библиотеки MPI (message parsing interface). В силу специфики расчетной области (вытянутый в вертикальном направлении цилиндр) применялась одномерная декомпозиция расчетной области, т.е. модель разрезалась на $N$ подобластей по координате $Z$, где $N$ – число процессоров параллельной вычислительной системы, задействованных при расчете конкретной задачи. Обмен между соседними процессорами осуществлялся средствами библиотеки MPI. Для повышения эффективности работы параллельного алгоритма применялись неблокирующие операторы обмена (Isend, Irecv), позволяющие осуществлять обмен данными в фоновом режиме, т.е. процесс пересылки/приема данных выполняется одновременно с расчетом внутри области. Вся область расчетов делится на ус-

ловно зависимые блоки, каждому такому блоку соответствует некоторый класс (объект, в котором реализована конечно-разностная схема, предназначенная для работы в фиксированной части расчетной области).

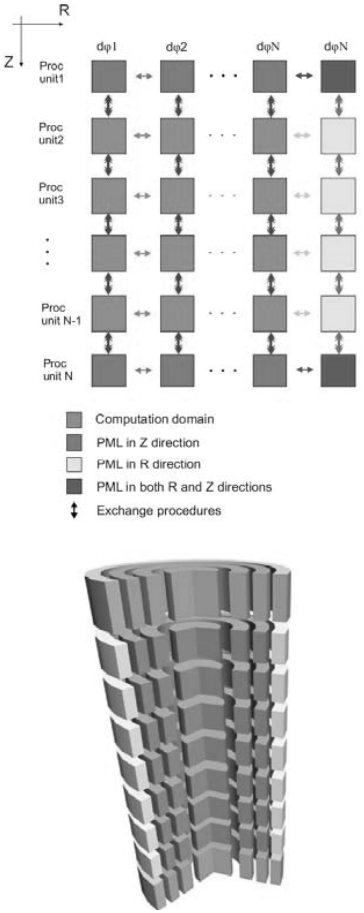



Рис. 4. Схематическое представление декомпозиции расчетной области

Геометрическая форма всех подобластей – есть кольцо прямоугольного сечения, кроме областей, содержащих точки с координатой $r = 0$ (рис. 4, первый столбец), их форма – цилиндр.

На рис. 4 (внизу) представлена трехмерная, более наглядная диаграмма декомпозиции расчетной области. При такой организации вычислений необходимо определить два типа процедур обмена данными:1) обмен между соседними процессорами (см. рис. 4) – двойные стрелки; 2) обмен между различными классами на одном процессоре – одинарные стрелки. Реализация и очередность выполнения процедур обмена в большой степени влияют на быстродействие алгоритма в целом.

## Список литературы


1. Asvadurov S., Druskin V. and Moskow S. 2007 Optimal Grids for Anisotropic Problems. Electronic Transactions on Numerical Analysis (ETNA) 26, 55–81.

2. Biot M. A. 1952 Propagation of elastic waves in a cylindrical bore containing fluid // Journal of Applied Physics 23, 997–1005.

3. Chen Y.-H. and Chew W.C. 1998. A three dimensional finite-difference code for the modelling of sonic logging tool // Jour. Acoust. Soc. Am., 103 (2), 702–712.

4. Kessler D., Kosloff D. 1991 Elastic wave propagation using cylindrical coordinates // Geophysics, 56 (12), 2080–2089.

5. 3d finite-difference synthetic acoustic logging in cylindrical coordinates / V.I. Kostin [et al.] // Geopthysical Prospecting, 2008. Electronical publication Doi: 10.1111/j.1365- 2478.2008.00643.x.

6. Krauklis P.V., Krauklis L.A. 1976 On compressional wave spectrum in a well with cemented casing string // Dynamic wave propagation theory problems, Vol. XVII. P. 156–164, Nauka (in Russian).

7. Lisitsa V. and Vishnevsky D. Lebedev type scheme for the numerical simulation of wave propagation in 3D anisotropic elasticity // Geophysical Prospecting, 2009, doi: 10.1111/j.1365-2478.2009.00862.x.

8. Pissarenko D., Reshetova G.V. and Tcheverda V.A. 2009 3D finite-difference synthetic acoustic logging in cylindrical coordinates // Geophysical Prospecting 57, 367–377.

9. Virieux J. 1986. P-SV wave propagation in heterogeneous media: Velocity – stress finite difference method // Geophysics 51(4), 889–901.

**С.И. Мальковский, В.В. Пересветов**

Вычислительный центр Дальневосточного отделения РАН, г. Хабаровск

## ГЕТЕРОГЕННЫЙ ВЫЧИСЛИТЕЛЬНЫЙ КЛАСТЕР ВЦ ДВО РАН

В докладе описывается гетерогенный вычислительный кластер ВЦ ДВО РАН, состоящий из трех разнородных групп вычислительных узлов. Рассматриваются полученные экспериментальные результаты исследования производительности сегментов этой вычислительной системы.

Гетерогенный вычислительный кластер отличается от однородного вычислительного кластера тем, что в его состав входят вычислительные узлы разных типов. Целесообразность создания гетерогенных вычислительных кластеров обусловливается приобретением в разные периоды времени различного вычислительного оборудования, которое необходимо эффективно использовать. В таком кластере можно выделять обладающие наиболее подходящими ресурсами узлы для решения определенных классов задач. При этом отсутствует необходимость поддержки нескольких специализированных вычислительных кластерных систем, так как все группы узлов в гетерогенном кластере обслуживаются одним управляющим узлом.

Для эффективного использования гетерогенной кластерной системы необходимо знать производительность всех сегментов кластера при решении основных типов задач. Также необходимо учитывать особенности работы на разнородных узлах установленного системного программного обеспечения (ПО) и различных параллельных технологий. Поэтому нами были проведены экспериментальные исследования производительности всех сегментов гетерогенного кластера, результаты которых приведены в работах [1–4]. Результаты тестирования могут быть также сопоставлены с данными [5], где экспериментально исследовалась производительность вычислительного кластера, построенного с использованием неспециализированного оборудования. В этой работе изложена также методика тестирования,

в которой проводится исследование производительности вычислительного оборудования на основе усреднения результатов многократных запусков тестовых задач с нахождением среднеквадратических отклонений.

Все узлы описываемого гетерогенного вычислительного кластера работают под управлением операционной системы Linux CentOS. Для диспетчеризации заданий используется PBS Torque с планировщиком Maui. Мониторинг осуществляется пакетом Ganglia. Установлено ПО Intel Cluster Toolkit. В качестве коммуникационной среды параллельных вычислений используется библиотека MPI в реализации Intel. Также на кластере доступна библиотека Intel MKL, компиляторы Intel C/C++/Fortran и GNU C/C++/Fortran.

Гетерогенный вычислительный кластер ВЦ ДВО РАН в настоящее время объединяет с помощью коммуникационной сети Gigabit Ethernet три типа узлов: Sun Blade X6440 Server, Sun Blade X6250 Server и HP ProLiant DL360 G5.

К первому типу относятся два сервера Sun Blade X6440 Server, каждый из которых оснащен четырьмя шестиядерными процессорами AMD Opteron Istanbul 8431 (2,4 GHz) и 96 GB оперативной памяти. Эти узлы могут использоваться по отдельности в качестве мультипроцессорных (24 вычислительных ядра) систем с общей памятью. Производительность одного такого сервера в тесте Linpack составила 182 GFlops или 79 % от пиковой. Экспериментальные исследования производительности узлов Sun Blade X6440 Server показали, что на каждом из них можно эффективно решать без взаимодействия с остальными узлами задачи широкого спектра в пределах до 24 процессов MPI или нитей OpenMP [1–2]. Хорошую масштабируемость на них показывают даже те задачи, эффективность решения которых напрямую зависит от скорости обмена данными между вычислительными ядрами.

На данных узлах компиляторы Intel icc и ifort превосходят компиляторы GNU лишь в отдельных случаях некоторых вычислительно-затратных программ с небольшим объемом меж-

процессорных взаимодействий. В остальных случаях различие в производительности программ с использованием протестированных компиляторов (Intel и GNU) несущественно. Применение технологии OpenMP в большинстве случаев приводит к получению производительности на уровне MPI. Более высокую эффективность технология OpenMP показывает лишь тогда, когда сильно загружается среда передачи данных.

В дополнение к стандартным тестам для узлов Sun Blade X6440 было проведено в [2] исследование зависимости производительности системы от привязки процессов к различным процессорным ядрам. Оно показало, что в случае когда процессы совместно работают с большими объемами данных, не помещающимися в кэш-память, наиболее верным будет разнесение их на разные физические процессоры. В случае передачи сообщений небольшого объема максимальная производительность будет достигаться при их запуске на одном физическом процессоре. В коллективных операциях MPI наблюдается большая зависимость производительности системы передачи данных от способа привязки процессов к различным процессорным ядрам, чем в двухточечных. Подробно результаты исследования производительности этого типа узлов описаны в [1–2].

Ко второму типу вычислительных узлов относятся пять серверов Sun Blade X6250 Server. Каждый из них оснащен двумя четырехъядерными процессорами Intel Xeon E5450 (3 GHz) и 16 GB оперативной памяти. Пиковая производительность одного такого узла составляет 80 GFlops. В тесте Linpack для пяти узлов достигнут уровень производительности 305 GFlops.

При тестировании этой группы узлов в качестве сети передачи данных использовалась сеть, основанная на технологии Gigabit Ethernet. Такие исследования позволили оценить производительность данной коммуникационной среды в современных вычислительных кластерах. Они показали, что для достигнутого к настоящему времени уровня производительности процессоров, эффективности кэш-пямяти и скорости обменов с оперативной

памятью коммуникационная сеть Gigabit Ethernet должна быть признана устаревшей для использования в таких системах [3]. Только в ограниченном числе задач она может быть использована без существенной потери производительности. В случае использования технологии Gigabit Ethernet при программировании нужно избегать коллективных операций, осуществлять пересылки данных равномернее, отдавать предпочтение асинхронным двухточечным операциям. Использование компиляторов Intel icc и ifort на узлах этого типа приводит иногда к очень большому отрыву от других, и они в целом являются более эффективными. Компиляторы GNU иногда незначительно превосходят другие компиляторы. Компиляторы LLVM для C/C++ и Fortran показали эффективность на уровне компиляторов GNU. В пределах одного узла технология OpenMP показывает, как правило, производительность выше, чем MPI. Подробно результаты исследования производительности второго типа узлов описаны в [3].

Третий тип узлов представлен восемью серверами HP ProLiant DL360 G5, построенными на базе двух двухъядерных процессоров Intel Xeon 5060 (3,2 GHz). Каждый из таких узлов оснащен 4 GB оперативной памяти и имеет пиковую производительность, равную 25,6 GFlops. Подробно результаты исследования производительности этого типа узлов описаны в [4].

В табл. 1 представлены результаты теста IMB для трех групп узлов гетерогенного вычислительного кластера с различным числом процессов $n$. Размер пересылаемых сообщений 8 байт. Эти результаты показывают, что задержки при передаче данных минимальны при запуске взаимодействующих процессов в пределах одного узла вычислительного кластера. Если же в качестве коммуникационной среды используется сеть Gigabit Ethernet – задержки сильно возрастают. Эти и нижеприведенные результаты были получены на наборе тестов NPB и IMB, собранных при помощи компилятора Intel версии 10.1. В качестве библиотеки MPI использовалась Intel MPI.

Таблица 1

**Результаты теста IMB**

| Тип сообщений | Sun Blade X6440 | | Sun Blade X6250 | | HP ProLiant DL360 | |
|---|---|---|---|---|---|---|
| | $n = 4$ | $n = 24$ | $n = 4$ | $n = 40$ | $n = 4$ | $n = 16$ |
| Sendrecv | 1,5 | 2,0 | 1,1 | 14 | 1,6 | 63 |
| Bcast | 1,9 | 2,9 | 1,4 | 33 | 2,1 | 149 |
| Allreduce | 2,8 | 3,8 | 2,6 | 183 | 3,7 | 145 |

В табл. 2 представлены усредненные результаты теста NPB EP (класс C) для трех групп узлов, которые позволяют оценить максимальную производительность (в MFlops) в отсутствие заметных межпроцессорных взаимодействий. Значения производительности в расчете на одно вычислительное ядро (результаты делятся на число задействованных ядер процессоров) практически одинаковы для любого числа ядер процессоров.

Таблица 2

**Производительность в тесте EP**

| Sun Blade X6440 | Sun Blade X6250 | HP ProLiant DL360 G5 |
|---|---|---|
| 31 | 51 | 23 |

В табл. 3 представлены аналогичные результаты в расчете на одно процессорное ядро для тестов NPB LU (класс C), в котором интенсивно используется коммуникационная среда для передачи сообщений. Взаимодействие параллельных процессов осуществляется большим числом синхронных передач сообщений MPI_Send небольшой длины. Среднеквадратические отклонения результатов лежат в диапазоне 0,3–4 %. Показатели разброса результатов возрастают с уменьшением класса сложности.

В табл. 4 представлены результаты теста NPB IS (класс C). В нем осуществляется параллельная сортировка большого массива целых чисел. Передача сообщений между процессами реализуется с помощью операций MPI_Alltoall и MPI_Allreduce. Из

всех тестов, входящих в состав NPB, он является самым требовательным к производительности среды передачи данных. Из результатов видно, что этот тест эффективно работает только тогда, когда все вычислительные процессы запущены в пределах одного узла кластера (Sun Blade X6440). В остальных случаях, когда для передачи данных используется Gigabit Ethernet, производительность падает до неприемлемо низкого уровня. Среднеквадратические отклонения экспериментальных результатов находились в пределах 1 % для класса сложности C.

Таблица 3

**Производительность в тесте LU**

| Число процессов | Sun Blade X6440 | Sun Blade X6250 | HP ProLiant DL360 G5 |
|---|---|---|---|
| 4 | 1180 | 511 | 470 |
| 16 | 904 | 765 | 650 |

Полученные экспериментальные результаты показывают, что в вычислительных кластерах производительность процессоров, на которых построены узлы, является важным, но не единственным фактором, определяющим возможности вычислительной системы. Другим важным фактором является производительность коммуникационной среды. Приемлемым вариантом построения гетерогенных вычислительных кластеров при использовании устаревшей технологии Gigabit Ethernet является применение вычислительных узлов с большим числом ядер. На каждом из таких узлов можно эффективно решать без взаимодействия с остальными узлами задачи широкого спектра в пределах до 24 процессов MPI или нитей OpenMP, как в случае описанных здесь узлов Sun Blade X6440.

Таблица 4

**Производительность в тесте IS**

| Число процессов | Sun Blade X6440 | Sun Blade X6250 | HP ProLiant DL360 G5 |
|---|---|---|---|
| 4 | 54 | 28 | 21 |
| 16 | 27 | 5,2 | 4,5 |

Во многих случаях технология MPI показала производительность на уровне технологии OpenMP. Применение компиляторов GNU обычно позволяет получить скорость вычислений на уровне компиляторов Intel, поэтому их можно рекомендовать для использования в научной и образовательной сферах деятельности.

**Список литературы**

1. Мальковский С.И., Пересветов В.В. Исследование производительности четырехпроцессорных узлов в составе вычислительного кластера // Суперкомпьютеры: вычислительные и информационные технологии: материалы междунар. науч.-практ. конф., Хабаровск, 30 июня – 2 июля 2010 г. – Хабаровск: Изд-во Тихоокеанского гос. ун-та, 2010. – С. 77–84.

2. Мальковский С.И., Пересветов В.В. Зависимость производительности передачи сообщений MPI от порядка распределения процессов в четырехпроцессорном узле вычислительного кластера // Суперкомпьютеры: вычислительные и информационные технологии: материалы междунар. науч.-практ. конф., Хабаровск, 30 июня – 2 июля 2010 г. – Хабаровск: Изд-во Тихоокеанского гос. ун-та, 2010. – С. 85–92.

3. Мальковский С.И., Пересветов В.В. Оценка производительности вычислительного кластера на четырехъядерных процессорах // Информационные и коммуникационные технологии в образовании и научной деятельности: материалы межрегион. науч.-практ. конф., Хабаровск, 21–23 сентября 2009 г. – Хабаровск: Изд-во Тихоокеанского гос. ун-та, 2009. – С. 261–268.

4. Щерба С.И., Пересветов В.В. Сравнительный анализ эффективности программного обеспечения для вычислительных кластеров // Информационные и коммуникационные технологии в образовании и научной деятельности: материалы межрегион. науч.-практ. конф. (г. Хабаровск, 21–23 мая 2008 г.). – Хабаровск: Изд-во Тихоокеанского гос. ун-та, 2008. – С. 363–369.

5. Пересветов В.В., Сапронов А.Ю., Тарасов А.Г. Вычислительный кластер бездисковых рабочих станций. Препринт № 83. Вычислительный центр ДВО РАН. – Хабаровск, 2005. – 50 с.

**С.В. Марьин, С.В. Ковальчук, А.В. Ларченко**

НИИ наукоемких компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики

# ИНТЕЛЛЕКТУАЛЬНАЯ ПЛАТФОРМА УПРАВЛЕНИЯ КОМПОЗИТНЫМИ ПРИЛОЖЕНИЯМИ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ

В настоящее время существует обширный класс задач, решение которых требует использования высокопроизводительных вычислительных ресурсов. При этом в последнее время существенным интересом со стороны исследователей характеризуются задачи, обладающие высокой структурной сложностью. Примерами такого рода задач могут служить задачи моделирования сложных систем [1]. Под сложной системой в данном контексте подразумевается система, а) состоящая из большого числа компонентов; б) характеризующаяся наличием дальних связей между компонентами; в) обладающая многомасштабной изменчивостью. Характерными примерами такой системы могут служить климатическая система «океан–атмосфера» с учетом мелкомасштабной, синоптической, сезонной и межгодовой изменчивости, или эпидемиологическая система (от масштабов отдельного вируса до всемирной популяции носителей). Рациональным подходом к решению подобных задач является композитный подход, использующий существующие программные решения в качестве процедурных элементов в составе более сложного алгоритма. Во многих прикладных областях эта задача облегчается за счет возможности использования объемной информационной, алгоритмической и программной базы, разработанной ранее. Тем не менее зачастую такая база характеризуется существенной разнородностью (вследствие независимой разработки отдельных решений), неполнотой (например, в результате неполной реализации цикла разработки программного обеспечения), а зачастую неоднозначностью (в случае существования альтернативных методов, алгоритмов и/или их программных

реализаций). Следствием этой проблемы является сложность конструирования композитных программных решений на основе существующей базы программных компонентов. Другой насущной проблемой является структурное усложнение и рост неоднородности доступных сегодняшнему разработчику программно-аппаратных платформ для высокопроизводительных вычислений: наблюдается рост популярности систем с программно-аппаратной архитектурой, отличной от традиционных суперкомпьютерных решений (ПЛИС, GPGPU); объединение вычислительных ресурсов (в том числе посредством сети Интернет) в единые вычислительные комплексы, зачастую отличающиеся сложной динамической топологией. Таким образом, актуальной становится разработка средств унификации доступа к разнородным вычислительным ресурсам и автоматизации процесса управления вычислениями в такой неоднородной среде. Наиболее остро эта проблема встает в случае организации работы с динамическими композитными приложениями: возникает необходимость отображения композитного приложения на множество вычислительных сервисов в составе используемой вычислительной инфраструктуры таким образом, чтобы построенное отображение характеризовалось наилучшей производительностью в ходе решения поставленной задачи. В рамках решения этой задачи необходимо обеспечение не только технологической, но и информационной интеграции используемых вычислительных ресурсов. Такая интеграция, в свою очередь, позволяет обеспечить: а) непрерывность потока исполнения в рамках композитного приложения при условии автоматизации промежуточных процессов (конвертирования и анализа данных, управления коммуникационной средой и т.п.); б) информационную поддержку пользователя в процессе конструирования и исполнения композитного приложения. При этом наилучшим решением является то, которое максимально изолирует пользователя от технических сложностей вычислительного эксперимента. Идеальная система взаимодействует с пользователем на языке соответствующей предметной области, автоматизируя все технические процессы на основании имеющихся требований.

В данной работе рассматривается структура и базовые принципы работы платформы, за счет привлечения экспертных знаний, обеспечивающей: а) интеллектуальную поддержку процесса конструирования композитных приложений; б) построение оптимального отображения построенного композитного приложения на доступные вычислительные ресурсы; в) реализацию выполнения приложения на выбранных ресурсах с максимальной изоляцией пользователя от технических особенностей этого процесса.

**Использование базовой концепции iPSE.** Разрабатываемая платформа реализуется на базе концепции iPSE (Intelligent PSE) [3], обеспечивающей должный уровень информационной и технологической интеграции композитных приложений, унифицированный доступ к разнородным высокопроизводительным инфраструктурам, а также интеллектуальную и инструментальную поддержку пользователя в процессе организации и проведения вычислений. Концепция iPSE строится на развитии логики PSE и определяет принципы построения сред для моделирования сложных систем в форме интеллектуальной оболочки управления параллельными вычислительными процессами в распределенной иерархической среде, включающей в себя вычислительные системы различной архитектуры. Такой подход расширяет достоинства традиционных PSE за счет обеспечения эффективного параллельного исполнения композитных приложений в силу того, что использует для управления параллельными вычислениями симбиотические знания об особенностях предметной области и специфике вычислительного процесса:

1. В рамках iPSE формализуются не только методы и вычислительные алгоритмы, но и экспертные знания об организации процесса решения задачи в рассматриваемой предметной области средствами компьютерного моделирования.

2. iPSE предоставляет единый интерфейс взаимодействия для разнородных предметно ориентированных программных модулей и компонентов.

3. iPSE изначально ориентирована на поддержку высокопроизводительных вычислений на базе платформ с различной

архитектурой. При этом централизованное управление эффективностью выполнения сценария позволяет избежать конфликтных ситуаций при разделении ресурсов между различными вычислительными модулями и разными пользователями.

Программная система, построенная на базе концепции iPSE, может быть отнесена к классу интеллектуальных систем, обладающих сложной распределенной структурой (структурная сложность), характеризующихся широким спектром решаемых задач (функциональная сложность), а также учитывающих особенности работы в условиях неопределенности (информационная сложность).

**Использование экспертных знаний.** Управление разнородными вычислительными ресурсами, оптимальное построение композитных приложений для решения поставленной задачи, организация информационной и технологической поддержки пользователя в значительной степени базируются на экспертных знаниях. Для эффективной реализации указанных процессов требуется использование соответствующих механизмов формализации и автоматического использования знаний. В рамках концепции iPSE можно выделить два базовых класса знаний, отличающихся своей ролью в процессе работы программного комплекса: предметно-зависимые знания о соответствующей области, к которой принадлежит решаемая задача, и технологические знания о характеристиках и особенностях использования программных и аппаратных средств.

На сегодняшний день существует множество способов описания знаний: правила формальной логики, фреймовые структуры, семантические сети и пр. В рамках концепции iPSE предлагается использовать онтологический подход [4], ориентированный на детальную формализацию избранных областей знаний. Указанный подход в настоящее время отличается высокой популярностью в качестве инструмента формализации описательных знаний о предметных областях. Кроме того, онтологическая структура традиционно представляет собой динамически расширяемое описание, включающее в свой состав

множество отдельных онтологий, что позволяет в рамках предлагаемого подхода осуществить динамическую модификацию доступной базы знаний.

Формируя структуру онтологии, необходимо учитывать ряд особенностей, как проистекающих из требований, определяемых объектом исследования (композитными высокопроизводительными приложениями с динамической компонентной архитектурой на основе прикладных сервисов), так и определяемых необходимостью универсализации структуры онтологии в предположении о возможности ее адаптации для задач произвольной предметной области. При этом производится декомпозиция онтологии на несколько базовых уровней, различающихся по степени абстракции, с которой рассматривают анализируемое композитное приложение:

1. *Уровень метаописания* представляет собой структурированные знания: а) о процессе функционирования всей системы, целиком описанной на всех уровнях; б) о многоуровневых моделях производительности, учитывающих все уровни абстракции, способах их синтеза и методах использования; в) о способах анализа протекающих в суперкомпьютерном приложении процессов.

2. *Абстрактное описание.* На данном уровне композитное приложение и его компоненты рассматриваются как средства реализации определенных методов компьютерного моделирования, обладающих соответствующим набором параметров.

3. *Реализация.* Промежуточное описание, связывающее вычислительный модуль, доступный посредством сервиса с индивидами классов, находящихся на уровне абстрактного описания. Совокупность индивидов данного уровня определяет структуру доступных вычислительных модулей, а также допустимые процедуры работы с ними.

4. *Ресурс.* Наиболее низкоуровневое описание сервисов, предоставляющее информацию о конкретных вычислительных сервисах, их характеристиках, платформах их исполнения. Этот уровень описания независим от предметной области, поскольку

основное множество его классов соответствует понятиям из области информационных технологий.

**Работа с композитными приложениями.** Реализуя работу с композитными приложениями в рамках концепции iPSE, интеллектуальная платформа обеспечивает поддержку основных этапов построения композитного приложения. На начальном этапе пользователь описывает постановку задачи в терминах своей предметной области, создавая таким образом семантическое описание композитного приложения, или *мета-workflow* (MWF), без указания условий ее выполнения в распределенной вычислительной среде. Второй этап проектирования состоит в переходе к описанию в терминах *абстрактного workflow* (AWF). AWF отличается от MWF тем, что оперирует конкретными методами, допускающими реализацию с использованием доступных вычислительных сервисов. Тем не менее в рамках AWF эти методы не ассоциируются с какими-либо вычислительными сервисами и определяются преимущественно в терминах предметной области. Выбранные методы подбираются посредством поиска на основе пользовательских критериев (требований к функциональным характеристикам, ресурсоемкости, точности и пр.) Третьим этапом проектирования является детализация workflow до уровня *конкретного workflow* (CWF) с целью оптимизации структуры и характеристик композитного приложения по времени выполнения. Базовыми процедурами построения CWF являются: выбор конкретных вычислительных сервисов, выбор эффективных способов распараллеливания по каждому из сервисов (внутренний параллелизм), выбор эффективных способов распараллеливания на уровне композитного приложения, динамическая модификация структуры CWF (в том числе миграция задач) и пр. Описание приложения в форме CWF эквивалентно алгоритмической записи, что позволяет автоматически генерировать сценарий исполнения, который может быть использован непосредственно для запуска приложения под управлением интеллектуальной платформы.

**Управление вычислительной инфраструктурой.** Одной из ключевых подсистем разрабатываемой платформы является

подсистема управления распределенными платформами, доступными программному комплексу. Основными ее задачами являются организация эффективного унифицированного доступа к разнородным программно-аппаратным платформам и множеству вычислительных сервисов, выполняющихся на этих платформах, а также планирование процесса вычислений. Архитектура данной подсистемы и ее связь с остальными логическими подсистемами в рамках концепции iPSE представлена на рисунке.

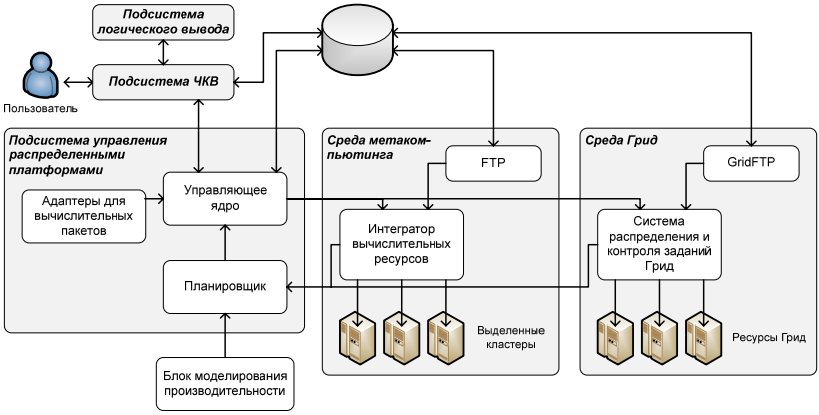



Рис. Функционирование подсистемы управления распределенными платформами в рамках концепции iPSE

Управляющее ядро подсистемы управления распределенными платформами получает описание решаемой пользователем задачи в виде абстрактного WF, контролирует исполнение элементов, входящих в его состав, и предоставляет возможности мониторинга процесса вычислений. Кроме того, управляющее ядро при помощи адаптеров вычислительных пакетов формирует корректные входные файлы для выбранных пакетов на основе данных, предоставленных пользователем.

Планировщик получает информацию о вычислительных ресурсах, их загрузке, выполняющихся задачах, оценивает посредством моделей производительности время выполнения за-

дания на каждом вычислительном ресурсе, и с использованием возможностей блока моделирования производительности принимает решение, какое задание на каком ресурсе запустить.

В соответствии с решением, выработанным планировщиком, управляющее ядро осуществляет непосредственный запуск задач, а также контроль их исполнения и постобработку. В том числе реализуются следующие функции:

- возможность формирования CWF на основе полученного AWF и дополнительных параметров предметной области;
- автоматическое определение свободных вычислительных ресурсов оптимальных для указанной задачи (т.е. позволяющих минимизировать время счета);
- расчет приблизительного времени вычисления для указанной задачи;
- формирование параметров запуска (команды запуска, скриптов, входных файлов) для указанной задачи;
- запуск решения задачи на выбранных ресурсах;
- предоставление внутрисистемного интерфейса для получения сообщения о завершении процесса вычислений (успешном или неуспешном);
- выдача информации о состоянии задачи.

В результате проведенных работ была разработана интеллектуальная платформа управления композитными приложениями в распределенных вычислительных средах, обеспечивающая: а) эффективное использования экспертных знаний в процессе управления распределенными вычислениями с целью повышения производительности реализуемых композитных приложений; б) поддержку процесса поэтапного конструирования структуры композитного приложения в соответствии с требованиям пользователя и критериями эффективности, предъявляемыми к высокопроизводительным приложениям; в) интеграцию разнородных вычислительных ресурсов и их эффективное применение в ходе выполнения композитных приложений. Разработанная платформа была реализована в рамках концепции iPSE, что позволяет рассматривать ее как технологическую базу для практической реализации проектов на основе данной концепции.

В ходе работ были проведены испытания разработанной платформы на базе вычислительных инфраструктур, работающих в рамках моделей метакомпьютинга и Грид. Результаты испытаний показали возможность эффективного использования разработанной платформы в качестве надстройки над вычислительными ресурсами, доступными в рамках данных моделей.

### Список литературы

1. Boccara N. Modeling Complex Systems. // Springer New York, 2004. — 397 p.

2. Rice J.R., Boisvert R. F. From Scientific Software Libraries to Problem-Solving Environments // IEEE Computational Science & Engineering. – 1996. – Vol. 3. No. 3. – P. 44–53.

3. Бухановский А.В., Ковальчук С.В., Марьин С.В. Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации // Изв. высших учебных заведений. Приборостроение. – 2009. – № 10. – С. 5–24.

4. What Are Ontologies, and Why Do We Need Them? / B. Chandrasekaran, J.R. Josephson, V.R. Benjamins // IEEE Intelligent Systems. 1999. – Vol. 14, No. 1. – P. 20–26.

**[1]А.Г. Масич, [1]Г.Ф. Масич, [1]Р.А. Степанов, [2]В.А. Щапов**

[1] Институт механики сплошных сред УрО РАН, г. Пермь
[2] Пермский государственный технический университет

## СКОРОСТНОЙ I/O-КАНАЛ СУПЕРВЫЧИСЛИТЕЛЯ И ПРОТОКОЛ ДЛЯ ОБМЕНА ИНТЕНСИВНЫМ ПОТОКОМ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

### Общие сведения об эксперименте

Экспериментальная установка (ЭУ) использует метод PIV (Particle Image Velocimetry) [4] – оптический метод измерения полей скорости жидкости или газа в выбранном сечении потока. Принцип метода: импульсный лазер создает тонкий световой

нож и освещает мелкие частицы, взвешенные в исследуемом потоке. Положения частиц в момент двух последовательных вспышек лазера регистрируются на два кадра цифровой камеры. Скорость вихревого потока определяется расчетом перемещения, которое совершают частицы за время между вспышками лазера. Измерительная часть установки генерирует поток данных 1-10-100 Гбит/с в зависимости от разрешения, частоты и режимов работ камер (моно/стерео/томография).

Области применения PIV: гидро- и аэродинамика лабораторных течений, физическое моделирование технологических процессов в энергетике, химической промышленности, диагностика обтекания реальных и модельных объектов в авиа- и автомобилестроении и т.д.

**Разработанные архитектурные решения**

Порты ввода вывода и коммуникации

Идея основана на прямом вводе в память узлов супервычислителя экспериментальных данных (рис. 1).

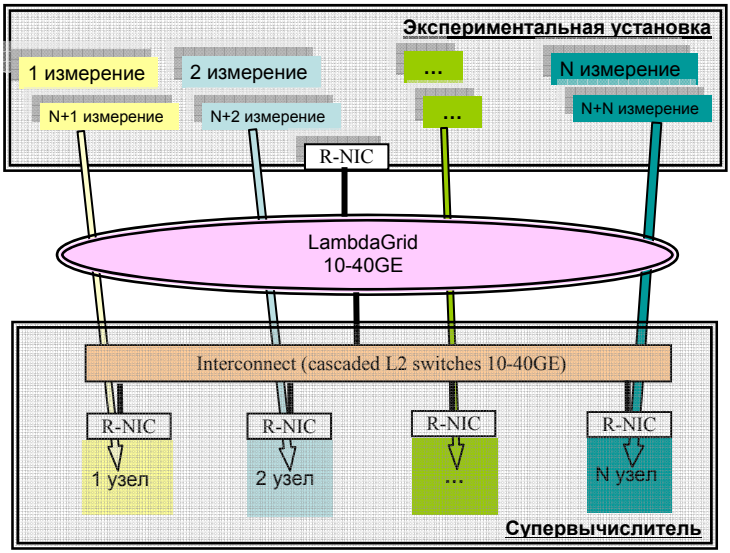



Рис. 1. Идеальная модель обработки интенсивных потоков данных

Задержка обмена данными между системами имеет два компонента: задержки в оконечных системах и задержки в сети. LambdaGrid-технологии [1] решают задачу скоростной передачи данных между взаимодействующими системами.

Задержка приема/передачи в оконечных системах определяется скоростью порта и временем поступления данных от Ethernet порта сетевого адаптера (NIC) в буфер приложения и, наоборот, из буфера приложения в сеть (порт адаптера). Скорости портов непрерывно растут (1-10-40-100GE), и основная проблема заключается в передаче данных приложению. Эта задержка возникает как на передающей, так и на приемной стороне и определяется внутренней сущностью NIC. В качестве I/O-портов связываемых по LambdaGRID оконечных систем целесообразно использование интеллектуальных NIC-карт (Intelligent Ethernet adapter), которые аппаратно поддерживают стеки протоколов передачи данных (TOE NIC - TCP Offload Engine) и технологии удаленного прямого доступа к памяти (R-NIC) для разгрузки CPU-узлов в связи с переходом на скорости 10–40–100GE.

Тестовая инфраструктура

Для разработки и тестирования протокола и программного обеспечения (ПО) на площадке ИМСС УрО РАН была создана инфраструктура (рис. 2).

Основные вычислительные узлы и управляющий узел суперкомпьютера МВС-1000/16П (16 вычислительных узлов, операционная система Linux) объединены двумя приватными подсетями 100 Мбит/с:

192.168.3.0/24 – сеть передачи команд и данных с управляющего узла на вычислительные узлы;

192.168.2.0/24 – сеть внутреннего интерконнекта суперкомпьютера для обмена данными между вычислительными узлами.

Экспериментальная установка (ЭУ) PIV (компьютер для управления камерами и лазером с программным обеспечением Actual Flow под управлением операционной системы Windows)

подключена на скорости 100 Мбит/с непосредственно к сети внутреннего интерконнекта суперкомпьютера.

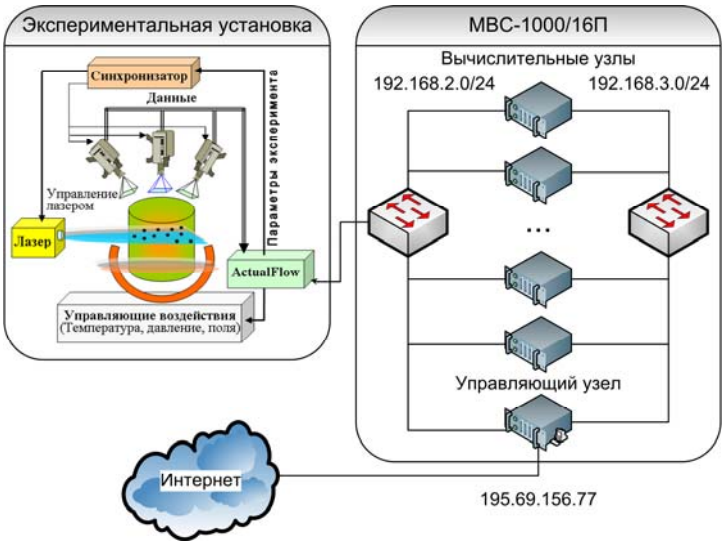



Рис. 2. Структура сети передачи данных

### Потоки данных

Каждое измерение, сгенерированное ЭУ, является парой изображений, созданных через определенный интервал времени. Согласно требованиям прикладной задачи каждое измерение можно рассчитывать на суперкомпьютере независимо от других. Это позволяет отказаться от однозначного отображения измерений на вычислительные узлы.

Поскольку измерение является парой изображений, необходимо передать с ЭУ на вычислительный узел суперкомпьютера два блока бинарных данных. Результат обработки каждого измерения (скорости частиц, команды управления экспериментом) является блоком бинарных данных, которые необходимо возвращать на ЭУ. Помимо результата вычисления с вычислительных узлов кластера на ЭУ может передаваться служебная текстовая информация, например, предназначенная для записи

в журнал работы сервера на ЭУ (диагностика, тестирование, анализ производительности).

Программное обеспечение и протокол передачи данных

Разработанное ПО предназначено для обеспечения взаимодействия прикладных процессов ЭУ с вычислительными узлами суперкомпьютера. Взаимодействие осуществляется с использованием разработанного протокола передачи данных, получившего название «Протокол PIV».

Протокол PIV использует TCP, работает по схеме запрос-ответ и предназначен для передачи двух блоков бинарных данных как от сервера к клиенту, так и от клиента к серверу. Поля заголовков кодируются в сетевом порядке байт.

Формат пакета изображен на рис. 3.

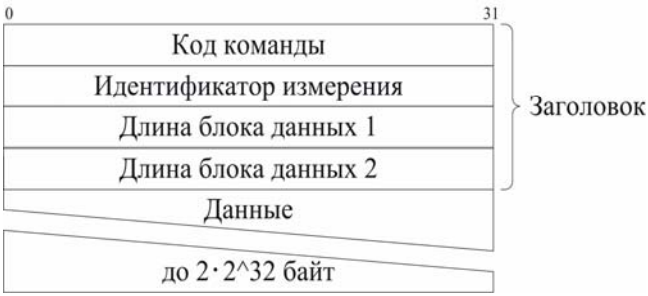



Рис. 3. Формат пакета

Поле «Код команды» содержит один из следующих кодов команд:

- Команды запросов клиента:
  - ✓ OPCODE_GET – запрос у сервера данных следующего измерения.
  - ✓ OPCODE_PUT – передача на сервер результатов обработки измерения.
- Команда ответов сервера:
  - ✓ OPCODE_ERROR – выполнение запроса закончилось ошибкой. TCP-сессия разрывается.
  - ✓ OPCODE_EOD – данных больше нет. Клиент завершает работу.

- ✓ OPCODE_GET_RESP – код ответа на запрос OPCODE_GET. В пакете с этим кодом передаются запрошенные данные.
- ✓ OPCODE_PUT_RESP – код ответа на запрос OPCODE_PUT. Подтверждение от сервера об успешном приеме данных.

Поле «Идентификатор измерения» содержит порядковый номер измерения. Поля «Длина блока данных 1» и «Длина блока данных 2» – длины в байтах соответственно первого и второго подблока данных в поле «Данные» пакета.

Диаграммы допустимых последовательностей кодов команд в пакетах (для одного цикла запрос-ответ) представлены на рис. 4.

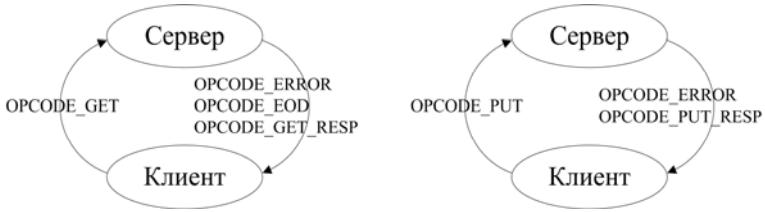



Рис. 4. Диаграммы допустимых последовательностей кодов команд в пакетах

Разработанный протокол добавляет 16 байт заголовочных данных на каждое отправляемое измерение.

### Схема работы протокола

На рис. 5 представлена временная диаграмма обмена данными между вычислительным узлом и экспериментальной установкой.

В случае возникновения ошибки передачи данных сторона, обнаружившая ошибку, закрывает TCP-соединение. При закрытии соединения сервер добавляет в очередь готовых к обработке измерений те измерения, которые были переданы через закрытое TCP-соединение, но на которые не был прислан результат расчета. Это позволяет избежать случая, когда из-за

сбоя некоторые измерения окажутся не обработанными. Клиентское ПО при разрыве TCP-соединения прекращает проведение текущего расчета и после таймаута заново инициализирует соединение с серверным ПО на ЭУ.

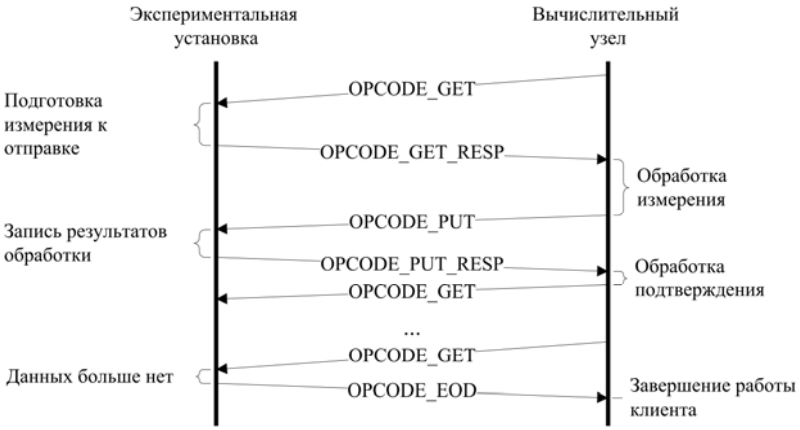



Рис. 5. Временная диаграмма протокола

### Серверное программное обеспечение

Серверное ПО запускается на экспериментальной установке. Оно построено по событийно-ориентированной архитектуре. Использование событийно-ориентированной архитектуры и механизма неблокирующих сокетов позволяет в асинхронном режиме обслужить множество подключенных клиентов внутри одной нити исполнения программного кода. Положительный эффект достигается за счет фактического распараллеливания операций подготовки данных и процессов передачи данных через сеть. Для уменьшения влияния на производительность блокирующих операций чтения данных с диска сервер реализован многопоточным.

### Клиентское программное обеспечение

Клиентское ПО запускается на вычислительных узлах суперкомпьютера. Оно осуществляет взаимодействие между прикладным алгоритмом, который выполняет необходимые расче-

ты, и серверным ПО. Для параллельного запуска множества клиентов на различных вычислительных узлах суперкомпьютера используется MPI.

Для реализации алгоритма прикладной задачи в код клиента необходимо встроить функцию обработки измерения, написанную на языке C. Прототип этой функции:

```
struct Response {
  char *log;    size_t log_length; /*Поле служебной информации, длина блока данных*/
  char *data; size_t data_length;/*Поле результатов расчета, длина блока данных*/ };
void processing(struct Response *res, uint32_t id, char *a, size_t a_len, char *b, size_t b_len);
```

Первым параметром передается указатель на структуру Response, через которую функция возвращает результаты расчета. Второй параметр – идентификатор измерения, последующие параметры – указатели на блоки данных и длины соответствующих блоков. При заполнении структуры Response для выделения памяти под данные log и data необходимо использовать функцию malloc. После отправки результатов на сервер выделенная память будет автоматически освобождена.

Анализ производительности разработанного решения

Исследование производительности было проведено путем измерения скорости передачи данных между экспериментальной установкой под управлением Windows и вычислительными узлами кластера МВС-1000. График зависимости скорости передачи данных от размера блока и от числа подключенных узлов приведен на рис. 6.

Низкая скорость передачи блоков данных размером 8 Кб (передаваемое измерение содержит два блока данных) показывает недостаточную эффективность стека протоколов TCP/IP в случае, когда IP-пакеты передаются частично заполненными. При размерах блоков, близких к размерам данных реальных измерений скорость достигает максимума, начиная с 2–4 потоков передачи данных.

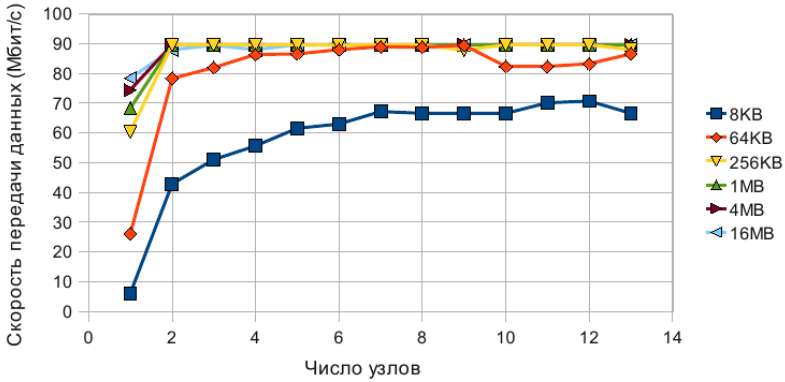



Рис. 6. Зависимость скорости передачи данных от числа подключенных узлов

Максимум скорости имеет значение порядка 90 Мбит/с, что составляет 90 % от пропускной способности канала. Для стека протоколов TCP/IP 90 % использования канала является значением, близким к предельно достижимой эффективности [5].

Спроектирован протокол и разработано программное обеспечение для передачи данных с экспериментальной установки на узлы вычислительного кластера.

Исследование производительности показало, что при передаче блоков данных размером более 256 Кбайт удалось добиться 90% использования канала передачи данных под полезную нагрузку. Результат в 90% и отсутствие падения скорости с ростом числа подключенных клиентов указывает на то, что ограничителем пропускной способности стала существующая локальная сеть. Из этого можно сделать вывод, что разработанное решение имеет запас производительности, который позволит использовать решение в сетях с большей скоростью передачи данных.

## Список литературы

1. Масич А.Г., Масич Г.Ф. GIGA UrB RAS подход к LambdaGrid парадигмам вычислений // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: тр. междунар. суперкомпьютерной конф. – М.: Изд-во МГУ, 2010. (в печати)

2. Степанов Р.А., Масич А.Г., Масич Г.Ф. Инициативный проект «Распределенный PIV» // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: тр. всерос. суперкомпьютерной конф. – М.: Изд-во МГУ, 2009. – С. 360–363.

3. Инфрастуктура распределенного эксперимента / А.Г. Масич (и др) // XVI конф. представителей региональных научно-образовательных сетей «RELARN-2009»: тез. докл. – М.–СПб, 2009. – С. 58–60.

4. Adrian R.J. Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimentry vs. particle image velocimentry // Appl. Opt. 1984. Vol. 23. Pp. 1690-1691.

5. Dykstra P. Protocol Overhead. – URL: http://sd.wareonearth.com/~phil/net/overhead/.

**[1]В.П.Матвеенко, [1]Р.А. Степанов, [1]Б.И. Мызникова, [1]И.Э. Келлер, [2]М.Г. Бояршинов**

[1] Институт механики сплошных сред УрО РАН, г. Пермь
[2] Пермский государственный технический университет

## ОПЫТ ОРГАНИЗАЦИИ МАГИСТРАТУРЫ ПО ВЫСОКОЭФФЕКТИВНЫМ ВЫЧИСЛИТЕЛЬНЫМ ТЕХНОЛОГИЯМ В МЕХАНИКЕ В ПЕРМСКОМ ГОСУДАРСТВЕННОМ ТЕХНИЧЕСКОМ УНИВЕРСИТЕТЕ

Пермский край известен высокоразвитыми горнодобывающим, химическим, металлургическим, строительным комплексами, машино- и приборостроением. На предприятиях края осуществляется разработка и производство наукоемкой продукции, в том числе авиационных и ракетных двигателей, газотурбинных установок, артиллерийских систем, оборудования для добычи нефти, оптоволоконных гироскопов, новых материалов на основе порошковых нанокомпозиций. Естественно, что в пермских инженерных и научных коллективах возникают задачи обработки информации, математического моделирования, проектирования и оптимизации

указанных процессов, требующие применения высокоэффективных вычислительных алгоритмов и масштабных вычислительных работ. Поэтому в Пермском государственном техническом университете создан Центр ВВС, оснащенный мощным вычислительным кластером. В связи с этим стала чрезвычайно актуальной задача профессиональной подготовки магистров, способных квалифицированно разрабатывать алгоритмы и программные коды для существующей техники.

С 2010 года в Пермском государственном техническом университете открыт набор в магистратуру «Высокоэффективные вычислительные технологии» по направлению 150300 «Прикладная механика». Квалификационное требование, которому должен удовлетворять выпускник магистратуры, – подготовка к эффективному численному решению широкого спектра связанных краевых задач механики сплошной среды, описывающих физические процессы различной природы, с помощью известных (коммерческих) либо созданных самостоятельно пакетов программ и сред параллельных вычислений на кластерах ПГТУ, ИМСС и других российских вычислительных центров.

В магистратуру предполагается принимать не только бакалавров и специалистов направления 150300 «Прикладная механика», но и других направлений серии 15****, направлений «Прикладная математика и информатика», «Информатика и вычислительная техника» (при условии успешной сдачи вступительного междисциплинарного экзамена) ввиду широкой востребованности специалистов в области выполнения высокоэффективных вычислительных работ. Естественное различие начальных условий образования магистрантов планируется нивелировать номенклатурой и содержанием дисциплин, индиивидуальной работой со студентами и значительными объемами самостоятельной работы студентов под руководством профессорско-преподавательского состава ПГТУ и высококвалифицированных сотрудников научных учреждений Пермского центра Уральского отделения Российской академии наук.

Учебный план магистерской программы[1] включает следующие дисциплины, распределенные по четырем блокам (с указанием доли блока в процентах от общей аудиторной учебной нагрузки):

| № | Наименование дисциплины | Аудиторных часов в неделю | Семестр |
|---|---|---|---|
| 1. | Естественно-научный блок (20 %): | | |
| | Модели механики сплошных сред | 6 | I–III |
| | Уравнения математической физики (спецразделы) | 3 | II |
| 2. | Численные методы (30 %): | | |
| | Вычислительные методы линейной алгебры, дифференциального и интегрального исчисления | 2 | I |
| | Сеточные, проекционные и спектральные методы решения задач | 6 | II–III |
| | Пакеты прикладных вычислений механики сплошных сред | 4 | IV |
| 3. | Программирование и параллельные вычисления (30 %): | | |
| | Алгоритмические языки C++ и FORTRAN | 2 | I |
| | Технологии и пакеты распараллеливания задач | 10 | I–III |
| 4. | Гуманитарный блок (20 %): | | |
| | История и методология прикладной механики | 4 | I |
| | Английский язык в научном общении | 4 | I–II |
| | Педагогика | 1 | I |

По всем дисциплинам второго и третьего блоков предусмотрено выполнение курсовых работ.

Дисциплина «Модели механики сплошных сред» первого блока содержит элементы тензорного анализа, фундаментальные законы и основные уравнения механики, термодинамики и электродинамики сплошной среды и вытекающие из них клас-

[1] Учебный план доступен по адресу http://dl.dropbox.com/u/3612585/Magister/PlanVVTmag.pdf.

сические математические модели. В третьем семестре предполагается проведение занятий в форме семинаров, где с участием преподавателей, ведущих дисциплины специализации магистратуры, обсуждаются индивидуальные задачи, которыми магистранты занимаются в рамках своей научной работе, начиная со второго семестра. «Уравнения математической физики (спецразделы)» – классический курс, в рамках которого обсуждаются особенности постановок, вопросы существования и единственности решения, а также рассматриваются классические методы решения краевых задач на основе линейных уравнений в частных производных.

Учебные дисциплины второго блока предполагают изучение классических численных методов линейной алгебры, дифференциального и интегрального исчисления, сеточных методов решения обыкновенных дифференциальных уравнений и уравнений в частных производных, способов оценок погрешностей, определения условий устойчивости разностных схем и сходимости численных решений. Пристальное внимание уделяется высокоэффективным алгоритмам (расщепление, факторизация, переменные направления и проч.) решения многомерных задач математической физики. Значительный объем времени отводится на изучение проекционных методов, способов построения решений методами конечных и граничных элементов.

Третий блок включает курс алгоритмических языков C++ и FORTRAN, совершенно необходимых для создания кодов (как оригинальных прикладных пакетов, так и расширяющих стандартные), но, к сожалению, не нашедший места в Государственных стандартах и учебных планах подготовки бакалавров большинства из указанных выше направлений. Курс «Технологии и пакеты распараллеливания задач» в начальной своей части имеет целью дать навыки работы в среде параллельных вычислений на кластерах, работающих под управлением ОС Linux и Windows. Освоение принципов запуска и отладки задач, постановки их в очередь даст возможность легко использовать мощности любых вычислительных центров. Вторая часть курса

посвящена изучению программного интерфейса MPI, который позволяет обмениваться информацией между параллельными процессами. Заключительная часть курса состоит в ознакомлении с библиотеками стандартных алгоритмов линейной алгебры и с пакетами программ, позволяющими решать целый ряд стандартных задач механики деформируемых твердых тел, жидкостей и газов. Практическая сторона курса выражается в том, что у выпускника магистратуры будут сформированы компетенции, позволяющие ему преобразовывать существующие последовательные алгоритмы в параллельные и создавать собственные с использованием стандартных библиотек и пакетов прикладных программ. Лабораторные работы по дисциплине «Технологии и пакеты распараллеливания задач» проходят в классе, оснащенном современным высокоскоростным коммуникационным оборудованием, дающим возможность выхода на кластеры ПГТУ, ИМСС и других вычислительных центров России.

Все учебные дисциплины читаются докторами и кандидатами наук, профессорами и доцентами, главными и ведущими научными сотрудниками Института механики сплошных сред УрО РАН, ПГТУ, ПГУ, Пермского научного центра УрО РАН., как правило, в аудиториях и лабораториях ИМСС. Со второго семестра вплоть до защиты магистерской диссертации предусматривается постоянная научно-исследовательская работа в рамках семестровой работы и практик под руководством ведущих ученых из научных организаций города Перми, нацеленная на решение реальных прикладных задач по заказам организаций и предприятий Пермского края. Постоянное пребывание в научной среде является хорошим стимулом к самостоятельной исследовательской работе и самообразованию.

**В.Я. Модорский, Н.В. Струк**

Пермский государственный технический университет

# МОДЕЛИРОВАНИЕ ГАЗОДИНАМИЧЕСКИХ ПРОЦЕССОВ И НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ В ЭКСПЕРИМЕНТАЛЬНОЙ УСТАНОВКЕ

В технике известен сложный, опасный, в большинстве случаев непрогнозируемый режим, который получил название «флаттер». При флаттере реализуется особая форма колебательной неустойчивости, источником которой может являться резонансное взаимодействие системы «газ – конструкция». Частота и форма наблюдающихся при этом волн зависят от геометрических и физико-механических характеристик конструкции и параметров нагрузки.

Резонансные режимы могут приводить к различным функциональным отказам или разрушению конструкции. Таким образом, исследование взаимодействий газодинамического потока с конструкцией является актуальной задачей.

Для поиска опасных резонансных режимов в газонаполненной конструкции необходимо провести комплексное экспериментально-теоретическое исследование параметров динамического поведения системы «газ – конструкция».

Реализовать расчетно-экспериментального подход можно в два этапа. На первом этапе провести работы по моделированию процессов в экспериментальной установке раздельно для газа и конструкции. На втором этапе реализовать вычислительные эксперименты (ВЭ), моделирующие процессы в динамической системе «газ – конструкция» в связной постановке.

Численное моделирование резонансного взаимодействия газа с конструкцией может быть реализовано с помощью современных программных комплексов инженерного анализа. Для моделирования течений жидкости и газа предлагается использование FlowVision или CFX, а для оценки напряженно-деформированного состояния конструкции – ABAQUS или ANSYS.

В рамках первого этапа проведен газодинамический ВЭ, моделирующий работу экспериментальной установки.

Геометрия расчетной области для данной задачи была спроектирована в инженерном пакете SolidWorks и в формате VRML импортировалась в FlowVision. Твердотельная модель проточной части камеры представляет собой три соосных цилиндра, один из которых моделирует полость камеры, а два других – штуцеры для подвода и отвода рабочего тела.

Исследование течения газа для принятой конструкции проточной части модельной камеры выполнялось с использованием следующей физической модели:

процессы рассматривались в трехмерной динамической постановке;

поток рассматривался однофазным, в качестве рабочего тела выбран воздух;

стенки модельной камеры непроницаемые, нетеплопроводные;

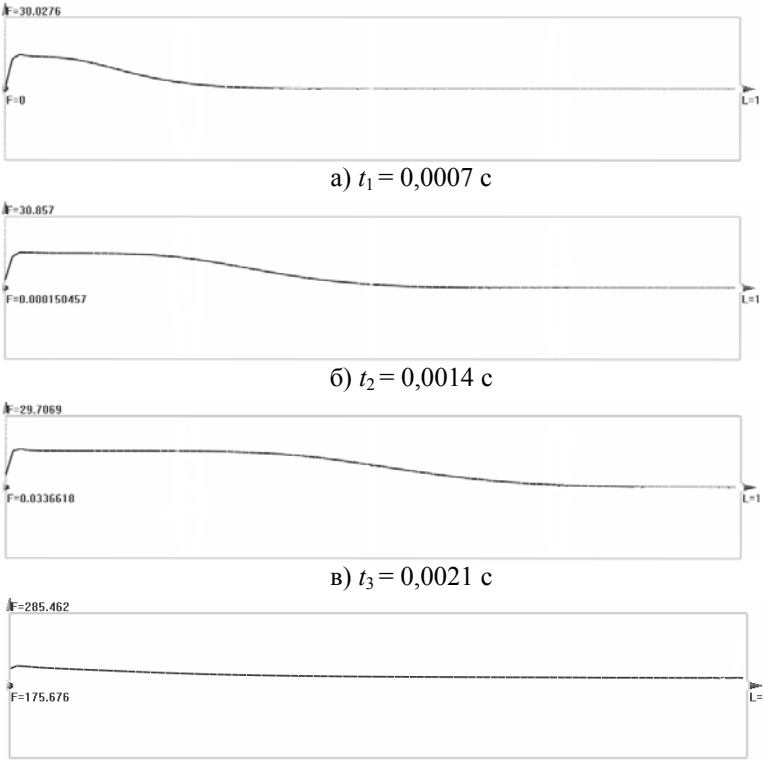
в начальный момент времени модельная камера заполнена воздухом, значение давления равно атмосферному, температура 293 К.

Для моделирования течения воздуха внутри камеры выбрана математическая модель «Полностью сжимаемая жидкость» FlowVision, которая включает в себя законы сохранения массы, импульса, энергии и уравнение состояния. Исходная система дифференциальных уравнений в частных производных замыкается начальными и граничными условиями. При описании границ использованы типы «Стенка», «Вход/Выход» и «Свободный выход» FlowVision.

Для решения поставленной задачи задавалась равномерная расчетная сетка 150×20×20, состоящая из 60 000 расчетных ячеек. Шаг по времени задавался величиной $3 \cdot 10^{-4}$с.

В качестве базового варианта рассмотрен процесс течения воздуха в модельной камере. Для автономной отладки газодинамической части задачи рассмотрены режимы формирования вынужденных колебаний.

Проведен анализ переходных режимов при пуске экспериментальной установки. Для исследования динамики установления процесса результаты выводятся в виде кинограмм (рис. 1).




а) $t_1 = 0{,}0007$ с

б) $t_2 = 0{,}0014$ с

в) $t_3 = 0{,}0021$ с

г) установившийся процесс

Рис. 1. Пример распределения избыточного давления вдоль оси модельной камеры, Па

Изменения давления во времени представлены на уровнеграммах (рис. 2), которые позволяют определить уровни максимального и номинального давления в соответствующих сечениях установки, а также параметры колебаний воздуха.

Показано, что в газодинамической полости без учета конструкции и внешнего возбуждения колебания в газе не поддерживаются.

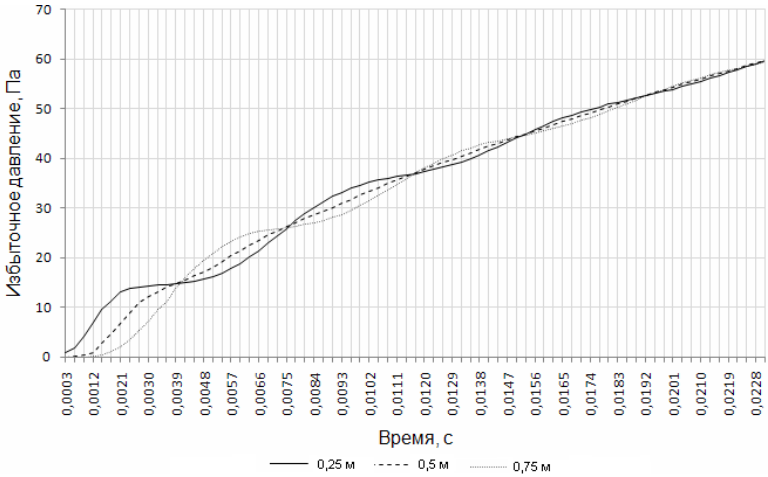



Рис. 2. Избыточное давление на различных расстояниях от места подачи воздуха

Расчет характеристик колебаний конструкции модельной камеры выполнялся в многопроцессорном пакете инженерного анализа ANSYS Workbench.

Твердотельная модель камеры экспериментальной установки была спроектирована в приложении Design Modeler. Она представляет собой два соосных полых цилиндра, один из которых моделирует корпус модельной камеры, а другой – регулировочный груз.

На твердотельной модели камеры была сгенерирована неоднородная сетка с фактором плотности 25. Сетка создавалась с помощью тетраэдрических твердотельных элементов.

К внутренней стенке торца модельной камеры прикладывалась циклическая нагрузка давления с амплитудным значением 1 МПа (10 атм.).

Для моделирования крепления модельной камеры использовалось фиксированное закрепление – Fixed Support (жесткая заделка).

По результатам расчета собственных частот и форм колебаний интересующие нас продольные колебания конструкции возникают при частоте 312 Гц, что близко к значению, полученному аналитическим методом.

При выполнении расчета вынужденных частот была построена амплитудно-частотная характеристика (рис. 3).

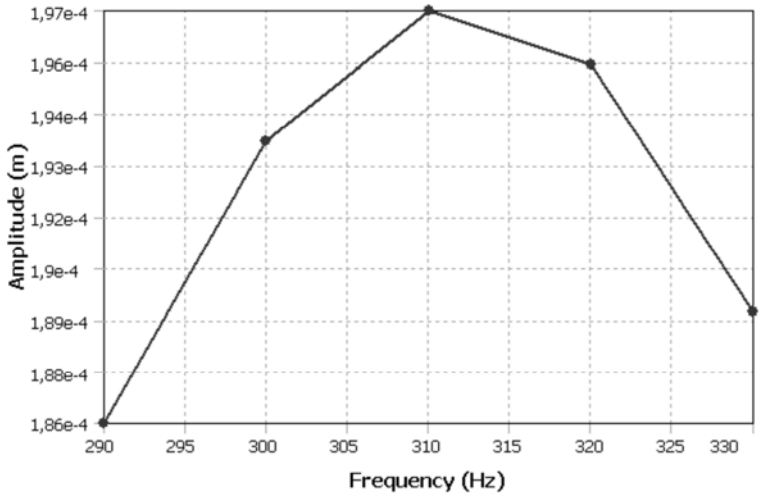



Рис. 3. Амплитудно-частотная характеристика

На данном рисунке видно, что максимальная амплитуда перемещений наблюдается при частоте приложения нагрузки 310 Гц. Собственная частота модельной камеры 312 Гц. Возникает условие резонансного взаимодействия.

Для реализации второго этапа по моделированию процессов в динамической системе «газ – конструкция» в связной постановке к вычислительным ресурсам предъявляются особые требования. Решение будет получено на ВВК ПГТУ с производительностью 4 ТФлокс.

**О.Г. Монахов**

Институт вычислительной математики и математической геофизики
СО РАН, Новосибирск

# РАСПАРАЛЛЕЛИВАНИЕ ЭВОЛЮЦИОННОГО АЛГОРИТМА ОПТИМИЗАЦИИ ФИНАНСОВЫХ СТРАТЕГИЙ ДЛЯ РЕАЛИЗАЦИИ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ[1]

**Введение и постановка задачи.** В практике биржевой торговли одним из основных направлений при выработке торговых стратегий (торговых алгоритмов) является технический анализ ценовых рядов с помощью множества индикаторов [1–4]. В соответствии с принятой торговой стратегией, выраженной в виде набора правил, поведением ценового ряда и значениями индикаторов инвестор принимает решение о совершении/несовершении сделки купли-продажи в данный момент времени. При совершении сделки инвестор руководствуется соображениями максимизации доходности и минимизации риска. Принятый набор правил, составляющий торговую стратегию, и используемые индикаторы имеют эмпирический характер и значения их параметров определяются в основном опытным путем (методом проб и ошибок). Однако, как показывают эксперименты, такой подход с использованием известных правил и статически задаваемых параметров часто приводит к убыточным стратегиям. Использование мощных вычислительных систем для торговли на бирже в США уже стало обозначаться новым термином «высокочастотная торговля» (high-frequency trading) и позволяет компьютерным программам (торговым роботам) самостоятельно отслеживать данные по нескольким индексам на фондовых биржах, оптимизировать торговые стратегии и совершать миллионы сделок за максимально короткий промежуток времени.

В работе рассматривается проблема поиска параметров данной стратегии биржевой торговли $S$ с целью оптимизации

[1] Работа выполнена при поддержке РФФИ, проект № 08-01-00857.

заданной целевой функции $F$, характеризующей ее качество. Будем считать, что цена на акцию представлена в виде ценового ряда $\{C_i\}$, $1 \le i \le N$, с заданной частотой (например, минутные или часовые цены), где $C_i$ – цена закрытия в момент $i$. Пусть $r_{i+1} = C_{i+1} - C_i$. Важными инструментами технического анализа рынка акций являются скользящие средние, индикаторы и осцилляторы, на основе которых формируются множество торговых стратегий и которые помогают инвестору принимать решения о купле-продаже акций [1–4].

Пусть мы имеем индикатор технического анализа $I_i^{(n)} = f(C_i, C_{i-1}, ..., C_{i-n})$. Обобщенная торговая стратегия $S(I_i^{(n)})$, основанная на индикаторе $I_i^{(n)}$, определяется следующими соотношениями:

$$\varphi_{i+1} = \begin{cases} 1, если\ I_i^{(n)} > \varepsilon_1, \\ \varphi_i, если - \varepsilon_2 \le I_i^{(n)} \le \varepsilon_1, \\ -1, если\ I_i^{(n)} < -\varepsilon_2. \end{cases}$$

где $\varepsilon_1, \varepsilon_2 > 0$ – уровни значимого изменения индикатора $I_i^{(n)}$. Состояние покупки в данной торговой стратегии наступает при $\varphi_{i+1} = 1$, а состояние продажи наступает при $\varphi_{i+1} = -1$. Решение о сделке (купли/продаже) принимается при смене состояний: $\varphi_i \varphi_{i+1} = -1$.

Эта стратегия $S(I_i^{(n)})$ будет использована как темплейт (с некоторыми модификациями) для определения торговых стратегий на основе различных индикаторов технического анализа и поиск оптимальных значений свободных параметров $(n, \varepsilon_1, \varepsilon_2)$, определяющих стратегию с наилучшими показателями доходности, и осуществляться с помощью ГА.

Поиск оптимальных стратегий (с наилучшими показателями доходности и/или риска) может осуществляться для каждого типа акций отдельно в динамике торговых сессий, с постоянной адаптацией к рыночной ситуации или в квазидинамическом режиме, когда расчет оптимальных параметров происходит

либо через заданные периоды времени, либо по выполнению определенных условий (например, по достижении заданного уровня потерь).

Пусть торговая стратегия $S$ содержит параметры $P = \{p_n\}, n > 0$, описывающие значения целочисленных и действительных коэффициентов и переменных, значения индексов, параметры структур данных, константы и некоторые примитивные операции алгоритма (величины инкрементов и декрементов, знаки переменных, логические операции и отношения, типы округления переменных).

Целевая функция $F$ оценивает величину доходности стратегии $S$, полученную при заданных значениях параметров $P = \{p_n\}, n > 0$ и при входных данных ценового ряда $\{C_i\}$: $F_i = F_i(S(P, C_j)), j \le i, 1 \le i \le N$.

Таким образом, проблема оптимизации торговой стратегии состоит в следующем: для данной стратегии $S$ и заданного набора значений ценового ряда $\{C_i\}$, $1 \le i \le N$, необходимо найти такие значения параметров $P^*$ стратегии $S$, что $F_N(S(P^*, C_i)) \ge F_N(S(P, C_i))$, для $1 \le i \le N$, при любых других значениях параметров $P \in Dom(P)$.

Целью данной работы является описание подхода к оптимизации торговых стратегий, основанного на эволюционных вычислениях, и его распараллеливанию. Представлен параллельный генетический алгоритм (ГА), реализованный на графических процессорах NVIDIA, который в процессе торговых сессий осуществляет автоматический поиск оптимальных параметров торговых стратегий и индикаторов с точки зрения максимизации показателей доходности. Для решения данной проблемы в работе предлагается подход, основанный на применении генетических алгоритмов (ГА) [5,6] с использованием предварительного знания прикладной области (множества индикаторов), выборе обобщенной схемы торговой стратегии, задаваемой в виде темплейта с параметрами [7], и ограничении пространства поиска оптимальных параметров.

Генетический алгоритм основан на моделировании процесса естественного отбора в популяции особей, каждая из которых представлена точкой в пространстве решений задачи оптимизации. Особи представлены структурами данных *Gen*–хромосомами, включающими свободные (неопределенные) параметры $p_k$ торговой стратегии *S*: $Gen = \{P\} = \{p_1, p_2, ..., p_k\}, k > 0$. Эти параметры определяют необходимую торговую стратегию *S(Gen)*. Каждая популяция является множеством структур данных *Gen* и определяет множество стратегии *S(Gen)*.

Примем, что целевая функция (fitness function, функция качества, функция пригодности) *F* вычисляет суммарную доходность $D_N$, полученную в результате торговли в соответствии с данной стратегией *S* за *N* шагов для заданного ценового ряда $\{C_i\}$, $1 \le i \le N$:

$$F = D_N = \sum_{m=1}^{N_{br}} (d_m^{br} - Comm), \text{ где } d_m^{br} = \frac{C_m^{sell} - C_m^{buy}}{C_m^{buy}},$$

$C_m^{sell}, C_m^{buy}$ – цены продажи и покупки в *m*-й сделке, $N_{br}$ – число сделок за *N* шагов моделирования, *Comm* – размер постоянных комиссионных за каждую сделку. Целью алгоритма является поиск максимума *F*.

**Экспериментальные результаты.** Предложенный генетический алгоритм с использованием темплейтов был успешно применен для адаптивной оптимизации торговых стратегий, основанных на следующих наиболее популярных инструментах технического анализа: экспоненциальнных скользящих средних (EMA–exponential moving average), индекса относительной силы (RSI–relative strength index), темпа изменения цены (ROC–price rate-of-change), момента (Momentum), метода схождения-расхождения скользящих средних (MACD–Moving Average Convergence/Divergence) [1–4].

Для экспериметов были рассмотрены ценовые ряды с минутными интервалами для акций ГАЗПРОМа (10 000 точек), NIKKEI (10000 точек), DJIA – Dow Jones Industrial Average (10000 точек), для периода с 16.04.2006 по 16.06.2006. Мы использовали первые 5000 точек для обучения и остальные точки – для тестирования.

Число итераций и размер популяции выбирались экспериментальным путем, основываясь на параметрах из [5,6]. Значения основных параметров в экспериментах следующие: размер популяции – от 229 376 до 1 548 288, число итераций – 30, частота мутации – 0,15, частота кроссовера – 0,7, комиссионные – 0,001.

Генетический алгоритм оптимизации торговых стратегий был реализован в системе эволюционного синтеза алгоритмов на основе шаблонов (TES – template-based evolutionary synthesis) [7] на языке программирования *C*, стратегии также задавались на этом языке. Параллельная реализация генетического алгоритма оптимизации стратегий биржевой торговли выполнена на основе распараллеливания по данным [8] для графического процессора Fermi GF100 на видеокарте NVIDIA GeForce 470 GTX 1280MB (448 процессоров, 1215 МГц) в системе программирования CUDA [9]. В связи с ограничением на память использовались ценовые ряды в 10 000 точек. В этом случае распараллеливание по данным сводится к равномерному распределению популяции по потокам системы. В конце итераций среди всех потоков выбирается лучшее решение, эта схема минимизирует взаимодействия и позволяет получить значительное ускорение (до 178). В таблице приведены результаты сравнения параллельной (на видеокарте NVIDIA GeForce 470 GTX 1280MB, 448 процессоров) и последовательной (на одном ядре процессора INTEL Core2Quad Q6700, 2,66 ГГц) реализации генетического алгоритма для оптимизации стратегии с MACD для акций ГАЗПРОМа при размере популяции *Pop*, времени исполнения *T* (с), и полученном ускорении *Sp* по отношению к процессору INTEL Q6700. Эксперименты показали, что при данных параметрах и ограничениях время выполнения задания для оптимизации стратегии на видеокарте NVIDIA GeForce 470 GTX эквивалентно времени выполнения на кластерной системе из 122 – 178 одноядерных процессоров с частотой 2,66 ГГц.

**Результаты сравнения параллельной и последовательной реализации генетического алгоритма**

| № п/п | *Pop* | *T* (с) – *INTEL Q6700* | *T* (с)– *NVIDIA 470 GTX* | *Sp* |
|---|---|---|---|---|
| 1 | 229376 | 21090 | 166 | 127 |
| 2 | 516096 | 62080 | 363 | 171 |
| 3 | 1032192 | 128800 | 723 | 178 |
| 4 | 1548288 | 131900 | 1081 | 122 |

Используемый генетический алгоритм позволил найти значения параметров торговых стратегий, обеспечивающие увеличение функции суммарной доходности на 12–156 % для различных индикаторов, по сравнению с известными ранее стратегиями [1, 3].

Представленый подход к оптимизации торговых стратегий, основанный на индикаторах технического анализа, эволюционных вычислениях и темплейтах, был успешно применен для поиска свободных параметров стратегий с целью максимизации функции суммарной доходности. Параллельная реализация генетического алгоритма оптимизации стратегий биржевой торговли на графических процессорах GF100 на видеокартах NVIDIA позволяет получить значительное ускорение и увеличить значения функции суммарной доходности. Дальнейшее развитие данного подхода будет направлено на эволюционный синтез [7] новых торговых алгоритмов, правил и стратегий с использованием комбинаций нескольких индикаторов, поиском новых функций для анализа ценовых рядов и исследованием новых подходов для распараллеливания.

## Список литературы

1. Achelis, S.B. Technical analysis from A to Z. – Chicago: Probus. 1996.

2. Артемьев С.С., Якунин М.А. Математическое и статистическое моделирование на фондовых рынках. – Новосибирск: ИВМиМГ СО РАН, 2003.

3. LeBeau, Charles , Lucas, David W. Computer analysis of the futures market. – New-York: IRWIN, 1992.

4. Weissman, Richard L. Mechanical Trading Systems, Hoboken, New Jersey: John Wiley and Sons, Inc., 2005.

5. Goldberg, D.E. Genetic Algorithms, in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.

6. Koza, J. Genetic Programming. – Cambridge: The MIT Press,1992.

7. Monakhov, O., Monakhova, E. Evolving Templates for Synthesis of Scientific Algorithms // Computational Technologies. – 2005, № 6. – P. 3–12.

8. Монахов О.Г., Монахова Э.А. Параллельные системы с распределенной памятью: структуры и организация взаимодействий. – Новосибирск: Изд-во СО РАН, 2000.

9. NVIDIA CUDA Programming Guide. Available at. – URL: http://www.nvidia.com/object/cuda_get.html.

**Д.А. Никитенко**

Научно-исследовательский вычислительный центр МГУ
им. М.В. Ломоносова

## РЕЙТИНГ TOP50 КАК ИНДИКАТОР РАЗВИТИЯ ОБЛАСТИ HPC

С появлением первых вычислительных систем перед исследователями встал вопрос правильного и своевременного выбора инструмента для решения конкретной задачи, т.е. возникла необходимость ориентироваться во всем стремительно расширяющемся многообразии современных технологий.

Чтобы помочь правильно сориентироваться в мире высокопроизводительных вычислительных систем и иметь возможность оперативно отслеживать тенденции развития данной области, Межведомственный суперкомпьютерный центр РАН и Научно-исследовательский вычислительный центр МГУ имени М.В.Ломоносова в мае 2004 года начали совместный проект по формированию списка 50 наиболее мощных компьютеров СНГ, предоставляя как можно более подробную информацию по системам: используемая аппаратная платформа, программное обеспечение, коммуникационная среда, область применения и т.д.

Рейтинг публикуется два раза в год: весной и осенью, т.е. со своеобразным квартальным сдвигом относительно объявления очередных редакций мирового рейтинга Top500. Это позволяет иметь уже четыре относительно равномерно распределенные по всему году контрольные точки, хотя бы для передовых вычислительных систем. К примеру, в редакцию Top500 от июня 2010 года вошли 11 вычислительных систем из России.

Примечательным является и тот факт, что эти вошедшие в мировой рейтинг системы выгодно отличаются своей эффективностью, если ее понимать как отношение достигнутой производительности на тесте Linpack к теоретически достижимой: 66-67 % по списку Top500 в последние годы и почти 75 % по вышеупомянутым 11 системам РФ.

Интернет-представительство проекта (*http://supercomputers.ru)* изначально проектировалось с заложенным функционалом, аналогичным уже хорошо зарекомендовавшему себя на мировом уровне рейтингу Top500, т.е., помимо самого рейтинга (кстати говоря, с большей детализацией по системам, нежели Top500) и некоторой базовой статистики, была доступна лишь профильная лента новостей. С расширением функционала стали доступными изменения по разделам статистики по сравнению с предыдущей редакцией списка, появилась расширенная информация по системам, где держатели систем имеют уникальную возможность продемонстрировать практически в свободной форме, как используется их система, подчеркнуть все самое интересное, выделить самое значимое. В этих сведениях может быть указан подробный список прикладного ПО, выдача Linpack, примеры решения конкретных задач, фотографии установки и многое другое, что может быть сочтено авторами важным. Однако, к сожалению, далеко не всегда держатели систем находят время для того, чтобы подготовить наполнение для такой расширенной информации по системе, а полнота именно этого раздела, безусловно, интересна исключительно широкому кругу специалистов: студентам, исследователям, прикладным специалистам, разработчикам, т.е. – всем.

На данный момент на сайте рейтинга доступен актуальный список 50 наиболее мощных вычислительных систем, архив всех редакций, возможность сформировать в произвольной форме список с только интересующими параметрами, новостная лента. Раздел статистики содержит как численные данные с графиками по отельным редакциям рейтинга, так и по всей истории его ведения.

Почему же рейтинг – индикатор? Известно, что мощные вычислительные системы выделяются не только своими возможностями как вычислители, но и сопутствующими издержками. Будь то первоначальная стоимость установки, стоимость поддержки или что-то еще, цена ошибки при выборе подходящей новой системы исключительно высока, тем более что речь идет о топовых системах топовой же стоимостью. В конкурентной борьбе выживают наиболее взвешенные варианты, а потому список наиболее мощных машин как раз индицирует результат наиболее ответственных выборов в пользу той или иной платформы для решения тех или иных задач, хотя, безусловно, могут быть и исключения из правил.

Если посмотреть на текущую статистику по рейтингу, то можно увидеть некоторые устоявшиеся тенденции, например подавляющее преобладание кластерных систем, доминирующая роль компании Intel как производителя основных процессоров или растущий масштаб суперкомпьютерных установок.

Однако есть и тенденции завуалированные. Например, при всех достоинствах и активном росте систем с использованием интерконнекта Infiniband в последнее время все больше стало появляться систем на базе Gigabit Ethernet. Объяснение достаточно простое: при таких больших масштабах использование дорогостоящего интерконнекта должно быть оправданным, а просто комплектовать системы «чтобы было» – непозволительно дорого.

Интересно также, что растет число систем в прикладной области применения (промышленность, финансы и т.п.), что говорит о растущем понимании целесообразности использования технологий HPC для создания конкурентного преимущества.

Что касается поставщиков, то осенняя редакция Top50 от сентября 2010 года показала смену лидера по количеству поставленных систем. По 15 систем у HP и IBM и всего 8 по сравнению с лидерскими 14 в предыдущей редакцией у Т-Платформ. Однако один только «Ломоносов» обладает достигнутой производительностью 350 TFlop/s на Linpack, что является почти одной третью от суммарной достигнутой производительности по всему списку…

В первом полугодии 2010 года произошло очень важное для всего сообщества высокопроизводительных вычислений событие – начато издание ежеквартального журнала «Суперкомпьютеры». Было принято решение объединить находящееся в данный момент в разработке интернет-представительство журнала с уже существующим сайтом рейтинга Top50 и таким образом дополнить статистический строгий ресурс (*http://supercomputers.ru)* живыми обсуждениями, статьями, интервью, экспертными мнениями.

В заключение хотелось бы подчеркнуть всеобщую заинтересованность не только в максимальной детализации сведений о действующих вычислительных системах, которая позволит наиболее адекватно оценивать то, какая именно система и для каких конкретно задач была установлена, но и в актуальности этих сведений, что не менее важно. Не раз составители рейтинга сталкивались с ситуацией, когда некоторая система была расформирована или претерпела модернизацию, однако об этом можно было догадаться только по косвенным признакам. Поэтому составители рейтинга исключительно благодарны как представителям поставщиков, так и конечным держателям систем, которые сообщают открытые сведения о своих системах своевременно, информируют о любых неточностях, вносят новые предложения по расширению функционала, проявляют всяческую инициативу и не остаются безучастными к достоверности и полноте этого крайне важного для всех в отрасли ресурса.

## Список литературы

1. Никитенко Д.А. Top50 — рейтинг наиболее мощных суперкомпьютеров СНГ // Суперкомпьютеры. – 2010. – № 2.

2. Никитенко Д.А. Рейтинг Top500 – что нового? // Суперкомпьютеры. – 2010. – № 3.

3. TOP 50 суперкомпьютеров (http://supercomputers.ru).

**А.В. Никологорская, Ф.Н. Ясинский**

Ивановский государственный энергетический университет

## ОБ ИСПОЛЬЗОВАНИИ СУПЕРКОМПЬЮТЕРА ПРИ ПОСТРОЕНИИ ГИБРИДНОГО МЕТОДА ПРОГНОЗА ПОТРЕБЛЕНИЯ ЭЛЕКТРОЭНЕРГИИ НА ОСНОВЕ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ

Построение оптимального прогноза величины потребляемой электроэнергии является актуальной задачей. В первую очередь это связано с расчётом затрат, планируемых на покупку и реализацию электроэнергии. От точности полученного прогноза будет зависеть экономия денежных средств компаний, работающих на рынке предоставления энергетических услуг.

В зависимости от характера представленных данных, предметной области и величины периода упреждения возникает проблема выбора одного из многих способов построения прогноза, который наилучшим образом будет удовлетворять заявленным требованиям.

В том случае, если наблюдаемое явление описывается с помощью временного ряда, т.е. последовательности числовой величины или набора величин, непрерывно изменяющихся во времени, особенно эффективными являются математические методы прогнозирования.

Временной ряд – это последовательность числовой величины или набора величин, непрерывно изменяющихся во времени. Анализ временных рядов с помощью математических методов позволяет получить будущее значение рассматриваемой величины на основе известных составляющих её временного ряда.

Была поставлена следующая задача: на основании предоставленной информации об энергопотреблении в Костромской области за прошлый период предсказать значения объемов потреб-

ляемой электроэнергии на сутки вперед. В данном случае временной ряд образуют значения объёмов потребляемой электроэнергии, изменяющиеся с течением времени. При этом различный характер динамики потребления электроэнергии в рабочие и нерабочие дни привел к разбиению исходного ряда на две составляющих: для будних дней и для выходных соответственно.

В данной работе исследована принципиальная возможность построения гибридного метода прогнозирования и возможность использования суперкомпьютера для ускорения вычислений.

При построении гибридного метода применён следующий подход: в качестве основы были выбраны несколько методов прогнозирования с различными характеристиками и принципами действия. Далее результаты их работы объединяются в единый синтетический алгоритм с использованием определенного набора коэффициентов $\alpha_i \geq 0, \quad \sum_i \alpha_i = 1 \quad i = 1,2,3$ (рис. 1).

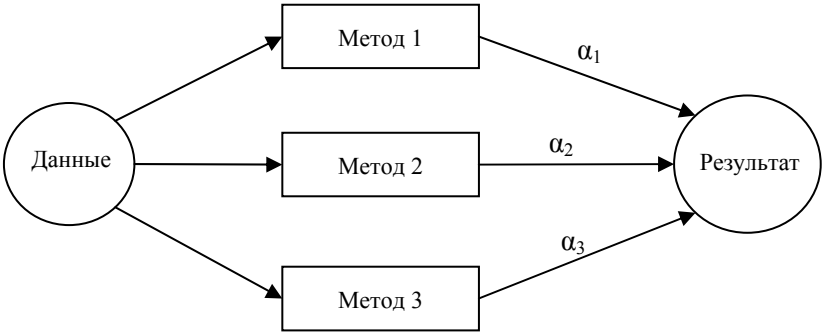



Рис. 1. Схема гибридного алгоритма

В качестве базовых методов прогнозирования были использованы три алгоритма. Выбор первого метода – фильтра Винера обусловлен квазипериодическим характером зависимости энергопотребления от времени [4, 5]. Второй – нейросетевой подход позволяет учесть нелинейность наблюдаемых процессов и сложный характер внутренних закономерностей между элементами, составляющими временной ряд [6]. Третий подход –

эволюционное моделирование призван решить задачу оптимизации [2, 3]. Параметры и характеристики этих методов подробно рассмотрены в [1].

Одной из принципиальных и сложных задач является подбор оптимального набора коэффициентов $\alpha_i$ для получения наименьшей ошибки прогнозирования.

Чтобы избежать полного перебора параметров ставится линейная задача оптимизации для нахождения указанных коэффициентов:

$$r(t) - (\alpha_1 \cdot f(t) + \alpha_2 \cdot n(t) + \alpha_3 \cdot m(t)) \to \min,$$

где $r(t)$ – реальное значение величины потребляемой электроэнергии; $f(t)$ – величина потребляемой электроэнергии, полученная с применением фильтра Винера; $n(t)$ – величина потребляемой электроэнергии, полученная с применением нейронной сети; $m(t)$ – величина потребляемой электроэнергии, полученная с применением эволюционного моделирования; $\alpha_i \geq 0, \quad \sum_i \alpha_i = 1 \quad i = 1,2,3$.

Хотя задача оптимизации и позволяет снизить количество итераций, всё же каждая итерация представляет собой очень ресурсоёмкую операцию, поскольку в ней необходимо решить одну и ту же задачу различными методами. Поэтому использование суперкомпьютера является необходимым для ускорения процесса нахождения коэффициентов. Кроме того, это позволяет применять поисковый алгоритм оптимизации с несколькими начальными точками для большего обхвата области решения.

Одним из вариантов является использование классических многопроцессорных суперкомпьютеров, поскольку это дает возможность равномерно распределить нагрузку на все процессоры в системе. Они объединяются в тройки (для трёх коэффициентов) и действуют сообща, находя оптимальный набор коэффициентов.

Другим перспективным вариантом можно считать суперкомпьютер на базе графических ускорителей по технологии CUDA. В данной системе вычисления производятся множеством блоков, состоящих из некоторого числа обрабатывающих пото-

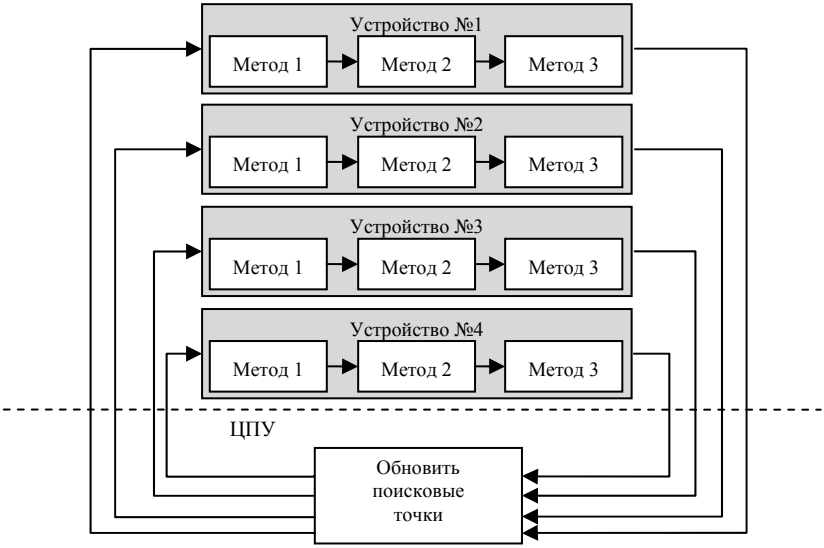
ков. В отличие от MPI Nvidia CUDA представляет собой систему с общей памятью. Технически чип графического ускорителя представляет собой несколько унифицированных мультипроцессоров, каждый из которых содержит по 8 CUDA-ядер [7].

Используемый суперкомпьютер Ивановского института ГПС МЧС России содержит в себе два графических ускорителя GTX 295, каждый из которых имеет по два чипа на плате. Общее количество ядер 240 CUDA-ядер в чипе × 2 чипа × × 2 карты = 960 CUDA-ядер при суммарном объёме видеопамяти 3,5 Гбайт.

Одним из вариантов разделения данных для решения задачи является назначение каждому потоку своего метода и исходных данных и дальнейшее совмещение полученных результатов. Однако, хотя количество ядер и велико, концепция CUDA не предполагает нагружать каждый поток большим количеством вычислений. Вместо этого используется подход, при котором тысячи потоков исполняют небольшой набор одинаковых команд над собственными данными. Это обусловлено ограничениями архитектуры CUDA [7]. В ней в каждом мультипроцессоре за один раз выполняются только 32 потока, называемые «warp». Причём желательно, чтобы потоки в этом наборе выполняли одну и ту же операцию, иначе warp разбивается на отдельные части.

Следовательно, лучше нагрузить каждое графическое устройство какой-либо одной задачей. К тому же, если для указанных базовых методов прогнозирования использовать параллельные схемы для системы CUDA (например, использование библиотеки CUFFT [7]), то можно достичь существенного ускорения расчётов.

В этом случае эффективной схемой будет использование каждого из 4 доступных устройств для решения одной поисковой задачи, т.е. использование 4 поисковых точек. Каждое устройство последовательно выполняет параллельные версии методов прогнозирования и изменяет свои коэффициенты в зависимости от выбранного алгоритма оптимизации (рис. 2).




Рису. 2. Схема гибридного алгоритма для системы CUDA
с несколькими графическими ускорителями

Таким образом, использование подобной схемы позволяет эффективно использовать вычислительные мощности суперкомпьютера на графических ускорителях для поиска оптимального соотношения коэффициентов гибридного метода.

## Список литературы

1. Никологорская А.В., Сидоров С.Г. Опыт прогнозирования энергопотребления в энергосетях Костромской области // Высокие технологии, исследования, промышленность. Исследование, разработка и применение высоких технологий в промышленности: сб. тр. IX Междунар. науч.-практ. конф. – СПб.: Изд-во Политехн. ун-та, 2010. – С. 266–270.

2. Букатова И.Л. Эволюционное моделирование и его приложения. – М.: Наука, 1979.

3. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: ФИЗМАТЛИТ, 2003.

4. Сергиенко А.Б. Цифровая обработка сигналов. – СПб.: Питер, 2002.

5. Зверев В.А., Стромков А.А. Выделение сигналов из помех численными методами. – Н. Новгород: ИПФ РАН, 2001.

6. Головко В.А. Нейронные сети: обучение, организация и применение: учеб. пособие для вузов / под общ. ред. А.И. Галушкина. – М.: ИПРЖР, 2000.

7. http://developer.nvidia.com/object/gpucomputing.html – NVIDIA GPU Computing Developer.

**А.С. Никонов[1], А.В. Русаков[2]**

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

## ВОСПРОИЗВОДИМОЕ ИСПОЛНЕНИЕ ПАРАЛЛЕЛЬНЫХ SCOOP-ПРОГРАММ

SCOOP (Simple concurrent object-oriented programming) [2, 3, 4] – технология для разработки параллельных программ. Она появилась как дополнение к языку Eiffel [7] и была интегрирована в EVE [8] – исследовательскую версию EiffelStudio.

Множество примеров доказали простоту использования SCOOP [3], однако разработка параллельных программ все еще сопровождается некоторыми проблемами. Одна из основных сложностей – отладка, так как поведение параллельной программы во время исполнения зависит от планировщика операционной системы. Под поведением параллельной программы понимается очередность доступа различных потоков к общей памяти. Возникает задача записи и повторения поведения параллельной программы для последующей отладки. Чтобы решить эту проблему в общем случае, необходимо модифицировать планировщик операционной системы, что невозможно, например, для коммерческих систем. В технологии SCOOP используется собственный планировщик и простая модель синхронизации, что позволяет решить задачу другим способом.

В данной работе реализована техника, помогающая пользователям распознать ошибки синхронизации в SCOOP-программах.

Данная техника может быть использована как основа для создания процедур, тестирующих параллельные SCOOP-программы.

**Техника повторения параллельных SCOOP-программ.** Для повторения многопоточных программ мы должны записать расписание работы потоков и повторить в точности это же расписание при следующем запуске. Расписание работы потоков – это последовательность временных интервалов. Каждый интервал содержит инструкции конкретного потока. Границы интервалов соответствуют моментам переключения контекста исполнения программы. В дальнейшем мы будем ссылаться на расписание работы потоков, полученное с помощью планировщика операционной системы, как на *физическое расписание*. Однако, используя идею *логических расписаний*, мы можем решить эту же задачу, не прибегая к помощи планировщика операционной системы.

**Логическое расписание.** Чтобы лучше понять идею логического расписания, рассмотрим пример простой SCOOP-программы, представленной ниже.

В примере поток *MAIN* создает поток *thread1*. Оба потока *MAIN* и *thread1* имеют доступ к общей памяти, объекту *shared_obj*. *MAIN* читает, а *thread1* читает и пишет в *shared_obj, k* и *j* – локальные переменные потоков.

```
class
                MAIN

create
                make

feature
make
local
                j : INTEGER
        do
                create shared_obj
                create thread1.make ( shared_obj )
                run ( thread1 )
                j := 20
                io.put_string ( "shared = " + shared_obj.value.out + ",
                j = " + j.out )
        end
run ( a_thread : attached separate MYTHREAD )
do
        a_thread.run
end

feature {NONE}
```

```
        thread1 : separate MYTHREAD
        shared_obj : OBJECT
end
```

```
class
        MYTHREAD
create
        make

feature
        make ( an_object : OBJECT )
        do
                obj := an_object
        end

        run
        local
                k : INTEGER
        do
                k := 5
                obj.add ( k )
        end

feature {NONE} -- Implementation
        obj : OBJECT
end
```

```
class
        OBJECT

feature
        add ( a_value : INTEGER )
        do
           value := value + a_value
        end

        value : INTEGER

end
```

На рис. 1 изображены примеры исполнения программы (физические расписания) на однопроцессорной машине.

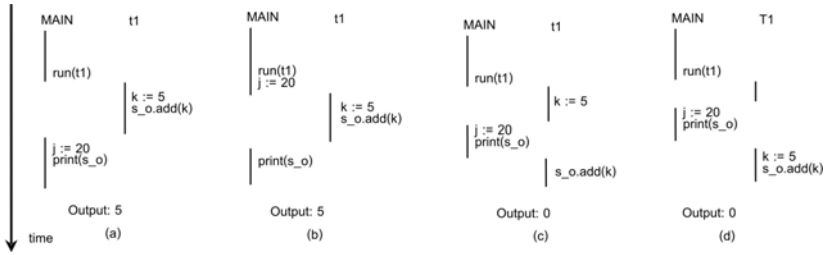



Рис. 1. Примеры исполнения многопоточной программы

Результаты выполнения программы различны и зависят от порядка доступа к общей памяти. Мы будем помещать физические расписания с одинаковым порядком доступа к общей памяти в один класс эквивалентности. В данном примере расписания (a) и (b) принадлежат к одному классу эквивалентности, (c) и (d) – к другому. Обозначим все *физические расписания* из одного класса эквивалентности как *логическое расписание*.

События синхронизации могут влиять на порядок доступа к общей памяти и, следовательно, на возможное логическое расписание. Простая SCOOP-модель не предоставляет явных примитивов синхронизации, таких как мьютексы и семафоры, синхронизация обеспечивается автоматически при каждом сепаратном вызове [2]. Поэтому мы можем рассматривать каждый сепаратный вызов как примитив синхронизации. Фактически, обязательным условием работы техники повторения параллельной программы является запись и воспроизведение последовательности событий синхронизации.

**Запись поведения параллельной SCOOP-программы.** Подход, применяемый для получения логического расписания, использует «глобальные часы» (целочисленный счетчик) планировщика SCOOP и «локальные часы» для каждого SCOOP-процессора. Алгоритм, работающий внутри SCOOP-планировщика, представлен ниже.

1. В начале локальные и глобальные счетчики имеют одинаковые значения (скажем, ноль).

2. Когда очередной сепаратный вызов готов к исполнению на конкретном процессоре, SCOOP-планировщик перехватывает вызов и сравнивает значение глобального счетчика с локальным счетчиком процессора-исполнителя.

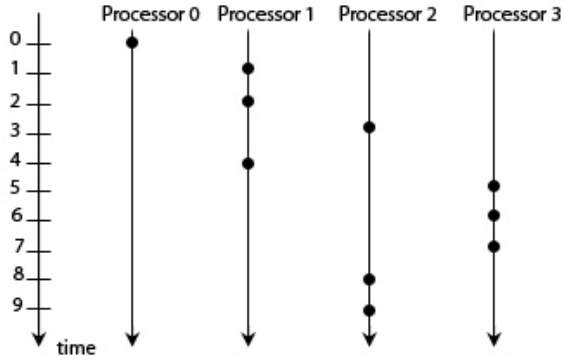
3. Если эти значения различны, планировщик только что обнаружил конец предыдущего логического интервала и начал новый логический интервал.

4. Планировщик SCOOP увеличивает значение глобального счетчика, синхронизирует с ним локальный счетчик процес-

сора и переводит в состояние «готов к исполнению» текущий сепаратный вызов.

Результат работы алгоритма – последовательность логических интервалов. На рис. 2 представлен результат работы алгоритма.

На рис. 2 временная ось ассоциирована с планировщиком SCOOP. Каждая точка обозначает сепаратный вызов, связанный с определенным процессором. Нам необходимо знать только момент, когда планировщик SCOOP обрабатывает текущий сепаратный вызов. Но не имеет значения, когда сепаратный вызов начнет физически исполняться и когда он будет закончен.




Processor 0: [0, 0]; Processor 1: [1, 2], [4, 4];
Processor 2: [3, 3], [8, 9]; Processor 3: [5, 7]

Рис. 2. Определение логического расписания

**Идентификация SCOOP-процессоров.** Логическое расписание не содержит достаточно информации для воспроизведения поведения параллельной программы. От запуска к запуску мы получаем различные идентификаторы потоков, а следовательно, и SCOOP-процессоров. Однако процедуру создания SCOOP-процессоров контролирует планировщик SCOOP. Поэтому мы можем обеспечить однозначную идентификацию процессора, введя ссылку на его родителя. Рассмотрим следующий пример:

```
class
        APPLICATION

create
        make

feature
        make
                do
                        create obj1.make
--creating new separate object inside a make feature
                        create obj2.make
--creating new separate object inside a make feature
                        create obj3.make
--creating new separate object inside a make feature
                end
feature {NONE}
        obj1, obj2, obj3 : separate OBJECT
end
```

Модель SCOOP гарантирует последовательное создание процессоров. Далее, конструкторы объектов могут исполняться параллельно. Но это не имеет значения, так как мы уже определили номер дочернего процессора и зафиксировали его родителя.

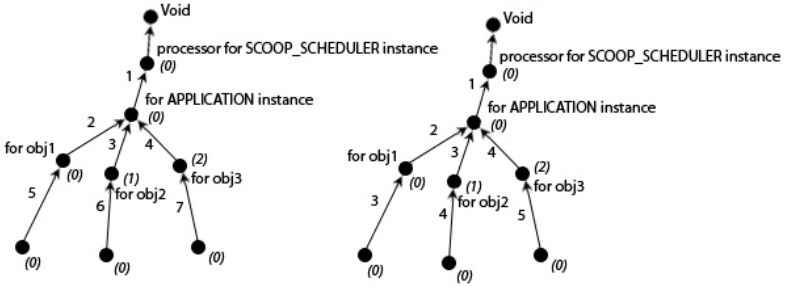



Рис. 3. Механизм создания SCOOP-процессоров. Независимо от порядка создания мы получаем одинаковую структуру процессоров

Пример создания SCOOP-процессоров в разном порядке представлен на рисунке 3. Числа над ребрами графа означают момент времени создания дочернего процессора. Числа в круглых скобках означают локальный номер процессора – для различения процессоров на одном и том же уровне дерева процессоров.

Теперь мы можем однозначно идентифицировать SCOOP-процессор при каждом запуске программы. Например, для процессора, который отвечает за объект *obj2,* мы получаем следующий идентификатор {1, 0, 0} и т.д.

**Воспроизведение параллельной SCOOP-программы.** После идентификации процессоров и получения логического расписания у нас есть достаточно информации для воспроизведения SCOOP-программы. Далее мы опишем алгоритм воспроизведения, встроенный в планировщик SCOOP.

1. Вначале *global_tick* должен быть установлен в ноль.

2. Увеличить *global_tick* и получить следующий сепаратный вызов из кольцевой очереди задач планировщика SCOOP. Каждый сепаратный вызов содержит SCOOP-процессор, который должен использоваться для выполнения вызова. Обозначим его как «текущий процессор».

3. Получить ранее записанное логическое расписание, *LSI,* для текущего процессора. $LSI = \{[a_1, b_1], [a_2, b_2], ..., [a_n, b_n]\}$. Здесь используются уникальные идентификаторы процессоров для нахождения текущего процессора в логическом расписании.

Если $\exists i \in \{1, ..., n\} / [a_i, b_i] \in LSI \wedge global\_tick \in [a_i, b_i]$, тогда выполнить текущий сепаратный вызов, иначе перейти на шаг 2 и продолжить.

Теперь возможно записывать и воспроизводить поведение параллельной SCOOP-программы в целях отладки. С использованием данной техники как основного механизма становится возможным создание тестирующих процедур, которые последовательно покрывают все множество состояний параллельной SCOOP-программы.

Тестирование – наиболее важная техника для обеспечения качества программного обеспечения. Так, недетерминированное исполнение параллельной программы не позволяет покрыть удовлетворительным способом все множество состояний программы. Поэтому тестирование должно быть основано на воспроизведении конкретного программного состояния. При этом все состояния должны генерироваться определенным образом, покрывая все множество возможных состояний. В отсутствие подобной техники ошибки синхронизации трудно отследить и воспроизвести, что делает отладку и тестирование очень непростым процессом. В данной

работе был реализован подход, который может лечь в основу процедур, тестирующих параллельные SCOOP-программы.

Проект выполнен в рамках стажировки в ETH (Цюрих, Швейцария). Авторы благодарят профессора Б. Мейера (ETH), а также лабораторию Интел-ННГУ «Информационные технологии» (ITLab).

## Список литературы

1. Choi J.-D. and Srinivasan H. Deterministic Replay of Java Multithreaded Applications // IBM T. J. Watson Research Center. – Hawthorne, NY, USA, 10532.

2. Meyer B. Object-Oriented Software Construction, chapter 31, 2nd edition // Prentice Hall, 1997.

3. Morandi B., Bauer S., and Meyer B. SCOOP – a contract-based concurrent object-oriented programming model / Technical report, ETH Zurich and Ludwig-Maximilians-Universitat Munchen, 2009.

4. Nienaltowski P.: Practical framework for contract-based concurrent object-oriented programming / PhD dissertation 17061, Department of Computer Science, ETH Zurich, February. 2007.

5. Leblanc T.J. and Mellor-Crummy J.M. Debugging parallel programs with instant replay // IEEE Transactions on Computers, C-36(4):471-481, April, 1987.

6. Tai K.C., Carver R.H., and Obaid E.E. Debugging concurrent Ada programs by deterministic execution // IEEE transactions on Software Engineering, 17(1):45-63, January 1991.

7. Nikonov A., Rusakov A. Научный отчет по проекту (англ.) – URL: http://se.ethz.ch/projects/andrey_nikonov/report.pdf.

**И.Ю. Ошева, А.А. Ташкинов, В.Е. Шавшуков**

Пермский государственный технический университет

# КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ МЕХАНИЧЕСКОГО ПОВЕДЕНИЯ ПРИЗМАТИЧЕСКИХ ОБРАЗЦОВ ИЗ ПРОСТРАНСТВЕННО-АРМИРОВАННЫХ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ ПРИ СЖАТИИ[1]

Одной из важнейших характеристик любого материала является прочность на сжатие. Известно, что для рассматриваемых композиционных материалов с трехмерно-направленными ортогональными армирующими системами свойства, определенные из испытаний на образцах, не совпадают со свойствами, реализуемыми в конструкциях. В связи с этим актуальным является выявление и изучение факторов, влияющих на результаты измерения прочностных свойств таких материалов на образцах традиционной формы.

Геометрическая модель образца построена на основе размеров реальных образцов, используемых в испытаниях на сжатие. Образец имеет форму прямого параллелепипеда. Армирующий каркас получен переплетением трех семейств нитей, причем каждое семейство образует с двумя другими прямой угол. Все нити представляют собой прямые круговые цилиндры. Нити расположены друг от друга на равном расстоянии. Грани образца параллельны армирующему каркасу. Ось образца параллельна координатной оси *0z,* опорная поверхность параллельна координатной плоскости *x0y*. Армирующий каркас симметричен относительно координатных осей. Геометрическая модель образца представлена на рис.1.

В работе рассматривалось три типа моделей – три задачи (рис. 2):

[1] Работа выполняется при финансовой поддержке грантов Российского фонда фундаментальных исследований РФФИ № 09-08-99117, РФФИ №10-08-96062.

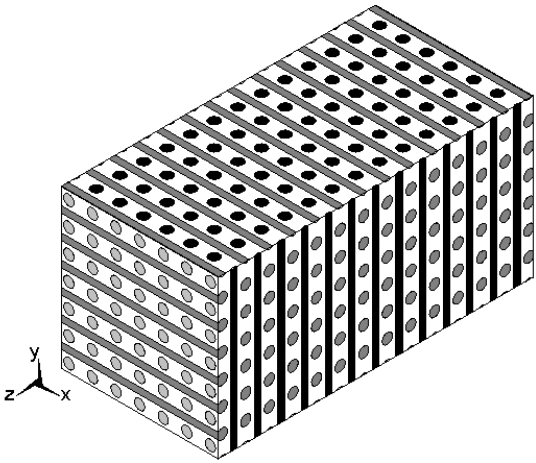



Рис. 1. Твердотельная модель образца

*1. «Идеальная модель»* – рассматривается только как образец, а действие испытательной машины заменяется граничными условиями на гранях образца.

*2. «Модель с учетом трения»* – перемещение на образец передается через моделируемые плиты испытательной машины. Модель позволяет учесть трение, возникающее между плитами и образцом.

*3.* «*Модифицированная модель»* – модель испытания аналогична второй задаче, только торцы образца залиты мягким сплавом.

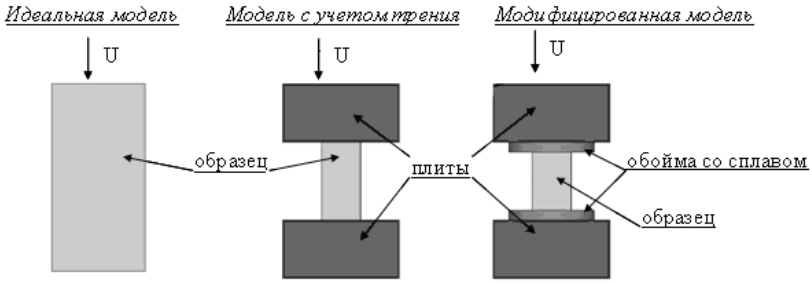



Рис. 2. Модели экспериментов

Анализ механического поведения крупноячеистых пространственно-армированных композитов проводился численно с использованием коммерческой лицензионной версии пакета ANSYS 11.0 на кластере Центра высокопроизводительных вычислительных систем Пермского государственного технического университета. Запуск на счет проводился с использованием удаленного доступа. Среднее количество элементов и узлов геометрических моделей составляет 380 000 и 513 000 соответственно.

**С.В. Панков**

Южный Федеральный университет, г. Ростов-на-Дону

## МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ПРОЦЕССОВ В СИСТЕМАХ С АРХИТЕКТУРОЙ ТИПА ТОР

Рассматривается класс задач, адекватно реализуемых на параллельных распределённых вычислительных системах (кластерах) с архитектурой типа тор. При этом схема взаимодействия процессов в таких реализациях характеризуется высокой степенью регулярности. Многие задачи, реализуемые с помощью технологии распределённого программирования SPMD (Single Program Multiply Data), могут быть отнесены к этому классу. Например, задачи математического моделирования, использующие сеточные (конечно-разностные) вычислительные методы.

Представляется подход к моделированию и анализу поведения таких систем на базе формализма *L*-программ. Этот формализм включает в себя логико-математическую модель параллельных вычислений – *L*-программы (впервые описана в [1]) и средства анализа *L*-программ (разработаны в [2]). Обсуждается свойство нетупиковости систем. Этот подход направлен на автоматизацию анализа рассматриваемых систем. Обеспечивает компактность модели, независимость её вида от числа процессов. Это, в свою очередь, влечёт за собой независимость анализа системы от её размера.

**Основные понятия.** Моделирование распределённых систем осуществляется на уровне абстракции, не учитывающем вычислительные действия процессов и принимающем во внимание только операции посылки и приёма сообщений. В качестве архитектуры вычислительной системы здесь рассматривается прямоугольная решётка, замкнутая сама на себя в тор. Процессы развиваются на вычислительных узлах решётки. Горизонтальные линии связи решётки (с номерами от 0 до $m$-1) образуют продольные кольца тора, а вертикальные линии связи (с номерами от 0 до $n$-1) формируют поперечные кольца тора. Каждый процесс идентифицируется парой координат, образованной номером продольного кольца тора и номером его продольного кольца.

$L$-программы представляют собой систему переходов $S=(Q,T)$, где $Q$ – множество состояний, а $T \subseteq Q \cdot Q$ – отношение перехода на $Q$. Множество состояний $P \subseteq Q$ называется *инвариантом* системы переходов $S$, если справедливо $\forall q',q \in Q$ $((q',q) \in T \ \& \ q' \in P) \Rightarrow q \in P)$. Через $fin[S]$ обозначим множество *заключительных состояний* системы переходов $S$, т.е. $fin[S]=\{q \in Q \mid \forall q' \in Q((q,q') \notin T)\}$.

**Модели регулярных схем взаимодействия процессов в виде *L*-программ.**

**1. *Язык предметной области клеточных L.***

**Сорта:**

*LONG* – множество значений $\{0,1,...,m-1\}$ продольной координаты процесса;

*LATER* – множество значений $\{0,1,...,n-1\}$ поперечной координаты процесса.

**Функции:**

$\_+_m1$: $LONG \to LONG$ – увеличивает значение продольной координаты процесса на 1 по модулю $m$;

$\_-_m1$: $LONG \to LONG$ – уменьшает значение продольной координаты процесса на 1 по модулю $m$ (определяется через функцию $+_m1$, как $i-_m1=(i+m-2)+_m1$, где $i \in LONG$);

$\_+_n1$: $LATER \to LATER$ – увеличивает значение поперечной координаты процесса на 1 по модулю $n$;

$\_-_n1$: $LATER \to LATER$ – уменьшает значение поперечной координаты процесса на 1 по модулю $n$.

В $L$ также будут входить стандартные арифметические операции и отношения.

**Предикаты:**

!: $LONG \times LATER \times LONG \times LATER$ – задаёт для процесса, определяемого первой парой аргументов, запрос на передачу сообщения процессу, задаваемому второй парой аргументов;

?:$LONG \times LATER \times LONG \times LATER$ – задаёт для процесса, определяемого первой парой аргументов, запрос на приём сообщения от процесса, задаваемого второй парой аргументов.

**Переменные:** $i,i'$: $LONG$; $j,j'$: $LATER$**.**

Во всех следующих ниже моделях предполагается, что в начальный момент справедливо $\forall i,j,i',j'\ (\neg?(i,j,i',j') \wedge \neg!(i',j',i,j))$.

**2. *Синхронная модель взаимодействия процессов со всеми своими соседями.***

1) $\$\ i,j,i',j'\ \ \neg!(i,j,i',j') \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee j'=j+_n1 \wedge i'=i) \Rightarrow$

$!(i,j, i',j')$

2) $\$\ i,j,i',j'\ \ \neg?(i,j,i',j') \wedge !(i',j',i,j) \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee$

$j'=j+_n1 \wedge i'=i) \Rightarrow ?(i,j,i',j')$

3) $\$\ i,j,i',j'\ \ ?(i,j,i',j') \wedge !(i',j',i,j) \wedge (i'=i-_m1 \wedge j'=j \vee i'=i+_m1 \wedge j'=j \vee j'=j-_n1 \wedge i'=i \vee$

$j'=j+_n1 \wedge i'=i) \Rightarrow \neg?(i,j,i',j') \wedge \neg!(i',j',i,j)$

Эта $L$-программа состоит из трёх *правил*. Каждое правило имеет левую часть (*условие* в виде $L$-формулы) и правую часть (*действие*, $L$-формула специального вида [2]), которые объединены знаком $\Rightarrow$. Стоящая перед правилами конструкция $\$\ i,j,i',j'$ называется *синхронизатором* по переменным $i,j$, $i',j'$. Первое правило моделирует инициализацию операции посылки сообщения процессом с координатами $(i,j)$ процессу с координатами $(i',j')$. Благодаря синхронизатору $\$\ i,j,i',j'$ это правило срабатывает для всех значений переменных $i,j,i',j'$, для которых истинно его условие. Тем самым моделируется инициализация посылки

сообщения каждым процессом $(i,j)$ каждому из четырёх своих соседей – $(i',j')$. (В реальной системе это соответствует рассылке процессом сообщений своим соседям в любом порядке.) $L$-программа работает по шагам. На первом её шаге может сработать только первое правило, условия остальных правил будут тождественно ложны.

Второе правило может исполниться только после первого и моделирует инициализацию операции приёма сообщения каждым процессом $(i,j)$ от каждого из четырёх своих соседей – $(i',j')$. В реальной системе это соответствует любому порядку приёма сообщений процессом от своих соседей. Третье правило может исполниться только после второго и моделирует завершение всех (благодаря синхронизатору) операций приёма и передачи сообщений. Затем снова исполняется первое правило, и т.д.

**3.** ***Модели взаимодействия процессов со своими соседями в поперечных и продольных кольцах тора.***

1) $\$\, i,j,i' \quad \neg!(i,j,i',j)\wedge(i'=i+_m 1) \Rightarrow !(i,j,\, i',j)$

2) $\$\, i,j,i' \quad \neg?(i,j,i',j)\wedge!(i',j,i,j)\wedge(i'=i-_m 1) \Rightarrow ?(i,j,\, i',j)$

3) $\$\, i,j,i' \quad ?(i,j,i',j)\wedge!(i',j,i,j)\wedge(i'=i-_m 1) \Rightarrow \neg?(i,j,i',j)\wedge\neg!(i',j,i,j)$

Первое правило моделирует инициализацию посылки сообщения процессом с координатами $(i,j)$ своему нижнему соседу с координатами $(i+_m 1,j)$. Синхронизатор $\$\, i,j,i'$ вынуждает осуществить это каждый процесс $(i,j)$. Второе правило моделирует инициализацию процессом $(i,j)$ приёма сообщения от верхнего соседа, процесса $(i-_m 1,j)$. Синхронизатор $\$\, i,j,i'$ вынуждает это проделать каждый процесс $(i,j)$. Третье правило моделирует завершение всех операций приёма и передачи в поперечных кольцах тора. Как и в предыдущей модели, правила могут срабатывать циклически, вслед друг за другом (первое, второе, третье, снова первое и т.д.). Аналогичным образом моделируется взаимодействие процессов со своими соседями в продольных кольцах тора:

1) $\$\, i,j,j' \quad \neg!(i,j,i,j')\wedge(j'=j+_m 1) \Rightarrow !(i,j,\, i,j')$

2) $\$\, i,j,j' \quad \neg?(i,j,i,j')\wedge!(i,j',i,j)\wedge(j'=j-_m 1) \Rightarrow ?(i,j,\, i,j')$

3) $\$\, i,j,j' \quad ?(i,j,i,j')\wedge!(i,j',i,j)\wedge(j'=j-_m 1) \Rightarrow \neg?(i,j,i,j')\wedge\neg\, !(i,j',i,j)$

В случаях поперечных и продольных колец процесс передаёт сообщение следующему в кольце процессу, а принимает от предыдущего. Если объединить две последние $L$-программы, будет моделироваться взаимодействие и в поперечных, и в продольных кольцах, осуществляемые относительно друг друга асинхронно. Выбор правил, срабатываемых на одном шаге $L$-программы, из этих двух групп правил происходит недетерминированно.

**4.** ***Асинхронная модель взаимодействия процессов со всеми своими соседями.***

1) $\neg!(i,j,\ i\text{-}_m1,j)\wedge\neg!(i,j,\ i+_m1,j)\wedge\neg!(i,j,\ i,j\text{-}_m1)\wedge\neg!(i,j,\ i,j+_m1) \Rightarrow$
$i,j,\ i\text{-}_m1,j)\wedge!(i,j,\ i+_m1,j)\wedge!(i,j,\ i,j\text{-}_m1)\wedge!(i,j,\ i,j+_m1)$

2) $\neg?(i,j,\ i\text{-}_m1,j)\wedge\neg?(i,j,\ i+_m1,j)\wedge\neg?(i,j,\ i,j\text{-}_m1)\wedge\neg?(i,j,\ i,j+_m1)\wedge$
$!(i\text{-}_m1,j,i,j)\wedge!(i+_m1,j,i,j)\wedge!(i,j\text{-}_m1,i,j)\wedge!(i,j+_m1,i,j) \Rightarrow$
$i,j,\ i\text{-}_m1,j)\wedge?(i,j,\ i+_m1,j)\wedge?(i,j,\ i,j\text{-}_m1)\wedge?(i,j,\ i,j+_m1)$

3) $\$\ i',j'\ \ ?(i,j,i',j')\wedge!(i',j',i,j)\wedge(i'=i\text{-}_m1\wedge j'=j\vee i'=i+_m1\wedge j'=j\vee j'=j\text{-}_n1\wedge i'=i\vee j'=j+_n1\wedge i'=i) \Rightarrow$

Первое правило на одном шаге $L$-программы может исполниться для произвольного множества процессов $(i,j)$, для которых не инициализировались операции посылки сообщений всем своим четырём соседям. В результате исполнения правила такая инициализация моделируется. Второе правило на одном шаге $L$-программы может исполниться для произвольного множества процессов $(i,j)$, для которых не инициализировались операции приёма сообщений от всех четырёх соседей, а операции посылки сообщений от них процессу $(i,j)$ уже инициализированы. В результате исполнения правила моделируется инициализация приёма сообщений процессом $(i,j)$ от всех своих соседей. Третье правило моделирует завершение операций приёма и передачи для всех (благодаря синхронизатору $\$\ i',j'$) соседей процессов $(i,j)$, для которых такие операции были инициализированы. Число таких процессов $(i,j)$ выбирается недетерминированно. Данная модель не предполагает глобальной синхронизации процессов, как в модели 1. Она учитывает различную скорость функционирования процессов.

**Анализ нетупиковости.** *Нетупиковым* состоянием системы $S$ называются её заключительное состояние из $fin[S]$, в котором справедливо $\forall i,j,i',j' (\neg?(i,j,i',j') \wedge \neg!(i',j',i,j))$, т.е. ни один из процессов не имеет незавершённых операций посылки и приёма сообщений. Пусть $D \subseteq Q$ – это множество нетупиковых состояний. Система переходов $S$ относительно заданного множества начальных состояний $I \subseteq Q$, если существует такое множество состояний $W \subseteq Q$, что справедливо следующее: 1) $I \subseteq W$; 2) $\forall q',q \in W ((q',q) \in T \ \& \ q' \in W) \Rightarrow q \in W)$. т.е. $W$ – инвариант; 3) $W \cap fin[S] \subseteq D$.

Алгоритмы из [2] позволяют получить логические формулы, выражающие в языке $L$ отношение перехода $T$, множество $fin[S]$ для моделей в виде $L$-программ. Если множества $W$ и $D$ также выражаются $L$-формулами, то анализ нетупиковости моделируемых систем сводится к доказательству истинности формул логики предикатов первого порядка. Для систем с регулярными схемами взаимодействия удаётся находить компактные формулы-инварианты, независящие от числа процессов, что в частности, показывалось в [3]. Всё это обеспечивает независимость анализа нетупиковости от размера системы.

## 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. Научный сервис в сети ИНТЕРНЕТ: материалы Всерос. науч. конф. Новороссийск, 19–24 сент. 2007 г. – М.: Изд-во МГУ, 2007. – С. 85–88.

**В.В. Пекунов**

Ивановский государственный энергетический университет

# ТЕОРИЯ ОБЪЕКТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ ПОСЛЕДОВАТЕЛЬНЫХ И ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ. ОСНОВНЫЕ СЛЕДСТВИЯ И ПРИЛОЖЕНИЯ В ПРОГРАММИРОВАНИИ, МОДЕЛИРОВАНИИ И ПОРОЖДЕНИИ ПРОГРАММ

В данной работе на основе самых общих предположений выводится теория объектно-событийных моделей (ОСМ) последовательных и параллельных процессов и определяются ее наиболее перспективные приложения.

**Лемма А1**. В процессе исполнения любой последовательной или параллельной программы любая вычислительная система (машина) осуществляет серию переходов между звеньями обработки, отображаемую трассой исполнения. Одноименным, однозначно идентифицируемым по имени элементам трассы исполнения соответствует факт прохождения системы через одно и то же звено обработки. Каждому этапу трассы соответствует один или несколько (обрабатываемых параллельно на одном и том же интервале времени) элементов. Любое звено реализует параметризованный алгоритм, обрабатывающий внутренние переменные своего состояния и/или внешние данные. Его исполнение определяется комплексом: а) глобальных данных вычислительной системы; б) параметров вызова, которые могут быть реализованы в рамках глобальных данных; в) значений внутренних переменных. Иных побочных эффектов нет.

**Доказательство**. Любая программа, реализующая вычислимый по Тьюрингу алгоритм, может быть реализована с помощью машины Тьюринга (МТ) или комплекса таких машин. Для МТ всегда можно построить трассу исполнения, которая может быть однозначно отображена в трассу любой иной последовательной вычислительной системы. Элементами трассы МТ будут имена состояний, однозначно соответствующие уникальным звеньям обработки. Для комплекса МТ также можно построить

общую трассу, комбинируя их частные трассы по принципу одновременности исполнения.

Алгоритм звена обработки в предельном случае соответствует фрагменту программы переходов из одного состояния МТ. Его работа зависит от глобального контекста (ленты) и внутреннего состояния (положения головки).

**Утверждение У1**. Модель процесса исполнения, адекватно отражающая логику переходов исходной системы между элементами трассы исполнения, каждый из которых подразумевает параметризованный вызов соответствующего алгоритма, осуществляющего преобразования внутренних и внешних данных, тождественные преобразованиям, выполняемым исходной системой при прохождении через указанные элементы трассы исполнения, является моделью исходной программы.

**Доказательство**. Вытекает из А1.

**Утверждение У2**. Назовем факт прохождения программы через один или несколько этапов трассы ее исполнения *событием*. Процесс обработки события означает исполнение этих и только этих этапов. В контексте календаря событиям соответствуют непересекающиеся интервалы времени обработки. Между этими событиями существует строгий порядок следования, соответствующий порядку прохождения трассы.

**Доказательство**. Непересекаемость интервалов следует из А1, поскольку этапы трассы исполняются последовательно. Строгий порядок событий следует из отношения строгого порядка интервалов времени.

**Следствие У2.1**. Априорно считаем параллельным исполнение этапов трассы, соответствующих одному событию. Для установления последовательности их исполнения необходимы средства, устанавливающие очередность прохождения.

**Лемма Л1**. Параметризованный алгоритм А, вызываемый при прохождении программы через одноименные элементы трассы исполнения, обладающий внутренним состоянием, реализуется объектом некоего класса С. Каждому факту прохождения соответствует событие, приводящее к вызову отдельного метода класса С.

**Доказательство.**

1. Возможность сопоставления события отдельному методу следует из А1, У2 и определения метода в рамках объектно ориентированного подхода. Метод интерпретируется как фрагмент алгоритма звена обработки, активный в контексте события.

2. Если одному или нескольким различным событиям соответствуют различные алгоритмы обработки, то их реализация в отдельных методах строго логична. В случае единого алгоритма достаточно прибегнуть к копированию методов.

3. Внутреннее состояние алгоритма А и производных от него алгоритмов представляется полями класса С.

4. Параметризация вызовов исходного алгоритма А, таким образом сводится к параметризации эквивалентных им по действию вызовов методов класса С.

**Базовая теорема ОСМ (ТМ1).**

А. Модель исходной последовательной или параллельной программы, может быть представлена графово-событийной схемой. Узлами графа являются объекты, реагирующие на события параметризованными вызовами методов, модифицирующих состояние объекта и глобальные данные. События происходят в порядке, определяемом их календарем.

Б. Если в ответ на одно событие вызываются методы нескольких или даже всех объектов, то необходимо специфицировать порядок их последовательного или параллельного исполнения. Если для двух объектов определено отношение строгого порядка следования, их методы исполняются последовательно. Иначе параллельно. Отношение задано дугами графа.

В. Граф должен быть связным и ациклическим.

**Доказательство.**

А. Следует из У1, У2, Л1

Б. Следует из самого утверждения.

В. Любая программа имеет единственные начальную и конечную точки. Реагировать на одно и то же событие потенциально могут все узлы. Любой промежуточный узел находится

в отношении строгого следования по отношению, по меньшей мере, к этим двум точкам. Это антирефлексивное, транзитивное и антисимметричное отношение. Следовательно, граф является связным и сетевым, то есть ациклическим.

**Лемма Л2.** Узел, имеющий несколько исходящих дуг в узлы, инцидентные по входам только ему, в зависимости от событийной схемы является точкой принятия решения о переходе последовательной программы и/или точкой порождения параллельных ветвей.

**Доказательство**. Если следующие узлы некоторых исходящих ветвей предусматривают реакцию на одно и то же событие, их исполнение на данном участке стартует параллельно (это следует из теоремы ТМ1, учитывая, что по условию леммы данные узлы не инцидентны). Если на некоторое событие реагирует узел лишь одной из ветвей, то ее исполнение на данном участке является исключительным, что эквивалентно исполнению конструкции принятия решения.

**Лемма Л3.** Узел, имеющий несколько входящих дуг в узлы, инцидентные по выходам только ему, в зависимости от событийной схемы является конечной точкой условной конструкции последовательной программы и/или точкой слияния параллельных ветвей.

**Доказательство**. Аналогично доказательству Л2.

**Теорема об интерпретации связей (ТМ2)**. В ходе реакции на одно событие методы объектов, соответствующих узлам, исполняются в порядке, соответствующем идеологии сетевого графика работ, если отсутствие реакции объекта на событие смоделировать реакцией, состоящей в вызове пустого или предопределенного метода.

**Доказательство**. При реакции на одно событие узлы модели активизируются по входам и активизируют выходы в соответствии с И-логикой, это следует из ТМ1, Л2, Л3 с учетом сделанного допущения о моделировании отсутствия реакции. Получаем логику интерпретации сетевого графика работ.

**Теорема о планировании событий (ТМ3)**. Модель программы, трасса исполнения которой заранее неизвестна или меняется в зависимости от внешних данных, реализуется путем генерации событий и подписки на события в методах объектов-узлов модели. Начальный календарь содержит как минимум одно событие, на которое реагирует как минимум один объект.

Каждый объект А имеет возможность включить в календарь новое событие. Планирование события является атомарной транзакцией.

Каждый объект А может подписать любой метод любого иного объекта В либо на любое из еще не произошедших событий, либо на текущее, но тогда и только тогда, когда объект В следует за А. Подписка – атомарная операция по отношению к В.

На любое событие объект реагирует только в одном методе. Это либо метод, указанный при проектировании модели, либо последний из указанных по подписке, либо пустой/предопределенный метод.

**Доказательство**. Программируемый переход от узла А к узлу В в предельном случае может быть реализован путем условного планирования узлом А такого события, которое может быть в дальнейшем обработано исключительно узлом В. С этой целью фактически вводятся понятия подписки на событие по умолчанию и программируемой подписки. Чтобы модель программы начала работу, необходимо как минимум одно начальное событие, которое обрабатывается хотя бы одним узлом. Атомарность операции планирования события является необходимым условием отсутствия коллизий в ходе исполнения параллельного процесса.

**Следствие**. Если доступ к календарю открыт из любого объекта, календарь должен относиться к глобальным данным.

**Теорема ТМ4.** Событие удаляется из календаря сразу же после его наступления.

Теорема будет доказываться совместно с теоремой ТП1.

**Утверждение У3**. Объектно-событийная модель процесса/программы/алгоритма должна удовлетворять определениям, данным теоремами ТМ1, ТМ2, ТМ3, ТМ4.

**Теорема о самодостаточности ОСМ (ТМ5)**. Объектно-событийная модель является самодостаточной в смысле реализации.

**Доказательство**. Любой метод любого класса модели является неким алгоритмом, который может быть реализован предельной (атомарной) или непредельной (декомпозируемой) объектно-событийной моделью, вложенной в общую.

**Теорема об абстрактной предельной ОСМ (ТП1)**. Предельной абстрактной (атомарной) объектно-событийной моделью назовем модель, включающую единственный узел и обладающую календарем, вмещающим не более одного события. Такая модель равномощна машине Тьюринга (МТ).

**Доказательство**. Определим соответствие. Множество команд вида $q_i a_i \rightarrow q_j a_j d_j$ любой программы ($q_i$ и $q_j$ – исходное и следующее состояние, $a_i$ и $a_j$ – символы алфавита в текущей ячейке ленты до и после исполнения команды, $d_j$ – направление перемещения головки), исполняемой МТ, разобьем на группы относительно $q_i$. Если рассматривать переход в $q_i$ как событие с идентификатором $q_i$ в текущем контексте, то указанные группы команд являются методами некоего класса, реагирующими на такое событие выполнением элементарных действий по замене $a_i$ на $a_j$ в текущей ячейке, планированию нового события с идентификатором $q_j$ и перемещению головки в направлении $d_j$. Лента МТ рассматривается как глобальные данные, положение головки – внутреннее состояние (поле класса). МТ всегда обладает начальным состоянием, эквивалентным начальному событию в календаре. Все требования У3 для МТ удовлетворены.

Поскольку идентификация событий в календаре должна быть уникальной, а идентификаторами являются неоднократно встречающиеся состояния МТ, будем хранить в календаре только одно следующее событие. Этого можно добиться, удаляя из календаря текущее событие сразу же после его наступления.

Таким образом, доказана не только текущая теорема, но и теорема ТМ4.

**Теорема о реальной предельной ОСМ (ТП2).** Предельной реальной (неатомарной) ОСМ назовем процедуру с планированием повторного входа [0]. Это модель, состоящая из одного узла-объекта, обладающего единственным методом реакции на все события. Планирование события ограничивается планированием в начало и конец календаря. Такая модель в пределе позволяет реализовать любой алгоритм.

**Доказательство.** Описательные возможности такой ОСМ покрывают все возможности предельной атомарной ОСМ, эквивалентной машине Тьюринга. Следовательно, предельная неатомарная ОСМ также позволяет описать любой алгоритм. В такой одноузловой системе календарь событий может являться атрибутом единственного метода обработки. Целесообразно рассматривать его как обычную процедуру. Календарь отобразим в план этапов обработки данных. Процедура содержит серию команд, реализующих этапы обработки (реакцию на события), в том числе планирование новых этапов в начало или в конец плана. Согласно ТМ3 в начальном календаре должно присутствовать хотя бы одно событие, следовательно процедура обязана иметь по меньшей мере один начальный этап обработки в плане. Каждому этапу (вызову процедуры) соответствует комплекс значений фиксированного множества элементов данных – параметров процедуры. Такого рода процедуры назовем *процедурами с планированием повторного входа*.

**Следствие.** Процедура с планированием повторного входа способна реализовать последовательный алгоритм, декомпозируемый в серию схожих или идентичных по структуре и алгоритму решения подзадач с динамическим планированием последовательности их решения в соответствии со стратегией стека, дека или очереди.

**Теорема о последовательных и параллельных процессах (ТПР1)**. В рамках объектно-событийной модели некоторой программы любой последовательный или параллельный процесс может быть порожден планированием события и описан

фрагментом модели, в котором определена реакция на данное событие. Для описания параллельных процессов необходимо представить их порождение, слияние, синхронизацию, взаимное исключение и обмен данными.

**Доказательство.**

1. Реализуемость и механизмы *порождения и слияния* параллельных ветвей следуют из Л2, Л3. Целесообразно планировать соответствующие события порождения A1 и слияния A2. Порождаемые процессы планируют прочие события, необходимые для описания логики их работы, на промежутке $]A1,A2[$.

*Точки барьерной синхронизации* можно смоделировать m узлами, каждому из которых инцидентны по входам и по выходам дуги k параллельных ветвей, $1 \le m \le k$.

*Взаимное исключение* k параллельных ветвей (механизм критической секции) может быть смоделировано, например, разрывом точки синхронизации (при m = 1) на два узла – слияния и порождения, между которыми следует фрагмент модели, реализующий алгоритм в пределах критической секции. Планируются необходимые события слияния A1 и порождения A2. Каждая из сливаемых ветвей планирует одно событие прохождения критической секции на промежутке $]A1,A2[$. Такая схема гарантирует требуемую логику работы.

*Обмен данными между процессами* может быть реализован через глобальные данные с использованием взаимного исключения ветвей при доступе.

2. Произвольный последовательный процесс может быть запущен планированием нового события, обрабатываемого или отдельным узлом, методы которого описываются некоторыми, например, предельными ОСМ, реализующими произвольные алгоритмы согласно ТП1, или комплексом таких узлов, не порождающих параллельных по событиям ветвей. Согласно Л2 и Л3 в таком комплексе можно описать условное исполнение. Согласно ТМ3 можно представить переход, в том числе замыкающий цикл.

**Утверждение У4**. Текст программы также может являться следствием работы иной программы (системы порождения). Его отдельные фрагменты имеют однозначное соответствие результатам исполнения отдельных звеньев системы порождения и могут быть отражены в элементы трассы ее исполнения.

**Следствие**. Одному звену системы порождения программ могут соответствовать различные фрагменты генерируемой программы.

**Лемма Л4**. Фрагменты программы, решающей задачу, генерируемые одним и тем же звеном, являются алгоритмической реализацией некоей сущности (объекта/действия/отношения).

**Доказательство**. Звено работает по заданному алгоритму или совокупности алгоритмов, которые согласно Л1 представляют собой методы некоего класса. Класс по определению инкапсулирует данные и методы обработки, характерные для некоего класса объектов реального мира или действия или отношения. Эти сущности могут явно присутствовать в постановке задачи либо логически выводятся из нее.

**Следствие**. В таком контексте события эквивалентны этапам генерации кода или, что более правомерно, этапам жизненного цикла/проявлений указанной сущности.

**Теорема о порождении программ (ТМ6)**. Среда порождения программ может использовать объектно-событийные модели для описания и программирования решения некоторой задачи [0]. Объекты–узлы модели являются экземплярами классов предметной области задачи и включаются в модель, либо исходя из постановки задачи, сформулированной в терминах предметной области, либо исходя из логических следствий из этой постановки в соответствии с некоторой системой знаний о предметной области. Модель способна породить решающую задачу программу.

**Доказательство**. Следует из У4, ТМ1, Л4.

Сформулированы основные положения теории ОСМ. Доказан ряд утверждений и теорем, в том числе о предельных ОСМ. Даны теоретические выкладки, обосновывающие приме-

нение ОСМ и производных от них формализмов в программировании [0], порождении [0] и моделировании обычных и параллельных программ.

### Список литературы

1. Пекунов В.В. Процедуры с планированием повторного входа в языках высокого уровня при традиционном и параллельном программировании // Информационные технологии.– 2009. – № 8. – С. 63–67.

2. Пекунов В.В. Автоматизация параллельного программирования при моделировании многофазных сред. Оптимальное распараллеливание // Автоматика и телемеханика.– 2008. – № 7. – С. 170–180.

**А.А. Пепеляев**

Пермский государственный технический университет

## МОДЕЛИРОВАНИЕ ВЗРЫВА БЫТОВОГО ГАЗА В КИРПИЧНОМ ЗДАНИИ

Аварии зданий, вызванные взрывами бытового газа, происходят регулярно. Отказы отдельных элементов конструкций зачастую способны спровоцировать прогрессирующее разрушение здания. Данная проблема в основном затрагивает существующие газифицированные здания. Такие ситуации, как несанкционированное подключение к системе газоснабжения, халатность при пользовании газом и газовыми приборами в быту, не представляется возможным контролировать или регулировать их предотвращение.

При возникновении подобного рода ситуации внутри помещений происходит дефлаграционный взрыв – быстрое горение газовоздушной смеси, концентрация горючего в которой находится между нижним и верхним концентрационными пределами воспламенения. Взрывоопасное облако формируется

с учетом множества факторов внутри здания, таких как связь помещения, в котором происходит утечка газа, с соседними помещениями и атмосферой, а также наличие легкосбрасываемых преград, условий вентиляции помещений.

В зависимости от силы взрыва конструкции здания получают повреждения различного рода. Отказы отдельных элементов конструкций зачастую способны спровоцировать прогрессирующее разрушение здания (либо какой-то его части). На сегодняшний день проблема расчета на прогрессирующее разрушение формулируется следующим образом: конструктивная схема здания должна обеспечивать его прочность и устойчивость в случае локального разрушения несущих конструкций как минимум на время, необходимое для эвакуации людей.

Однако прежде чем рассчитывать здание на прогрессирующее разрушение, необходимо определить величину интенсивности взрывной нагрузки. На сегодняшний день существует большое количество нормативных, методических документов, позволяющих производить расчёты взрывов опасных веществ и оценивать воздействие ударной волны на здания и сооружения, расположенные в зоне распространения ударной волны взрыва. Но все нормативные и методические документы, представленные на российском рынке и в других странах, различаются не только в оценке воздействия поражающих факторов на здания, но и в расчётных показателях избыточного давления. Так, в Англии рекомендовано расчетный перепад давления при проектировании зданий принимать равным 34,5 кПа, что вызывает много споров и возражений.

При дефлаграционном взрыве реализуется принцип квазистатичности избыточного давления, который заключается в независимости взрывной нагрузки от пространственной координаты. Другими словами, давление, действующее в данный момент времени на любой конструктивный элемент ограждения (стены, потолок, пол, окна, двери и т.д.), одинаково во всех точках помещения. Величина избыточного давления для любого момента времени определяется темпом роста давления, вызванного вы-

делением продуктов сгорания на фронте пламени, и темпом снижения давления вследствие истечения газа (свежей смеси или продуктов сгорания) через открытый проём.

Математическая модель горения газа в воздухе основана на уравнениях, которые описаны в модели слабосжимаемой жидкости: Навье-Стокса, неразрывности, состояния идеального газа, закона сохранения энергии, краевых и начальных условий.

Для численной реализации задачи использовался метод конечных объемов и программный комплекс Flow Vision, решающий задачи газовой динамики и сопряжённые задачи взаимодействия потока с деформируемым телом.

Расчетные области для подобных задач включают в себя достаточно много ячеек, и процесс дискретизации решения осуществляется медленно. К тому же при решении данной задачи проводился многофакторный эксперимент, что дополнительно во много раз увеличивало потребности во времени. Было принято решение использовать параллельные вычисления при помощи вычислительного кластера, установленного в ПГТУ, что значительно сократило продолжительность вычислений.

Опираясь на результаты обследования реального кирпичного здания, в котором произошел взрыв бытового газа, с целью оценки адекватности используемых математических моделей процесса взрыва и изучения влияния различных параметров на величину интенсивности взрывной нагрузки, в программе Flow Vision методом конечных объемов мы решали задачу определения интенсивности взрывной нагрузки. В результате расчета была определена величина давления на стенках модели кухни, которая получилась равной 4–6 кПа.

Это значение давления использовалось при решении следующей задачи – оценки несущей способности конструкций здания в программном комплексе ANSYS. Решалась пространственная нелинейная краевая задача расчета напряженно-деформированного состояния реального здания с несущими кирпичными стенами и сборными перекрытиями методом конечных элементов. При решении механическое поведение мате-

риалов кирпичной кладки и железобетона моделировалось упруго-хрупкой моделью с учетом возможности структурного разрушения.

Наглядное сравнение картины трещин, полученных в ANSYS, с фактическим расположением повреждений позволяет говорить о достаточной адекватности модели. Данную модель в дальнейшем можно использовать для изучения влияния различных параметров на силу и место взрыва, что позволит с определенной долей вероятности судить о «живучести» зданий с разными конструктивными схемами.

**В.В. Пересветов**

Вычислительный центр Дальневосточного отделения РАН, г. Хабаровск

## ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ РОЯ ЧАСТИЦ В РЕШЕНИИ ОБРАТНЫХ ЗАДАЧ ЭЛЕКТРОМАГНИТНЫХ ЗОНДИРОВАНИЙ

В докладе представлены параллельные алгоритмы метода роя частиц – Particle Swarm Optimization (PSO) в численном решении коэффициентных обратных задач электромагнитных зондирований слоистых сред гармоническими дипольными источниками. Обратные задачи электромагнитных зондирований слоистых сред относятся к основным задачам структурной геофизики [1]. Для решения таких задач можно применять алгоритмы глобальной оптимизации, позволяющие находить глобальные экстремумы многоэкстремальных негладких функций невязки решений прямых задач с экспериментальными или эталонными значениями электромагнитных полей в заданной области. В решении обратных электромагнитных задач с помощью методов глобальной оптимизации применяются генетические алгоритмы [2–3], метод роя частиц [4] и др. В настоящем докладе представлены параллельные версии метода роя частиц, ранее в [3] применялись генетические алгоритмы решения рассматриваемых здесь задач.

**Постановка задачи.** Обратная задача электромагнитных зондирований формулируется как поиск экстремума функции невязки:

$$\inf_{k\in X_c}\sum_i\left(\left|\mathbf{E}_i^{\exp}-\mathbf{E}_i\right|^2+\chi^2\left|\mathbf{H}_i^{\exp}-\mathbf{H}_i\right|^2\right), \qquad (1)$$

где верхний индекс exp указывает на экспериментальные значения векторов электромагнитного поля, $i$ – индекс точки на заданном профиле, $k$ – решение обратной задачи (материальные и геометрические параметры среды), $X_c$ – компактное множество возможных решений. Коэффициент $\chi$ необходим для масштабирования значений вариаций магнитного поля. Источниками поля могут быть гармонические электрические и магнитные дипольные источники. Рассматриваемый здесь подход в общем случае предполагает возможность учета конечного числа дипольных источников.

Электромагнитное поле в плоскослоистой среде с неоднородностями представляется в виде суммы первичного и вторичного полей: $\mathbf{E}=\mathbf{E}^p+\mathbf{E}^s$, $\mathbf{H}=\mathbf{H}^p+\mathbf{H}^s$, где $\mathbf{E}^p,\mathbf{H}^p$ – первичное поле заданных источников в плоскослоистой среде; $\mathbf{E}^s,\mathbf{H}^s$ – вторичное поле, обусловленное присутствием неоднородного включения. Таким образом, прямая задача нахождения электромагнитного поля разбивается на две задачи. Первая прямая задача – найти первичное поле заданных источников в плоскослоистой среде. Решение этой задачи основано на фундаментальных решениях системы Максвелла для горизонтально-слоистых сред и методов интегральных преобразований [5]. Вторая прямая задача – задача нахождения вторичного трехмерного электромагнитного поля, в ней используется полученное решение первой задачи.

**Метод роя частиц.** При выборе конкретного варианта метода роя частиц нужно учитывать, что целевая функция (ЦФ) (1) может иметь не только несколько экстремумов, сравнимых с глобальным, но и близко расположенные друг другу неболь-

шие экстремумы, связанные как с шумами экспериментальных данных, так и погрешностями численных расчетов в решении прямых электромагнитных задач. Для решения задачи глобальной оптимизации в настоящей работе используется метод роя частиц [6], коэффициенты которого удовлетворяют условию сходимости. Вычислительные формулы инерционной модели метода роя частиц имеют вид

$$\begin{aligned} v_{ij}^{(k+1)} &= c_1 v_{ij}^{(k)} + U[0,c_2](p_{ij}^{(k)} - x_{ij}^{(k)}) + U[0,c_3](b_j^{(k)} - x_{ij}^{(k)}), \\ x_{ij}^{(k+1)} &= x_{ij}^{(k)} + v_{ij}^{(k+1)}, \end{aligned} \qquad (2)$$

где $x_{ij}^{(k)}$ и $v_{ij}^{(k)}$ – компоненты векторов позиции и скорости частиц; $i$ – номер частицы, $i = 1,\ldots, M$, $M$ – число частиц в популяции; $j$ – номер компоненты векторной переменной в пространстве размерности $n$; $k$ – номер шага итерационного процесса сходимости решения задачи оптимизации, $c_1$ – инерционный коэффициент; $c_2$ – коэффициент ускорения индивидуальной сходимости частиц; $c_3$ – коэффициент ускорения коллективной сходимости частиц; $U[0,a]$ – случайное число, сгенерированное по равномерному закону распределения в интервале $[0, a]$; $p_{ij}^{(k)}$ – компонента вектора лучшей позиции частицы; $b_j^{(k)}$ – компонента вектора лучшей позиции из всех частиц. Вычислительные формулы $p_{ij}^{(k)}$ и $b_j^{(k)}$ имеют вид

$$p_{ij}^{(k)} = \begin{cases} x_{ij}^{(k)}, \text{если } F(\mathbf{x}_i^{(k)}) < F(\mathbf{p}_i^{(k-1)}), \\ p_{ij}^{(k-1)}, \text{ иначе,} \end{cases}$$

$$b_j^{(k)} = p_{mj}^{(k)}, \; F(\mathbf{p}_m^{(k)}) = \min_i F(\mathbf{p}_i^{(k)}),$$

где $\mathbf{x}_i^{(k)}$ – вектор положения частицы с номером $i$, $\mathbf{p}_i^{(k)}$ – вектор ее лучшей позиции. В численных расчетах использовались значения коэффициентов $c_1 = 0{,}729$, $c_2 = c_3 = 1{,}49445$, которые удовлетворяют условию сходимости метода сужения роя частиц [6].

**Параллельные алгоритмы.** В настоящей работе представлены и сравниваются три параллельных алгоритма решения

рассматриваемой задачи методом роя частиц. В алгоритме 1 используется gbest-топология, в нем параллельные вычисления ЦФ позиций частиц выполняются на каждом шаге итерационного процесса. Передачи сообщений осуществляются с помощью коллективных процедур MPI_SCATTER и MPI_GATHER. При этом объем передаваемых данных незначителен: координаты частиц и значения ЦФ. Этот алгоритм эффективно распараллеливается в случае рассматриваемых здесь электромагнитных задач и других ЦФ, которые требуют большого объема вычислительных операций.

В алгоритме 2 каждый процесс создает свою популяцию частиц и далее вычисляет позиции частиц на каждом шаге итерационного процесса. Такие алгоритмы имеют важное преимущество по сравнению с алгоритмами gbest-типа: в случае многоэкстремальной ЦФ в них существенно уменьшается возможность преждевременной сходимости к локальному экстремуму. Через заданное число итераций на каждом процессе находятся частицы, имеющие позиции с наименьшим и наибольшим значениями ЦФ. Координаты и значение ЦФ лучшей частицы передается по топологии кольца соседнему процессу, а принятые от другого процесса параметры его лучшей частицы заменяют параметры худшей частицы в текущем процессе, но только в том случае, если ЦФ принятой частицы меньше. Сообщения MPI в этом алгоритме выполняются с помощью двухточечных операций MPI_ISEND, MPI_IRECV. Кроме предотвращения преждевременной сходимости этот алгоритм позволяет получить ускорение по сравнению со скалярным вариантом алгоритма 1, так как в нем можно использовать меньшее число частиц в популяциях (см. далее результаты численных экспериментов).

Алгоритм 3 отличается от алгоритма 2 тем, что на последнем шаге итерационного процесса дополнительно находится экстремум методом прямого поиска Нелдера-Мида [7] с использованием полученного методом роя частиц приближения.

Во всех трех алгоритмах начальные популяции создавались случайно во всей области определения. Для предотвраще-

ния замедления скорости итерационного процесса у границ области определения осуществлялась коррекция значений ЦФ на этих границах до большого значения.

**Результаты вычислительных экспериментов.** Проведены испытания описанных выше параллельных алгоритмов для следующих тестовых функций оптимизации: $\sum_{j=1}^{n} x_j^2$ (обозначена $F_{sph}$), $F_{Zah}$ – функция Захарова, $F_{Ros}$ – функция Розенброка (Rosenbrock), $F_{Ras}$ – функции Растригина, $F_W$ – функция Веерштрасса (Weirstrass), $F_{Ac}$ – функция Аклея (Ackley), $F_{Gr}$ – функция Гриванка (Griewank).

В настоящей работе приведены результаты решения обратной задачи зондирования трехслойной среды. Слоистые среды достаточно часто встречаются на практике, потому алгоритмы решения обратной задачи для них представляют значительный интерес. Первичное поле находилось с помощью программы вычисления электромагнитного поля произвольно ориентированного дипольного источника в слоистой среде, алгоритмы которой описаны в [5], авторами алгоритмов и программы являются С.И. Смагин и В.Н. Мазалов При решении обратной коэффициентной задачи неизвестными величинами являются следующие параметры среды: $\rho_1$ – удельное сопротивление верхнего полупространства, $\rho_2$ – удельное сопротивление промежуточного слоя, $\rho_3$ – удельное сопротивление нижнего полупространства, $z_1$ – граница раздела между верхним полупространством и промежуточным слоем, $z_2$ – граница раздела между промежуточным слоем и нижним полупространством. ЦФ функции невязки задачи электромагнитного зондирования (обозначена $F$) вычислялась через значения поля на вертикальном профиле. Диэлектрическая проницаемость всюду $\varepsilon = 9\varepsilon_0$. В представленных здесь результатах расчетов электромагнитного поля использовалось только одно значение частоты излучения 0,2 МГц вертикального магнитного диполя. Подробнее рассматриваемая электромагнитная модель описана в [3].

Приведенные ниже результаты расчетов усреднены по 300 запускам для тестовых функций и 30 – в случае электромагнитных задач. Число MPI-процессов равно 6. В алгоритме 2 обмены между процессами осуществлялись через две итерации.

В табл. 1 представлены средние значения относительной погрешности $\delta$ решения на основе алгоритмов 1 и 2 на шаге $k = 30$ итерационного процесса.

Таблица 1

**Средняя погрешность нахождения решения на шаге $k = 30$**

| Алг. | $F_{sph}$ $n=9$ $M=8$ | $F_{Zah}$ $n=9$ $M=8$ | $F_{Ros}$ $n=3$ $M=36$ | $F_{Ras}$ $n=2$ $M=36$ | $F_{Ras}$ $n=4$ $M=60$ | $F_W$ $n=9$ $M=36$ | $F_{Ac}$ $n=9$ $M=12$ | $F_{Gr}$ $n=2$ $M=24$ | $F$ $n=2$ $M=8$ | $F$ $n=4$ $M=12$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,13 | 0,48 | 0,076 | 0,01 | 0,12 | 0,16 | 0,096 | 0,037 | 0,045 | 0,15 |
| 2 | 0,055 | 0,23 | 0,035 | $2{,}5\times10^{-4}$ | 0,068 | 0,056 | 0,051 | 0,022 | $3{,}8\times10^{-4}$ | 0,031 |

В табл. 2 представлены средние значения числа шагов $k$, которые потребовались для достижения заданной относительной точности решения $\delta = 0{,}001$ с помощью алгоритмов 1 и 2. По этим данным можно оценить и сравнить ускорение параллельного решения. Алгоритм 1 для $F$ показал линейное ускорение при увеличении числа MPI-процессов (в проведенных расчетах – в 6 раз). Алгоритм 2 для всех типов ЦФ дает ускорение, приблизительно равное $k_1/k_2$ относительно скалярного варианта алгоритма 1.

Таблица 2

**Среднее число шагов для достижения заданной точности решения**

| Алг. | $F_{sph}$ $n=9$ $M=8$ | $F_{Zah}$ $n=9$ $M=8$ | $F_{Ros}$ $n=3$ $M=36$ | $F_{Ras}$ $n=2$ $M=36$ | $F_{Ac}$ $n=4$ $M=12$ | $F$ $n=4$ $M=36$ |
|---|---|---|---|---|---|---|
| $k_1$ | 141 | 360 | 692 | 28 | 58 | 62 |
| $k_2$ | 91 | 147 | 145 | 16 | 43 | 48 |

На рис. 1 и 2 показаны погрешности (среднее, минимальное и максимальное значения) решения задачи электромагнитного зондирования алгоритмами 1 и 2 соответственно в зависимости от шага итерационного процесса $k$ при $n = 2$, $M = 30$. Для каждого шага $k$ были проведены расчеты с начала с новыми значениями затравочных массивов генератора псевдослучайных чисел. Можно видеть, что алгоритм 1 на малом числе итераций не всегда сходится. Для меньших значений числа частиц $M$ алгоритм 1 и на любом шаге $k$ не всегда сходится в отличие от алгоритма 2, который показывает устойчивую сходимость на любом шаге, включая $M = 18$.

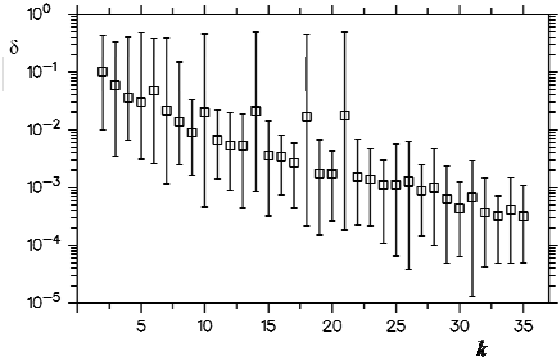



Рис. 1. Погрешность решения для $F$ (алгоритм 1)

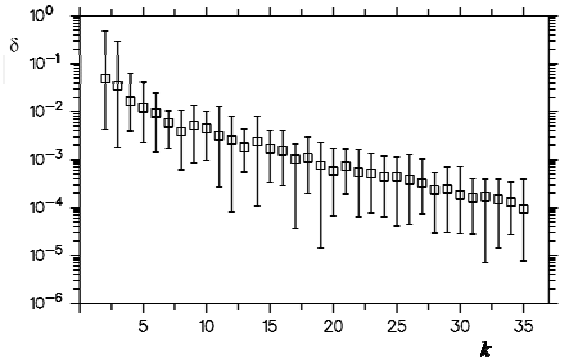



Рис. 2. Погрешность решения для $F$ (алгоритм 2)

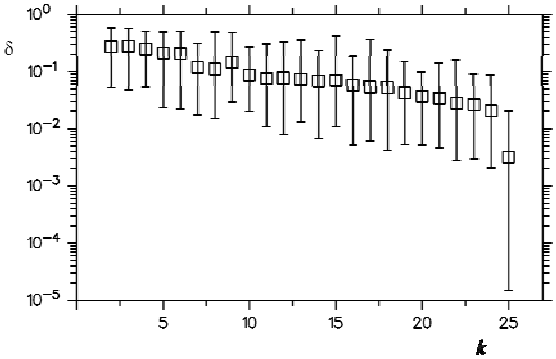



Рис. 3. Погрешность решения для $F$ с уточнением методом Нелдера-Мида (алгоритм 3)

На рис. 3 показана зависимость погрешности решения, полученная с помощью алгоритма 3 при размерности $n = 4$ и числе частиц $M = 36$. Можно видеть, что на последнем шаге уточнение методом Нелдера-Мида [7] существенно уменьшает погрешность, однако вычислительная сложность этого дополнительного этапа равна нескольким итерациям метода роя частиц.

## Список литературы

1. Вычислительные математика и техника в разведочной геофизике: справ. геофизика / под. ред. В.И. Дмитриева. – М.: Недра, 1990. – 498 с.

2. Jinlian Wang, Yongji Tan. 2-D MT Inversion Using Genetic Algorithm // Second International Conference on Inverse Problems. Institute of Physics Publishing. Journal of Physics: Conference Series, 2005. – No. 12. – P. 165–170.

3. Пересветов В.В. Генетические алгоритмы решения обратных задач электромагнитного зондирования // Параллельные вычислительные технологии (ПаВТ–2008): тр. междунар. науч. конф. (Санкт-Петербург, 28 января – 1 февраля 2008 г.). – Челябинск: Изд-во ЮУрГУ, 2008. – С. 433–437.

4. Semnani A., Kamyab M. An enhanced method for inverse scattering problems using Fourier series expansion in conjunction

with FDTD and PSO // Progress In Electromagnetics Research, PIER 76, 2007. – P. 45–64.

5. Мазалов В.Н., Пересветов В.В., Смагин С.И. Моделирование электромагнитных полей в слоистых средах с включениями. – Владивосток: Дальнаука, 2000. – 292 с.

6. Clerc M., Kennedy J. The particle swarm: Explosion, stability and convergence in a multi-dimensional complex space // IEEE Transactions on Evolutionary Computation, 6(1), 2002. – P. 58–73.

7. Банди Б. Методы оптимизации. Вводный курс: пер. с англ. – М.: Радио и связь, 1988. – 128 с.

**А.М. Першин, П.В. Писарев, Д.В. Зимин, В.Я. Модорский**

Пермский государственный технический университет

## ВЫЧИСЛИТЕЛЬНОЕ МОДЕЛИРОВАНИЕ ГИДРОДИНАМИЧЕСКИХ ПРОЦЕССОВ С УЧЕТОМ ВЗВЕШЕННЫХ ЧАСТИЦ И НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ В ФАСОННЫХ ИЗДЕЛИЯХ ТРУБОПРОВОДА

Известно, что оборудование, используемое при добыче и переработке полезных ископаемых, подвергается воздействию различных факторов, таких как коррозия, износ.

В ходе выполнения работы необходимо выполнить расчёты для трубопроводов технологического оборудования по переработке калийных солей. Это оборудование работает в контакте с пульпой и подвержено воздействию как химически агрессивной среды, так и механической смеси, вызывающей абразивный износ участков поверхности. При длительном воздействии такое оборудование изнашивается и требует замены. Это в полной мере относится и к фасонным изделиям трубопровода, через которые постоянно проходит поток гидропульпы, состоящий из водного раствора и твердых частиц солей KC1 и NaC1.

Применение новых материалов и технологиий позволяет решать подобные проблемы путем замены традиционно используемых сталей на композиционные материалы. Высокие физико-механические свойства, коррозионная стойкость, химостойкость и износостойкость обеспечили композитам успех в различных отраслях промышленности.

В данной работе исследуется поведение износостойких фасонных изделий трубопроводов из композиционных материалов. Целью проекта является улучшение эксплуатационных свойств конструкции, включая прежде всего такой параметр, как увеличение срока службы. Важной задачей является решение проблемы интенсивного износа: обоснованного выбора материалов, возможности прогнозирования скорости износа, оптимальных конструкторско-технологических решений.

**Подготовка и проведение вычислительного эксперимента по гидродинамической части задачи**

*Формирование физической модели гидродинамического процесса*.

В данном разделе рассматривается формирование расчетных моделей для анализа полей гидродинамических параметров в фасонных изделиях пульпопроводов и воздуховодах аспирации в ходе численного решения трехмерной по пространственным координатам гидродинамической задачи.

С учетом принятых допущений сформулирована следующая физическая модель:

- конструкция полагается трехмерной, ($x$,$y$,$z$);
- рабочее тело (1-я несущая фаза) представляет собой сжимаемую жидкость;
- рабочее тело (2-я несомая фаза) представляет собой сферические несжимаемые частицы диаметром 4 мм;
- камера постепенно заполняется частицами (соли).

**Подготовка вычислительного эксперимента по гидродинамической части задачи**

На рис. 1 и 2 показаны выполненные в пакете Solid Works конструктивные схемы сферического переходника с диаметром

входного патрубка 300 мм и тройника с патрубками диаметром 300 мм. Другие конструктивные схемы для сокращения объема аналогичной информации не приводятся.

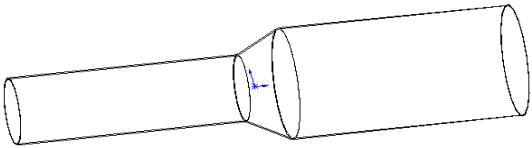


Рис. 1. Геометрическая модель переходника (прозрачная)

На рис. 3 и 4 показана расстановка на гранях расчётной области граничных условий, задаваемых в пакете FlowVision. Расчетная область представляет собой весь внутренний объем модели.

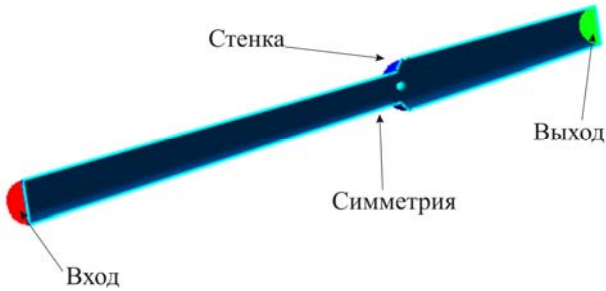



Рис. 2. Расстановка граничных условий переходника

Структура расчетной сетки следующая. Для лучшей сходимости решения и снижения погрешностей получаемых результатов необходимо применить сетку, ячейки которой имеют форму, близкую к форме куба. Помимо прочего при измельчении сетки желательно избежать резких отличий геометрических размеров соседствующих ячеек – линейные размеры соседних ячеек не должны отличаться более чем в 2 раза. Расчетная сетка, примененная для решения поставленной задачи, имеет 2 уровня адаптации. Для подводящего канала достаточно сетки с довольно крупными ячейками в силу того, что в нем не наблюдается каких-либо градиентов давления/скорости.

**Формирование физической модели оценки НДС конструкции**

*Построение физической модели*

Построение физической модели включает в себя идеализацию свойств конструкции и внешних воздействий.

В общем случае конструкция, изготовленная из реального материала, находящаяся под действием внешних нагрузок, может иметь много особенностей, включающих в себя несовершенство формы, свойств материала, особенности в характере внешнего нагружения и т. п. В практических расчетах учесть все имеющиеся особенности конструкции, материала и нагружения невозможно. Конечно, привлечение ЭВМ расширило возможности учета в прочностных расчетах некоторых из перечисленных выше особенностей, но необходимо понимать, что как бы ни были велики мощности современных ЭВМ, их быстродействие и объем памяти, но и они не безграничны. Поэтому, приступая к практическим расчетам, мы вынуждены подменять реальные тела некоторыми идеализированными объектами – «механическими моделями».

Таким образом, физическая модель может быть наделена лишь частью свойств реальной конструкции, а поэтому проще ее математическое описание. От того, насколько удачно выбрана физическая модель конструкции, зависит в конечном счете трудоемкость расчета и точность его результатов. Здесь многое зависит также от опыта расчетчика, его понимания работы конструкции, умения выделить те характеристики, которые в основном и определяют ее работу.

**Результаты гидродинамических расчётов**

В расчетах принимаем: $D_1$=350 мм, $D_2$=210 мм, α=60º, избыточное давление 1,6 МПа, массовая скорость на входе 5000 кг/м$^2$с.

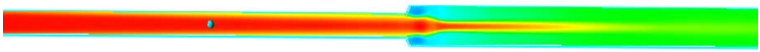


Рис. 3. Распределение давления в трубе

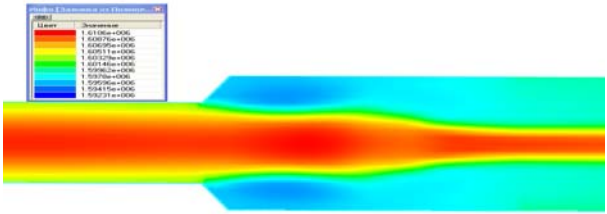


Рис. 4. Распределение давления в переходнике

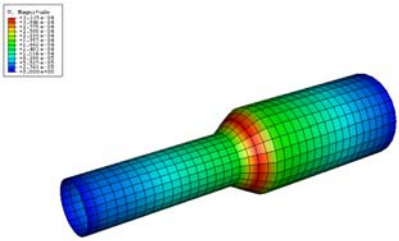


Рис. 5. Суммарное перемещение в слоях

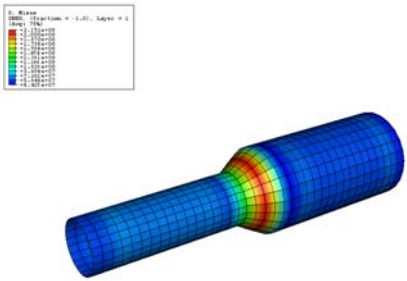


Рис. 6. Поля эквивалентных напряжений по Мизесу

В ходе вычислительных экспериментов были определены поля гидродинамических параметров в переходниках, отводах и тройниках различных типоразмеров. При этом произведена оценка параметров напряженно-деформированного состояния фасонных изделий (рис. 5, 6). Рассчитаны характеристики износа и определены области констаруктивных параметров при которых износ минимален.

**П.В. Писарев, В.Я. Модорский**

Пермский государственный технический университе

# ЧИСЛЕННЫЙ АНАЛИЗ ДИНАМИЧЕСКОГО НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ КОНЕЧНОМЕРНОГО ЦИЛИНДРА, НАГРУЖЕННОГО ГИДРОДИНАМИЧЕСКИМ ПОТОКОМ ЖИДКОСТИ НА МНОГОПРОЦЕССОРНОМ ВЫЧИСЛИТЕЛЬНОМ КОМПЛЕКСЕ

В представленной работе моделируется взаимовлияние конструкции, представляющей собой тонкостенный цилиндр конечной длины, и потока жидкости. Материал конструкции – резина. Торцы цилиндра жестко закреплены. Начальные данные: плотность твердого тела (резины) 1100кг/м$^3$, модуль Юнга 10 МПа, коэффициент Пуассона 0,45. Давление, действующее на входе, на выходе и на наружной поверхности трубы, 101300 Па. В качестве жидкой фазы принимается вода. Температура жидкости и окружающей среды 293 К. В работе определяются изменения давления жидкости и деформации при взаимном воздействии конструкции и гидродинамического потока. Решение сопряженных задач требует вычислительных ресурсов большой мощности. Данная задача реализована на кластере Пермского государственного технического университета, имеющего следующие характеристиками:

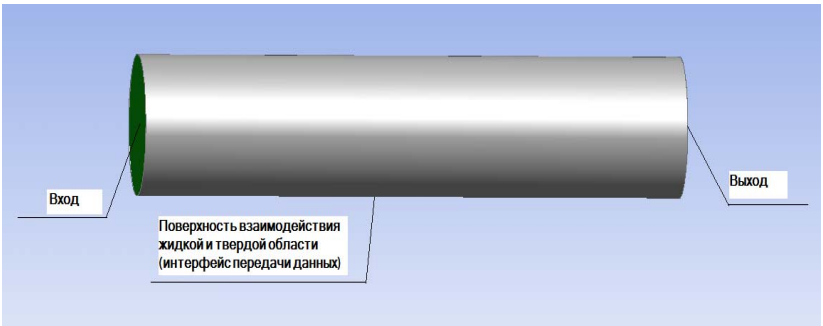



Рис. 1. Твердотельная модель

64 вычислительных узла;

128 четырех ядерных процессоров «Барселона-3» (всего 512 ядер);

пиковая производительность 4,096 Тфлоп;

производительность в тестовом пакете Linpack 78%;

объем системы хранения информации 12 ТБ;

объем оперативной памяти 512 Гбайт (8 Гбайт/узел).

В ходе расчетов использовались:

16 ядер для расчета гидрогазодинамической части задачи;

16 ядер для расчета НДС резиновой трубы.

Моделирование процессов производилось в многопроцессорном инженерном пакете ANSYS 12.1 на высокопроизводительном вычислительном комплексе ПГТУ. Твердотельная модель разработана в пакете Solid Works 2009 (рис.1).

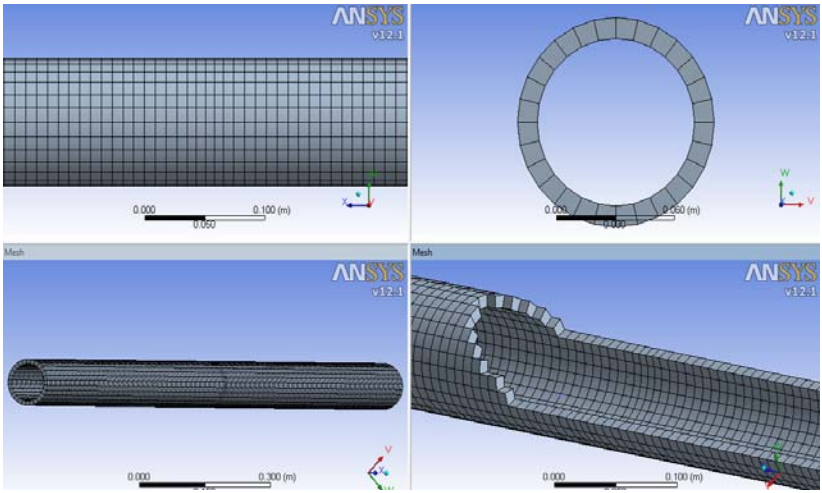



Рис. 2. Изображение расчетной сетки в плоскостях X0Y, Y0Z и изометрии

Расчетная сетка представляет собой совокупность гексаэдральных расчетных элементов, адаптированных в районе взаимодействия двух сред. Количество элементов расчетной сетки равно 6272 элементам. Изображение расчетной сетки твердого тела представлено на рис. 2. Качество и размерность

гексаэдральных элементов представлено на рис. 3. Изображение сетки для жидкой фазы представлено на рис. 4. Для достижения сходимости расчета размерность расчетных элементов твердотельной расчетной области и области жидкой фазы одинаковы.

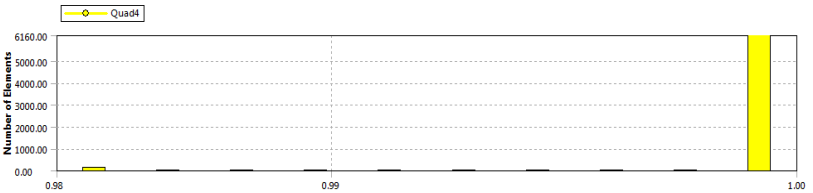



Рис. 3. Качество и размерность гексаэдральных элементов

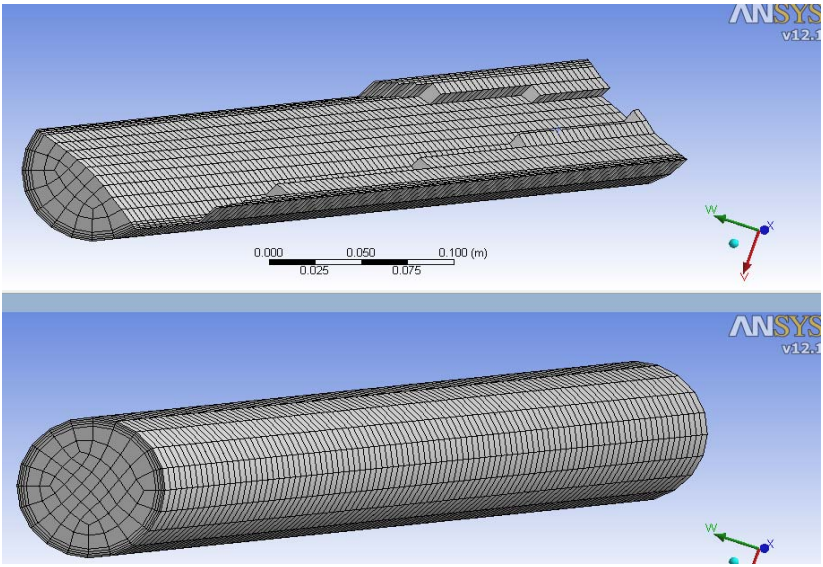



Рис. 4. Изображение сетки для жидкости

В данном расчете представлена модель несжимаемой жидкости. Эта модель позволяет исследовать течение вязкой жидкости при малых изменениях плотности и небольших числах Рейнольдса, $Re<10^4$. Предполагается, что изменения плотности обусловлены температурными или агрегатными неоднородностями.

Так как допустимые деформации материалов могут составлять сотни процентов, нельзя пренебрегать квадратом величины деформации. Для решения задач механики использовалась динамическая постановка с учетом динамической нелинейности описания материала резины.

Применительно к классу численных расчетов трубчатых конструкций, заполненных жидкостью, был разработан алгоритм моделирования в ANSYS 12.1 связанных задач гидроупругости (рис. 5).


Рис. 5. Алгоритм моделирования связанных задач гидроупругости в ANSYS 12.1

На рис. 6 представлено поле избыточного давления, действующего по внутренней поверхности цилиндра, в режиме установившегося потока жидкости.

Видно, что максимальное избыточное давление возникает в районе входа жидкой фазы и достигает величины 374,9 Па. Минимальное избыточное давление возникает в районе выхода жидкой фазы и достигает величины 37,49 Па.

На рис. 7 представлено поле напряжений, распределенных по внутренней поверхности твердого тела. Из рис. 7 видно, что зона максимальных напряжений располагается в местах изгиба,

вблизи торцов и составляет 0,428 МПа. Минимальное напряжение возникает в районе жесткой заделки.

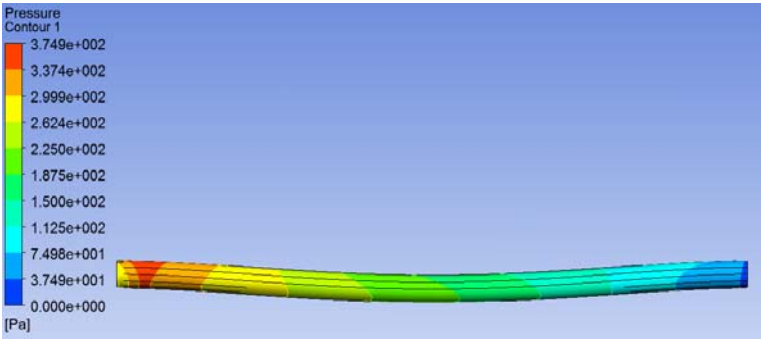



Рис. 6. Распределение давления по поверхности жидкости

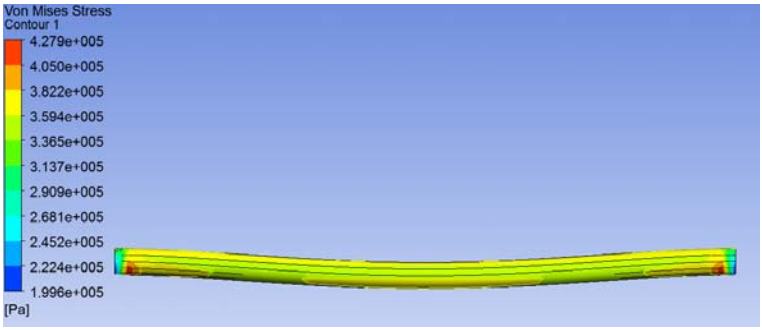



Рис. 7. Распределение напряжений по внутренней поверхности твердого тела

При проведении численных расчетов был разработан алгоритм моделирования в ANSYS CFX и ANSYS Mechanical связанных задач гидроупругости применительно к классу трубчатых конструкций, заполненных жидкостью. Данный алгоритм позволит рассматривать аэроупругие процессы взаимодействия в динамической системе «поток газа–деформируемая конструкция».

## Список литературы

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

2. Аэрогидроупругость конструкций / А.Г. Горшков [и др.]. – М.: ФИЗМАТЛИТ, 2000. – 592 с.

3. Вольмир А.С. Оболочки в потоке жидкости и газа: Задачи гидроупругости. – М.: Наука. Главная редакция физико-математической литературы, 1979. – 320 с.

4. ANSYS 12.1 Help.



**П.В. Писарев, В.Я. Модорский, А.А. Писарева**

Пермский государственный технический университет


## ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ СТРУИ ГОРЯЩЕГО ГАЗА С МЕТАЛЛИЧЕСКОЙ ПЛАСТИНОЙ С ИСПОЛЬЗОВАНИЕМ МНОГОПРОЦЕССОРНОГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА ПГТУ

Эксплуатация газовых горелок зачастую связана с большими тепловыми потерями. Избежать этого можно, изолировав нагревающую установку. Вместе с тем необходимо предотвратить перегрев стенок изолирующей конструкции при ее взаимодействии с газовым потоком. Устранение прогара и повышение качества таких конструкций требует уточнения методик расчета, комплексного учета особенностей рабочего процесса. При этом требуется решение проблемы в связанной постановке. В то же время решение сопряженных задач требует вычислительных ресурсов большой мощности. Данная задача реализована на кластере Пермского государственного технического университета имеющего следующие характеристиками:

64 вычислительных узла;

128 четырех ядерных процессоров «Барселона-3» (всего 512 ядер);

пиковая производительность 4,096 Тфлоп;

производительность в тестовом пакете Linpack 78 %;
объем системы хранения информации 12 ТБ;
объем оперативной памяти 512 Гб (8 Гб/узел);
В ходе расчетов использовались:
16 ядер для расчета гидрогазодинамической части задачи;
16 ядер для расчета НДС металлической пластины.


Рис. 1. Граничные условия на твердотельной модели

Моделирование производилось в многопроцессорном инженерном пакете ANSYS CFD (FLUENT, ANSYS Mechanical).

Твердотельная модель разработана в пакете Solid Works 2009 (рис. 1). Геометрическая модель представляет собой твердотельный канал газовой горелки, ось которого направлена на металлическую пластину. Ограничивающая область представляет собой параллелепипед и является расчетной областью газовой фазы. Поверхность пластины, соприкасающаяся с газом, образует интерфейс взаимодействия двух сред.

Расчетная сетка представляет собой совокупность гексаэдральных расчетных элементов, адаптированных в районе взаимодействия двух сред. Количество элементов расчетной сетки равно 1 200 000 элементов. Изображение расчетной сетки твердого тела представлено на рис. 2. Изображение сетки для жидкой фазы представлено на рис. 3. Для достижения сходимости расчета размерность расчетных элементов твердотельной расчетной области и области жидкой фазы одинаковы.

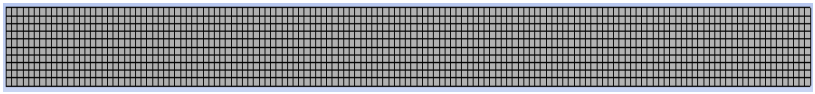


Рис. 2. Изображение расчетной сетки в плоскостях X0Y
для твердотельного региона

Рис. 3. Изображение расчетной сетки для жидкости

В данном расчете модель горения была заменена постоянным вдувом газа, с постоянными давлением и температурой на срезе горелки. В качестве математической модели была выбрана модель несжимаемой жидкости. Она позволяет исследовать течение вязкой жидкости при малых изменениях плотности и небольших числах Рейнольдса, $Re<10^4$. Предполагается, что изменения плотности обусловлены температурными или агрегатными неоднородностями.

Так как допустимые деформации материалов могут составлять сотни процентов, нельзя пренебрегать квадратом величины деформации. Для решения задач механики использовалось динамическая постановка с учетом динамической нелинейности описания материала резины.

Для моделирования теплообмена между твёрдой пластиной и газовой средой задавались модели теплопроводности (для твёрдого тела), конвекции (для газового потока), теплового излучения (лучеиспускание с поверхности твердого тела).

В ходе расчета получены поля распределения температуры на поверхности твердых тел (рис. 4). Из рис. 4 видно, что область максимальных температур возникает в пространстве между горелкой и пластиной и на поверхности пластины. Температура убывает по мере удаления от оси горелки.

На рис. 5. изображено распределение температур в плоскости X0Z, проходящей через ось канала горелки и вдоль пластины в момент времени $t$ = 1с. Максимальный нагрев наблюдается в области прямого соударения струи газа и металлической пластины.

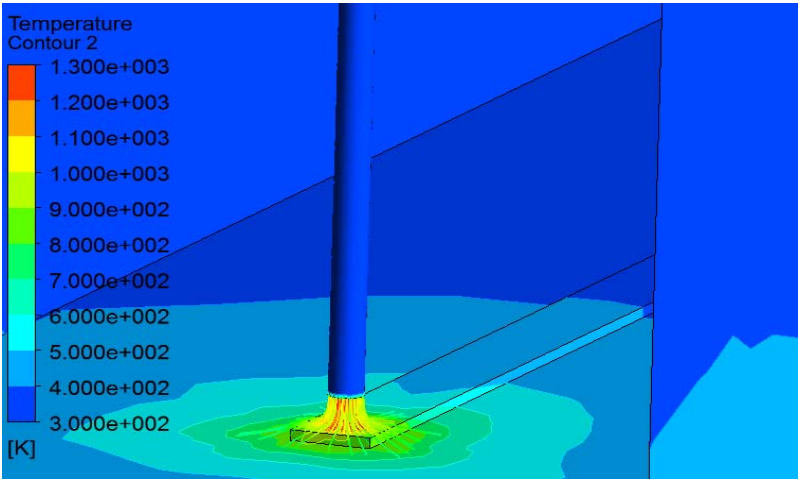



Рис. 4. Распределение температур по поверхностям твердых тел

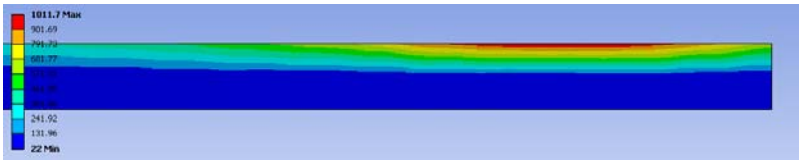



Рис. 5. Распределение температур в твердом теле (плоскость X0Z)

На рис. 6. изображено распределение нормальных напряжений по поверхности пластины в момент времени $t$ =1с. Видно, что высокие напряжения наблюдаются в районе торцов вблизи прямого контакта струи газа с пластиной. В зоне контакта наблюдается область низких напряжений.

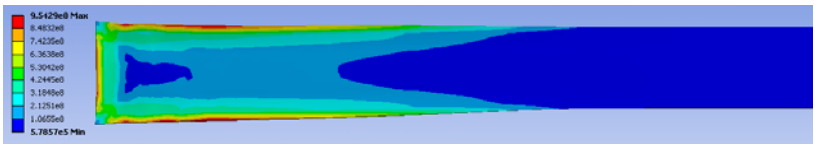



Рис. 6. Распределение нормальных напряжений
по поверхности пластины

При проведении численных расчетов был разработан алгоритм моделирования в ANSYS FLUENT и ANSYS Thermal связанных задач теплопередачи идеальным газом применительно к классу тонкостенных конструкций.

### Список литературы

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

2. Вольмир А.С. Оболочки в потоке жидкости и газа: Задачи гидроупругости. – М.: Наука. Главная редакция физико-математической литературы, 1979. – 320 с.

3. ANSYS 12.1 Help.

**О.В. Пищулина, В.Я. Модорский**

Пермский государственный технический университет

## ВЫЧИСЛИТЕЛЬНОЕ МОДЕЛИРОВАНИЕ НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ ПЕРСПЕКТИВНОЙ ОПРАВКИ ДЛЯ НЕПРЕРЫВНОЙ НАМОТКИ ТРУБ ИЗ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ

Создание высокотехнологичных производств связано с разработкой перспективных технологических установок. Предлагается создание современного производства труб из композиционных материалов с использованием подвижных оправок для непрерывной намотки. Их применение позволит добиться снижения стоимости изделий до уровня стоимости традиционных стальных труб за счет повышения производительности и снижения стоимости оборудования. При этом сохраняются все достоинства композиционных труб: долговечность, легкость, коррозионная стойкость.

Разработка новой конструкции оправки проводится в ходе вычислительного эксперимента. Базовый вариант оправки показан на рис. 1.

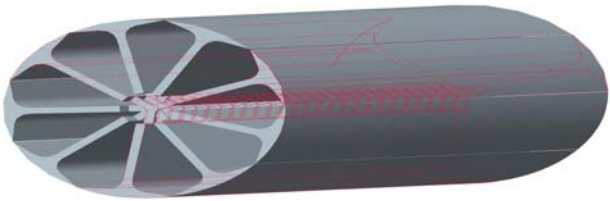


Рис. 1. Базовый вариант оправки

На первом этапе подготовки вычислительного эксперимента производится разработка физической модели:

конструкция трехмерная;

учитывается действующее давление на оправку при намотке $P = 0{,}6$ МПа;

конструкция закреплена консольно;

материал конструкции упругий, изотропный;

характеристики материала соответствуют Стали 40.

На втором этапе разработана математическая модель, которая включает в себя следующие уравнения:

уравнения равновесия в напряжениях;

геометрические соотношения Коши;

обобщенный закон Гука.

Исходная система дифференциальных уравнений в частных производных замыкается начальными и граничными условиями.

Разработан план проведения вычислительного эксперимента (табл. 1).

Таблица 1

**План проведения вычислительного эксперимента**

| Характеристика | Номер варианта | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | базовый | 2-й | 3-й | 4-й | 5-й | 6-й | 7-й | 8-й |
| D, мм | 150 | 200 | 320 | 150 | 150 | 150 | 150 | 150 |
| $S_{окон}$, мм$^2$ | – | – | – | 13 % | 23 % | 33 % | – | – |
| $h_{ребра}$, мм$^2$ | ГОСТ | ГОСТ | ГОСТ | ГОСТ | ГОСТ | ГОСТ | 3 | 1 |

Рассмотрим подробнее табл. 1. Варианты 2 и 3 отличаются от базового диаметром. Им соответствует определенный номер двутавра. Варианты 4, 5 и 6 отличаются от базового тем, что в полке элемента двутавра, служащей ребром жесткости, вырезаются «окна». Суммарная площадь всех окон составляет 13 % (вариант 4), 23 %(вариант 5) и 33 % (вариант 6) от площади полки двутавра без окон. Варианты 7 и 8 отличаются от базового тем, что толщина полки меньше, чем толщина полки базового варианта по ГОСТ.

Результаты проведения вычислительного эксперимента по оценке НДС оправки при действии наружного давления представлены в табл. 2.

Таблица 2

**Результаты вычислительного эксперимента**

| Номер варианта | Масса, (кг) | Макс. перемещения, $U_{max}(10^{-5}$ м) | Макс. деформации, $\varepsilon_{max}$ | Макс. напряжения, $\sigma_{max}$ (МПа) |
|---|---|---|---|---|
| 1-й базовый | 143,3 | 3,02 | 0,033 % | 66,61 |
| 2 | 258,9 | 3,8 | 0,035 % | 71,1 |
| 3 | 677,2 | 3,46 | 0,035 % | 70,29 |
| 4 | 124,2 | 3,38 | 0,033 % | 66,52 |
| 5 | 109,9 | 4,22 | 0,032 % | 64,18 |
| 6 | 95,6 | 7,73 | 0,042 % | 85,81 |
| 7 | 111,9 | 4,08 | 0,047 % | 95,73 |
| 8 | 80,5 | 173,59 | 0,002 % | 497,4 |

Наиболее рациональным представляется вариант 8, с окнами.

Результаты расчетов варианта 8 показаны на рис. 2–5. Построены поля перемещений (см. рис. 2), поля деформаций (см. рис. 3), поля напряжений (см. рис. 4) и поля перемещений вдоль оси OY (см. рис. 5) для оправки с толщиной ребра 1 мм без учета собственного веса, но с учетом давления на наружную поверхность оправки при намотке $P = 0{,}6$ МПа.

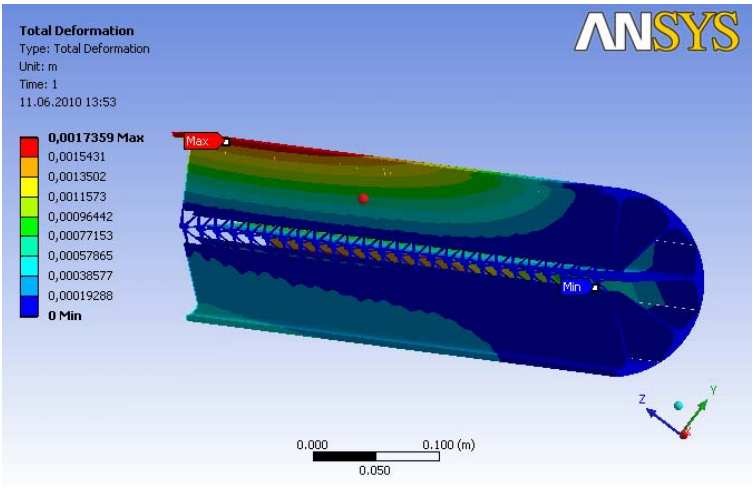



Рис. 2. Распределение суммарных перемещений, м

Анализ рис. 2 показал, что перемещения не превышают значений $4{,}089\times10^{-5}$ м. В заделке перемещения равны нулю.

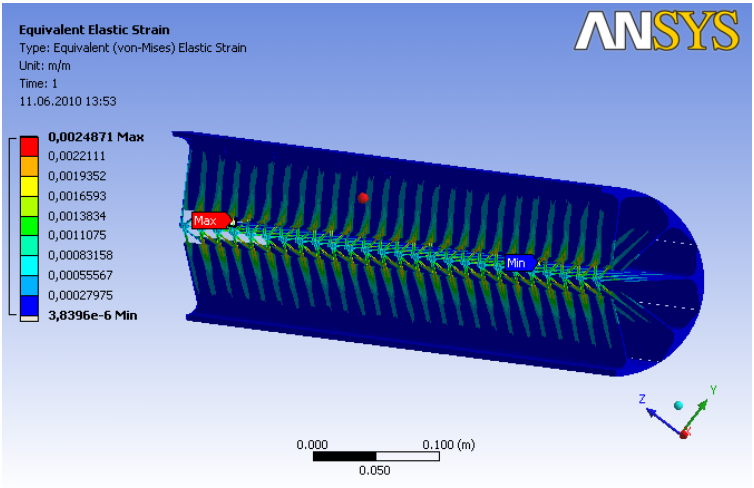



Рис. 3. Распределение деформаций

Анализ рис. 3 показал, что область максимальных деформаций располагается вблизи оси симметрии оправки, примерно на расстоянии 0,4 м от свободного торца оправки. Максимальные деформации достигают значения 0,002 %.

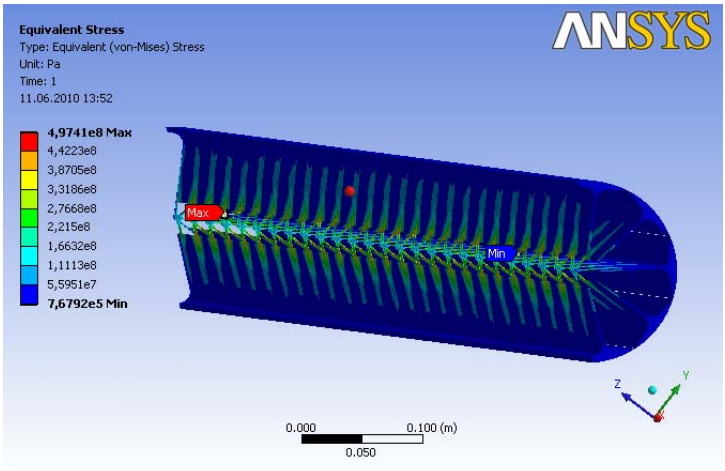



Рис. 4. Распределение напряжений, Па

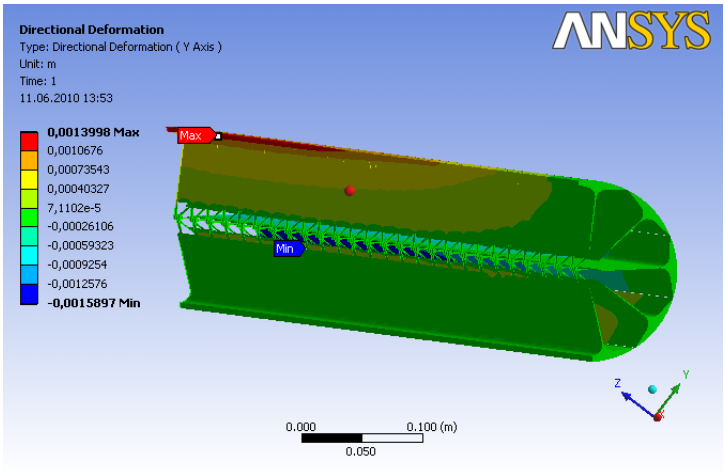



Рис. 5. Распределение перемещений OY, м

Анализ рис. 4 показал, что область максимальных напряжений располагается вблизи оси симметрии оправки, примерно на расстоянии 0,4 м от свободного торца оправки. Максимальные напряжения достигают значения 497,41 МПа.

Анализ рис. 5 показал, что перемещения вдоль оси OY не превышают значений $1{,}399\times10^{-3}$ м. В заделке перемещения равны нулю.

Полученные перемещения и напряжения меньше допустимых, а масса является минимальной.

**П.Н. Полежаев**

Оренбургский государственный университет

## АЛГОРИТМЫ ПЛАНИРОВАНИЯ ЗАДАЧ ДЛЯ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА С УЧЕТОМ СЕТИ И МНОГОПРОЦЕССОРНОСТИ УЗЛОВ

Существующие алгоритмы планирования задач для кластерных вычислительных систем, используемые в современных управляющих системах, демонстрируют неплохую эффективность работы. Однако имеется возможность дальнейшего роста их производительности за счет учета топологии вычислительной системы и многопроцессорности узлов.

Действительно, если алгоритм планирования будет назначать процессы параллельной задачи на топологически близкие вычислительные ядра, то это приведет к снижению времени ее исполнения в силу сокращения коммуникационных задержек при передаче данных между ее процессами. Это, в свою очередь, увеличивает производительность всей вычислительной системы.

Учет многопроцессорности вычислительных узлов при планировании также положительно сказывается на показателях работы кластерной системы, так как время выполнения коммуникационных операций между процессами параллельной программы, исполняющимися на соседних процессорах (или ядрах) одного узла, гораздо меньше, чем между процессами, исполняющимися на разных узлах.

Кроме того, при назначении параллельных программ на свободные вычислительные ядра необходимо учитывать сетевую конкуренцию между процессами одновременно исполняющихся задач. Ее снижение приводит к уменьшению времени выполнения сетевых коммуникаций, а это ведет к росту производительности кластерной системы.

Настоящее исследование направлено на разработку эффективных алгоритмов планирования задач для высокопроизводительных кластерных систем, учитывающих топологию, сетевую конкуренцию и многопроцессорность вычислительных узлов.

**Модель вычислительного кластера.** Разработанная модель вычислительного кластера [1] позволяет формировать аппаратную конфигурацию узлов вычислительной системы и соединяющей их высокопроизводительной сети с помощью взвешенного ориентированного графа. Множество вершин данного орграфа помимо вычислительных узлов также включает сетевые коммутаторы. Веса вершин-узлов представляют собой их статические параметры (количество вычислительных ядер, их производительность, объемы оперативной и дисковой памяти), а веса дуг – скорости передачи данных по соответствующим сетевым связям.

С помощью данной модели можно описать вычислительную систему произвольной топологии. В частности, в рамках данной работы она используется для задания вычислительных кластеров топологии толстого дерева, двумерного тора и звезды с однородными и неоднородными узлами.

**Исследуемые алгоритмы планирования задач на вычислительном кластере.** В рамках проводимого исследования были рассмотрены алгоритмы планирования задач, представляющие собой сочетания алгоритма выбора следующей задачи из очереди Most Processors First Served Scan (MPFS Scan) или алгоритма обратного заполнения Backfill с методами ее назначения на вычислительные ядра, учитывающими топологию вычислительной системы [1]: для топологии толстого дерева – Sorting Nodes by Performance (SNP), Fat Tree Sorting Commutators by Performance (FTSCP), Fat Tree Sorting Commutators by

Cores (FTSC); для топологии двумерного тора – модифицированные варианты алгоритмов Minimizing message-passing Contention 1x1 (MC1x1) и Minimizing message-passing Contention 1x1 Incremental (MC1x1+Inc); для топологии звезды – Sorting Nodes by Performance (SNP) и Sorting Nodes by Speed (SNS).

Также рассматривались сочетания алгоритмов MPFS Scan и Backfill с методами назначения, не принимающими топологию во внимание [2]: First Fit (FF), Best Fit (BF), Fastest Node Fist (FNF) и Random First (RF).

В ходе исследования были предложены собственные алгоритмы планирования, представляющие собой сочетания алгоритма MPFS Scan или Backfill с разработанными методами назначения Summed Distance Minimization (SDM) и Maximum Distance Minimization (MDM). Данные методы учитывают топологию вычислительной системы, сетевую конкуренцию между одновременно исполняющимися задачами и многопроцессорность вычислительных узлов.

Пусть методу назначения передается выбранная из очереди задача $J_j$, требующая для своего исполнения $n_j$ вычислительных ядер. Далее опишем принцип работы алгоритмов SDM и MDM.

Метод назначения SDM для каждого допустимого окна планирования $W \in WL'$ перебирает все сетевые устройства $d$ кластера (коммутаторы и узлы) и определяет для каждого из них $n_j$ ближайших вычислительных ядер окна $W$, подходящих по конфигурации для задачи $J_j$. Эти вычислительные ядра формируют возможное окно запуска $R_{W,d}$ задачи $J_j$. Для назначения в качестве результата $R$ выбирается окно $R_{W,d}$ с минимальным суммарным попарным расстоянием, если таких несколько, то выбирается окно с большей суммарной производительностью вычислительных ядер.

Данный метод является обобщением алгоритма Manhattan Median для случая произвольных топологий. За счет компактного размещения процессов параллельной задачи алгоритм SDM пыта-

ется минимизировать суммарное попарное расстояние между выделенными ей вычислительными ядрами. Это позволяет снизить сетевую конкуренцию между одновременно выполняющимися в системе задачами и уменьшить степень распределения процессов параллельных задач по вычислительной системе.

Метод назначения MDM для каждого допустимого окна планирования $W \in WL'$ перебирает все сетевые устройства $d$ кластера и для каждого из них запускает алгоритм поиска в ширину. В процессе его работы формируется множество достигнутых вычислительных ядер узлов окна $W$, которые подходят по конфигурации для задачи $J_j$, до тех пор, пока не будет получено $n_j$ ядер. При этом среди ядер, находящихся на одинаковом расстоянии от начального сетевого устройства в первую очередь выбираются те, которые имеют большую скорость. Результатом работы поиска в ширину является сформированное возможное окно запуска $R_{W,d}$. Для назначения задаче $J_j$ в качестве результата $R$ выбирается окно $R_{W,d}$ с минимальнальным максимальным расстоянием от первоначального сетевого устройства $d$.

Данный метод является аналогом алгоритма MC1x1 для произвольных топологий. За счет компактного размещения процессов параллельной задачи алгоритм MDM пытается минимизировать максимальное расстояние от выбранного центрального сетевого устройства до назначенных задаче вычислительных ядер.

**Симулятор вычислительного кластера и его управляющей системы.** Для экспериментального сравнительного исследования эффективности работы предложенных методов назначения с существующими вариантами использовался созданный в рамках настоящей работы симулятор вычислительного кластера и его управляющей системы.

Имитационная схема его работы представлена на рис. 1. Источник $I$ генерирует поток параллельных задач, отправляемых пользователями в очередь $Q$ управляющего узла $p_0$. Канал $S$ представляет собой планировщик, который в соответствии с заложенным в него алгоритмом осуществляет извлечение

задач из $Q$ и назначение их на свободные вычислительные ядра подходящих по конфигурации узлов.

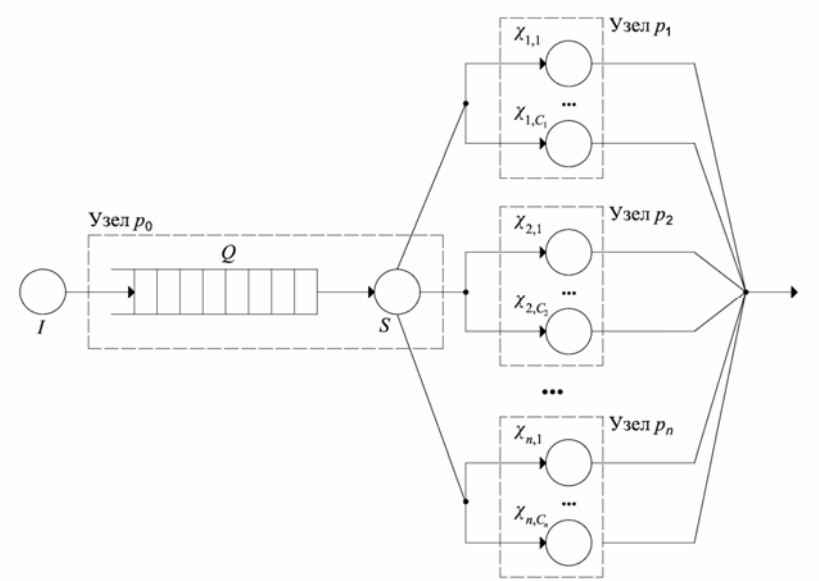



Рис. 1. Имитационная схема управляющей системы вычислительного кластера

Схема работы вычислительного узла $p_i$ приведена на рис. 2, $p_i$ соединен с другими узлами и коммутаторами вычислительной сети с помощью $r_i$ дуплексных связей. Все входящие пакеты, а также пакеты сообщений, генерируемых процессами, выполняющимися на локальных вычислительных ядрах, сначала поступают в очередь $Q_{inp.i}$, а затем маршрутизируются каналом обслуживания $R_i$. Если соответствующий пакет предназначен для локального вычислительного ядра, то он передается ему непосредственно, иначе – помещается в одну из очередей $Q_{out.i,1}, Q_{out.i,2}, \ldots, Q_{out.i,r_i}$, соответствующую выбранной алгоритмом маршрутизации исходящей связи. При выполнении маршрутизации каналом $R_i$ моделируется временная задержка величиной $l_{routing}$. Имитационная схема коммутатора представ-

ляет собой урезанный вариант имитационной схемы узла. Отличие только в том, что коммутатор не может выполнять вычисления, и поэтому у него отсутствуют вычислительные ядра.

На рис. 2. также приведена схема работы дуплексной связи $L_{ij} = \{E_{ij}, E_{ji}\}$, соединяющей сетевые устройства $d_i \in D$ и $d_j \in D$. Каналы обслуживания $E_{ij}$ и $E_{ji}$ добавляют к времени передачи пакета задержку величиной $b(E_{ij}) = b(E_{ji})$.

В качестве алгоритма работы симулятора используется алгоритм моделирования по принципу особых состояний (событий).

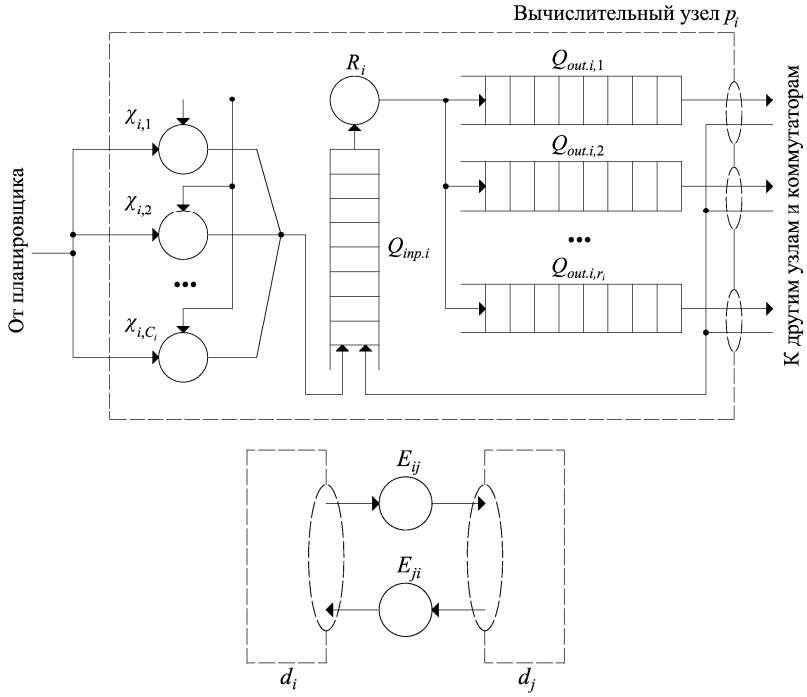



Рис. 2. Имитационная схема узла $p_i$ и дуплексной сетевой связи $L_{ij}$

Разработанная в настоящем исследовании модель загрузки вычислительной системы потоком задач использует для описания параметров генерируемых задач случайные величины с за-

данными законами распределения, а также определяет SPMD-структуру задачи в виде цикла, на каждой итерации которого происходит выполнение фаз коммуникации и вычислений. Данная модель позволяет генерировать типичные потоки задач, поступающие в очереди управляющих систем большинства современных вычислительных кластеров.

**Результаты экспериментального исследования.** Исследование проводилось для двух групп сценариев: вычислительный кластер с однородными узлами и однородной высокопроизводительной сетью и вычислительный кластер с неоднородными узлами и однородной сетью.

В первом случае моделировался кластер из 12 узлов, каждый из которых имеет 2 вычислительных ядра, 2 Гб оперативной памяти, 40 Гб локальной дисковой памяти, единичную относительную вычислительную скорость. Во втором – использовалась конфигурация из 12 узлов с суммарным количеством вычислительных ядер 24, разными объемами оперативной и дисковой памяти, разной относительной вычислительной скоростью.

Каждая группа сценариев рассматривалась для трех топологий: толстое дерево (два 6-портовых корневых коммутатора и три 8-портовых листовых коммутатора (4 восходящих порта и 4 узловых порта на каждом)), звезда (один 12-портовый коммутатор), двумерный тор размером 3×4 узла.

Оценка эффективности работы алгоритмов планирования осуществлялась с помощью системы количественных критериев и метрик, которая охватывает все аспекты состояния вычислительной системы: производительность, сбалансированность загруженности вычислительной системы, используемость памяти вычислительных узлов, гарантированность обслуживания задач, честность по отношению к задачам, характер распределения процессов параллельных задач по вычислительной системе и сетевой конкуренции между ними.

В частности, критерий производительности включает следующие метрики: средняя загруженность вычислительных ядер узлов кластера $U_{cores}$, потеря производительности кластера $CL$, максимальное время завершения задачи $C_{\max}$, среднее время

ожидания задач в очереди $\overline{T}_{wait}$ и среднее ограниченное замедление задач $\overline{s}_{lim}$.

Для получения достоверных результатов симуляция проводилась в каждом сценарии 100 раз для различных потоков из 500 случайно сгенерированных задач. Все вычисляемые значения метрик усреднялись по всем потокам.

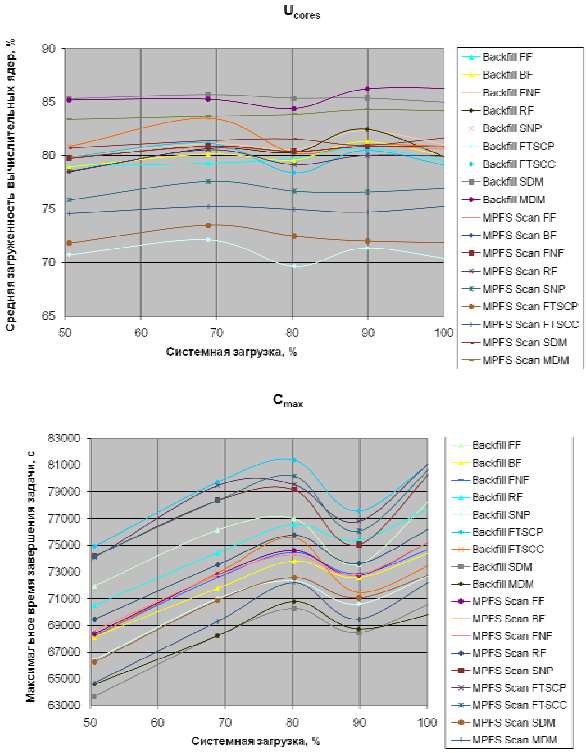



Рис. 3. Графики зависимости значений метрик производительности $U_{cores}$ и $C_{\max}$ от величины системной загрузки $L$

Сравнение алгоритмов планирования осуществлялось путем построения графиков зависимости метрик от величины системной загрузки $L$. Например, на рис. 3 изображены графики зависимости метрик производительности $U_{cores}$ и $C_{\max}$ от $L$ для сценария кла-

стера топологии толстого дерева с неоднородными вычислительными узлами. Заметим, что алгоритмы планирования в первую очередь сравнивались по критерию производительности, остальные критерии носили вторичный характер.

**Алгоритмы планирования, показавшие лучшие результаты в каждом сценарии**

| Группа сценариев | Топология | Наилучшие алгоритмы |
|---|---|---|
| Неоднородные вычислительные узлы и однородная сеть | Толстое дерево | Backfill SDM (при $L < 85$–$93$ %), Backfill MDM (при $L \geq 85$–$93$ %) |
| | Двумерный тор | Backfill SDM (при $L < 67$–$71$ %), Backfill MDM (при $L \geq 67$–$71$ %) |
| | Звезда | Backfill SDM, Backfill MDM |
| Однородные вычислительные узлы и однородная сеть | Толстое дерево | Backfill SDM (при $L < 75$–$79$ %), Backfill MDM (при $L \geq 75$–$79$ %) |
| | Двумерный тор | Backfill SDM (при 71–73 %), Backfill MDM (при $L \geq 71$–$73$ %) |
| | Звезда | Backfill SDM, Backfill MDM, Backfill SNP |

Полученные результаты экспериментального исследования сведены в таблицу. Для каждого сценарии может быть рекомендован свой алгоритм планирования, представляющий собой сочетания алгоритма выбора следующей задачи из очереди Backfill с разработанными методами ее назначения на вычислительные ядра MDM или SDM.

**Список литературы**

1. Гергель В.П., Полежаев П.Н. Теоретические основы экспериментального исследования алгоритмов планирования задач для вычислительного кластера с помощью симулятора // Вестн. Оренбург. гос. ун-та. 2010. – № 9(115). – С. 114–120.

2. Полежаев П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ–2010): тр. междунар. конф. – Челябинск: Изд-во ЮУрГУ. 2010. С. 287 – 298.

**С.В. Рощин, И.И. Зиновьев**

Владимирский государственный университет им. А.Г. и Н.Г. Столетовых

## ЗАДАЧА ОБНАРУЖЕНИЯ ЛИЦ НА ИЗОБРАЖЕНИИ С ИСПОЛЬЗОВАНИЕМ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

Задача обнаружения лица на изображении (face detection) часто является «первым шагом», предобработкой в процессе решения задачи «более высокого уровня» (например, узнавание лица, распознавание выражения лица). Однако и сама информация о присутствии и, возможно, количестве лиц на изображении или в видеопотоке может быть полезна для таких приложений, как охранные системы и содержательная индексация базы данных изображений или видеофрагментов.

Существующие алгоритмы обнаружения лиц можно разбить на две широкие категории. К первой категории относятся методы, отталкивающиеся от опыта человека в распознавании лиц и делающие попытку формализовать и алгоритмизовать этот опыт, построив на его основе автоматическую систему распознавания. Вторая категория опирается на инструментарий распознавания образов, рассматривая задачу обнаружения лица как частный случай задачи распознавания.

Основа методов первой категории – эмпирика является одновременно их сильной и слабой стороной. Трудно эффективно перевести неформальный человеческий опыт и знания в набор формальных правил, поскольку чересчур жесткие рамки правил приведут к тому, что в ряде случаев лица не будут обнаружены и, напротив, слишком общие правила приведут к большому количеству случаев ложного обнаружения. Методы вто-

рой категории дальше продвинулись в качестве обнаружения лиц на изображении. Но достигнуто это ценой их высокой вычислительной сложности. Во-первых, сами классификаторы часто включают в себя большое количество достаточно сложных вычислений; во-вторых, полный перебор всех возможных прямоугольных фрагментов изображения сам по себе занимает большое количество времени. Это затрудняет использование некоторых методов в системах реального времени (например, отслеживание перемещения лица в видеопотоке).

В качестве примера алгоритма второй категории возьмем классификатор Viola-Jones [1], в основе которого лежит алгоритм boosting (усиления слабых классификаторов). Для тестирования взята реализация данного алгоритма в библиотеке OpenCV (англ. *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) [3]. На практике алгоритм показал высокий уровень обнаружений и низкий уровень ложных срабатываний (рис. 1).

В OpenCV представлен однопоточный вариант алгоритма, что не позволяет использовать все возможности современных компьютеров. Поэтому был реализован собственный многопоточный вариант алгоритма. Поскольку задача поиска лица на изображении чаще всего локализована на одном компьютере и применение вычислительного кластера невозможно, для реализации многопоточного алгоритма использовалась технология OpenMP. Анализ алгоритма работы классификатора Viola-Jones и исходного кода библиотеки OpenCV показал, что сам процесс классификации участка изображения занимает немного времени, основную вычислительную сложность представляет перебор всех возможных прямоугольных участков изображения. Именно эта часть алгоритма и была распределена между всеми доступными процессорами. Для хранения промежуточных результатов используется разделяемый между потоками массив. В конце работы алгоритма главный поток обрабатывает полученные промежуточные данные с целью усреднения результатов, так как в результате работы алгоритма одно лицо может обнаруживаться несколько раз в разных масштабах.

Рис. 1. Результат работы классификатора Viola-Jones

Временные характеристики работы многопоточного алгоритма Viola-Jones приведены на рис. 2.

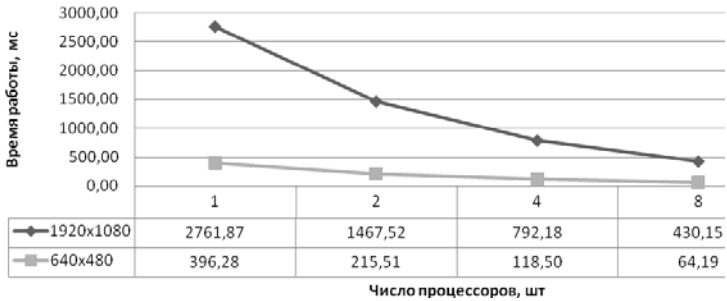



Рис. 2. Временные характеристики работы классификатора Viola-Jones

Из проведенных экспериментов становится ясно, что современные персональные компьютеры вполне справляются с задачей поиска лиц в видеопотоке с обычной веб-камеры в реальном времени. Для обработки видео высокого разрешения необходимы более эффективные алгоритмы.

## Список литературы

1. Paul Viola, Michael J. Jones. Robust real-time face detection // International journal of computer vision. – 2004, 57(2). – P. 137–154.

2. Cha Zhang and Zhengyou Zhang. A Survey of Recent Advances in Face Detection / Microsoft Research Technical Report, 2010.
3. http://opencv.willowgarage.com/wiki.
4. http://openmp.org/wp.

**А.В. Русаков**

Нижегородский государственный университет им. Н.И. Лобачевского

## ОБ ОДНОМ ПОДХОДЕ К СРАВНИТЕЛЬНОМУ АНАЛИЗУ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ В СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ

В настоящее время наблюдается неуклонный рост производительности вычислительной техники, что открывает все новые возможности для решения ресурсоемких задач, имеющих большое практическое значение. Такие задачи можно встретить в физике, химии, биологии, экономике, финансах и т.д. При этом одно из основных направлений развития состоит в частичном дублировании устройств, интегрированных на кристалле центрального процессора. Наличие и распространение таких вычислительных систем потребовало новых технологий, моделей, методов и программных средств системного уровня, ориентированных на их эффективное использование. Так, в научной периодике наблюдается неуклонный рост числа публикаций, связанных с новыми технологиями для параллельного программирования. Разрабатываются новые подходы (OpenMP, MPI), фреймворки (OpenCL, CUDA), библиотеки (TBB, CCR), языки (Cilk, Erlang, APL, Sisal) параллельного программирования. Одна из основных проблем заключается в сравнительной сложности разработки параллельных программ с нуля. Переход от обычного последовательного кода к параллельному требует определенных навыков системного программирования, не всегда присутствующих у прикладных программистов – ключевых потребителей новых технологий параллельных вычислений. Таким образом, на первый план выходят два вопроса: производительность получаемых параллельных программ и трудоемкость их разработки.

**Используемые критерии.** Очевидно, параллельные программы, созданные с применением высокоуровневых надстроек, в большинстве своем не могут опередить по производительности код, написанный с использованием низкоуровневых средств параллелизма. Однако в области прикладного программирования часто оправданна попытка пожертвовать незначительной долей производительности ради удобства разработки.

На сегодняшний день существует множество публикаций, в которых дается сравнение производительности программ с использованием различных технологий параллельного программирования, однако вопрос времени обучения и трудоемкости разработки программ с их использованием мало изучен. Одним из немногих примеров может служить работа [3].

В данной работе приводится сравнение особенностей выбранного набора технологий параллельного программирования для систем с общей памятью. Первая часть работы описывает анализ производительности каждой из технологий на примере тестовой задачи, во второй же части предлагается набор метрик для определения трудоемкости разработки программ с их использованием.

**Класс задач.** Сравнение выбранных технологий параллельного программирования производится на примере задач финансовой математики. Финансовая математика – раздел прикладной математики, связанный с финансовыми расчетами. Основу задач данной области составляют вычисления стоимостей производных ценных бумаг (опционов, акций, облигаций и других). Модель финансового рынка в данных задачах описывается с использованием систем стохастических дифференциальных уравнений, а алгоритмы решения обычно задействуют Монте-Карло-моделирование, методы оптимизации и численное интегрирование. Исходя из этого, решаемые задачи относятся к классу высокопроизводительных вычислений, однако имеют ясное прикладное значение и конкретный экономический смысл. Выбор данной области также обусловлен наличием у автора опыта решения подобных задач.

**Предлагаемый подход для анализа производительности.** Оценка производительности программ, написанных с использованием различных технологий параллельного программирования, заключается в стандартной процедуре сравнения времени выполнения параллельной программы со временем выполнения последовательной версии. Таким образом, при работе параллельного кода на системе с $p$ исполняющими устройствами во внимание берется величина $S_p = \frac{T_1}{T_p}$, называемая ускорением параллельного алгоритма, где $T_1$ – время вычислений на одном исполняющем устройстве, $T_p$ – время вычислений с использованием $p$ исполняющих устройств. Кроме того, для анализа эффективности той или иной технологии, применимой к конкретной задаче, рассматривается масштабируемость реализованного алгоритма при увеличении количества исполняющих устройств системы. В данном случае учитывается величина эффективности параллельного алгоритма $E_p = \frac{S_p}{p}$, равная отношению ускорения параллельной программы к количеству исполняющих устройств, на котором это ускорение было зафиксировано.

**Предлагаемый подход для анализа трудоемкости разработки.** Современная литература, посвященная вопросам управления качеством, содержит большое количество различных метрик, использующихся для оценки качества и сложности программного обеспечения и программного кода [6, 7]. Автором данной работы предлагается набор метрик, способных ярче в количественных значениях представить трудоемкость различных подходов к написанию параллельных программ. Далее приведен список метрик, сгруппированных по единицам измерения.

Временные:

- среднее время обучения технологии программирования;
- время разворачивания и настройки среды программирования;
- общее время разработки продукта;

- время разработки для каждой стадии;
- общие трудозатраты (в человеко-месяцах, человеко-часах).

Элементы кода:

- LOC'и (количество строк исходного кода – с комментариями и без);
- количество функций, операторов, операндов;
- метрики классовой сложности (количество классов, насыщенность классов, глубина наследования, связность, количество детей).

Тестирование и отладка

- количество ошибок при разработке и только на этапе тестирования;
- группы ошибок по типам;
- плотность ошибок на единицу кода;
- количество сборок кода.

Методом сбора временных метрик является фиксация самим программистом данных в процессе обучения и разработки. Метрики, связанные с элементами кода, собираются после написания программы при использовании вспомогательных средств интегрированных сред разработки. Данные о тестировании и отладке фиксируются средой разработки во время написания программы, а после группируются и усредняются программистом.

По результатам исследований состав метрик и методика могут быть модифицированы.

**Пример применения.** В качестве тестовой задачи в работе используется одна из задач финансовой математики – нахождение справедливой цены опциона Бермудского типа. Для анализа производительности были выбраны следующие технологии параллельного программирования для систем с общей памятью: OpenMP, Intel® Threading Building Blocks (TBB) и OpenCL.

**Постановка задачи.** Рассмотрим $N$-мерный финансовый рынок в непрерывном времени. Будем использовать модель Блэка-Шоулса [1]

$$dB_t = rB_t dt,\ B_0 > 0,$$

$$dS_t^i = S_t^i \left( (r^i - \delta^i)dt + \sigma^i dW_t^i \right),\ S_0^i > 0,$$

где процессы $B = (B_t)_{t \geq 0}$ и $S^i = (S_t^i)_{t \geq 0}$, $i = 1,...,N$ описывает поведение облигаций $B$ и акций $S^i$ в момент времени $t \geq 0$ соответственно. Цена облигаций $B$ зависит от процентной ставки $r$, цена акций $S^i$ определяется волатильностью $\sigma^i$, процентной $r^i$ и дивидендной $\delta^i$ ставками. Случайная стохастическая составляющая цены акций $S^i$ описывается $i$-й компонентой Винеровского процесса $W^i = (W_t^i)_{t \geq 0}$ ($W = (W^1,...,W^N)$). Пусть матрица ковариации $COV = (\mathrm{cov}_{ij})_{i,j=1..N}$ описывает зависимости между ценами акций в каждый фиксированный момент времени.

Рассмотрим одну из часто встречающихся разновидностей опциона – max-call-опцион, для которого функция выплаты $h(t, S_t)$ имеет следующий вид:

$$h(t, S_t) = (\max_{i=1...N}(S_t^i) - P)^+,$$

где положительная константа $P$ определяется опционным контрактом, $x^+ = \max(x, 0)$. Решим задачу вычисления стоимости опциона Бермудского типа, который может быть предъявлен к исполнению в любой из заранее фиксированных моментов $t \in Time = \{t_0 = 0, t_1, t_2, ..., t_d = T\}$, где момент $T$ задает истечение срока действия опциона.

**Метод решения задачи.** Для решения данной задачи определения автором был реализован популярный алгоритм имитационных деревьев для определения истинной цены многомерных опционов Бермудского типа Broadie–Glasserman Random Trees 1997 [5], описанный в [2]. Программная реализация основана на работе [4].

**Результаты экспериментов.** В этом разделе приведены результаты вычислительных экспериментов, полученные с использованием следующего программного и аппаратного обеспечения:

- CPUs:
  - Intel(R) Core(TM)2 Duo 2.34GHz, Cache L1 2*64KB, Cache L2 4MB, Mem 3328 MB

- 4x Intel Xeon X7560 (8 cores, 2.27 GHz), Mem 64 GB
- GPU: NVidia 9800GT, (Intel Core2Duo E8500@3, 8GHz, Mem 2GB)
- Intel® Math Kernel Library (Intel® MKL) 10.2.2
- Intel® Thread Building Blocks (Intel® TBB) 3.0
- Intel® C/C++ Compiler 11.1, Microsoft ® Visual Studio 9.0

Ускорение программы на многоядерных системах для каждой из параллельных реализаций наглядно отображено в диаграммах (рис. 1, 2)

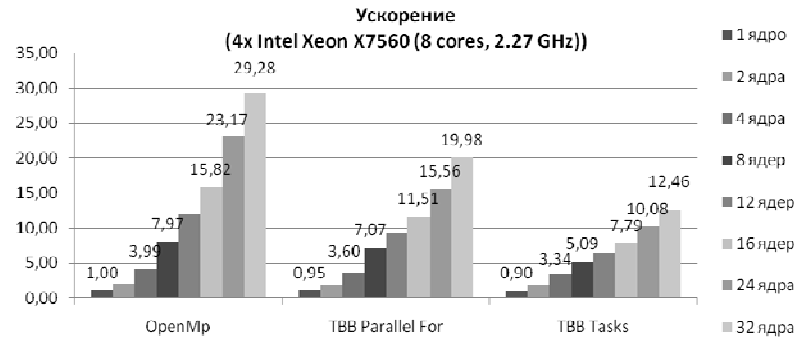



Рис. 1. Ускорение программы на 32-ядерной системе

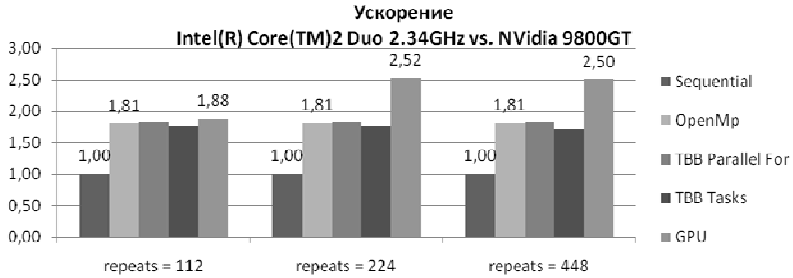



Рис. 2. Ускорение программы на 2-ядерной системе и графической карте

Анализируя полученные результаты, стоит отметить, что на настоящий момент автору не удалось добиться эффективного ис-

пользования высокоуровневой библиотеки TBB, предоставляющей гибкие возможности распараллеливания, по сравнению с технологией OpenMP. Это отчетливо видно на системах с большим количеством исполняющих устройств. Так, эффективность параллельного кода, созданного при помощи подхода TBB Tasks, на системе с 32 ядрами составляет всего около 39 %, при 63 % у TBB Parallel for и 91 % у OpenMP на аналогичном оборудовании. При этом необходимо отметить, что автор ранее не использовал Intel TBB. В дальнейшем планируется продолжить работу над оптимизацией этой версии. Для разработки GPU-версии использовалась технология OpenCL. Использование гетерогенных систем для исполнения параллельных программ на OpenCL по производительности оказалось чуть лучше остальных подходов на системе с двумя ядрами, но заметно проигрывает архитектурам с большим количеством ядер. Дальнейшая оптимизация OpenCL-версии также выглядит необходимой. Обращая внимание не только на производительность, но и на трудоемкость разработки, можно сделать следующие выводы.

Среди всех рассмотренных технологий для решения данной прикладной задачи, написание параллельного кода на OpenCL является самым трудоемким процессом. Это объясняется сложностью изучения архитектуры стандарта, большим количеством ошибок в коде программы даже у опытных программистов ввиду отсутствия отладчика ядер. Отдельные проблемы возникли в процессе разворачивания системы программирования для GPU.

Применение библиотеки TBB для написания программы потребовало значительного количества времени для реализации метода распараллеливания, основанного на логических задачах. Также этот подход оказался самым плохо масштабируемым на системах с большим количеством исполняющих устройств (в текущей реализации). Распараллеливание программы с помощью технологии OpenMP позволило добиться для данной задачи практически линейного ускорения на системах с количеством ядер до 32. Данный подход занял наименьшее время на обучение технологии среди всех рассмотренных и наименьшее количество времени для реализации самой программы.

В рамках работы, выполненной в лаборатории «Информационные технологии» (ITLab) ННГУ, проведен сравнительный анализ некоторых широко распространенных технологий и библиотек для параллельного программирования в системах с общей памятью, включая гетерогенные системы с использованием графических процессоров. Анализ проводился на основе решения тестовой задачи – определения справедливой цены опционов Бермудского типа методом имитационных деревьев (Broadie-Glasserman Random Trees). Автор благодарит компанию «Интел» за возможность тестирования программ на 32-ядерной архитектуре, полученной по результатам конкурса проектов Intel® Manycore Testing Lab. Для оценки трудоемкости разработки параллельных программ предложена система метрик. В дальнейшем планируется ее апробация.

## Список литературы

1. Black F., Scholes M. The pricing options and corporate liabilities // Journal of Political Economy. – 1973. – Vol. 3. – P. 637–659.

2. Broadie M., Glasserman P. Pricing American-style securities using simulation // Journal of Economic Dynamics and Control, June 1997. – Vol. 21. – P. 1323–1352.

3. A pilot study to compare programming effort for two parallel programming models / L. Hochstein [et al.]. – 28 December, 2007 (Preprint submitted to Elsevier).

4. Параллельная реализация одного алгоритма нахождения цены опционов Бермудского типа / А.С. Горбунова [и др.] // Высокопроизводительные параллельные вычисления на кластерных системах: материалы Пятого Международного науч.-практ. семинара / под ред. проф. Р.Г. Стронгина. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2005. – С. 60–67.

5. Эффективное использование вычислительных ресурсов для определения справедливых цен опционов бермудского типа / А.С. Горбунова [и др.] // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Вып. № 54. – СПб.: Изд-во СПбГУ ИТМО, 2008. – С. 145–151.

6. Новичков А. Метрики кода и их практическая реализация в IBM Rational ClearCase. – URL: http://www.ibm.com/developerworks/ru/edu/0108novich/section2.html, 04.08.2008.

7. Петрухин В.А., Лаврищева Е.М. Метрики качества программного обеспечения. – URL: http://www.intuit.ru/ department/ se/swebok/10/3.html, 24.09.2008.

**В.В.Рябов, С.В.Сидоров**

Нижегородский государственный университет им. Н.И.Лобачевского

## ИСПОЛЬЗОВАНИЕ КРИВЫХ ПЕАНО С РАСТУЩИМ УРОВНЕМ ДЕТАЛИЗАЦИИ В АЛГОРИТМАХ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ[1]

Данная работа продолжает развитие известного подхода к минимизации многоэкстремальных функций при невыпуклых ограничениях, описанного в работах [1–6] и получившего название *индексного метода* глобальной оптимизации. Подход основан на раздельном учете каждого ограничения задачи и не связан с использованием штрафных функций. При этом решение многомерных задач сводится к решению эквивалентных им одномерных. Соответствующая редукция основана на использовании кривых Пеано (называемых также развертками Пеано), однозначно отображающих единичный отрезок вещественной оси на гиперкуб, а также их обобщений («вращаемые развертки»), которые можно применять при решении задачи на кластерных системах с десятками и сотнями процессоров. Реализован прототип, использующий отображение единичного отрезка на гиперкуб с растущим порядком точности. Приведены результаты экспериментов, позволяющие оценить эффективность прототипа.

**Постановка задачи.** Рассмотрим задачу глобальной оптимизации вида

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

[1] Работа выполнена при поддержке совета по грантам Президента Российской Федерации (грант № МК-1536.2009.9).

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

где целевая функция $\varphi(y)$ и левые части ограничений $g_j(y), 1\le j\le m,$ удовлетворяют условию Липшица с соответствующими константами $L_j, 1\le j\le m+1,$ а именно

$$|\,g_j(y_1)-g_j(y_2)\,|\le L_j|y_1-y_2|,\ 1\le j\le m+1,\ y_1,y_2\in D.$$

Используя кривые типа развертки Пеано $y(x)$, однозначно отображающие отрезок $[0,1]$ на $N$-мерный гиперкуб $D$

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

исходную задачу можно редуцировать к следующей одномерной задаче:

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

Такая схема редукции размерности сопоставляет многомерной задаче одномерную, в которой соответствующие функции удовлетворяют равномерному условию Гельдера [1], а коэффициенты Гёльдера $K_j$ связаны с константами Липшица $L_j$ исходной задачи соотношениями $K_j\le 4L_j\sqrt{N}$ .

Различные варианты индексного алгоритма для решения одномерных задач и соответствующая теория сходимости представлены в работах [1, 4, 6].

**Использование множественных отображений (вращаемые развёртки).** Редукция многомерных задач к одномерным с помощью разверток имеет такие важные свойства, как непрерывность и сохранение обобщённой липшицевости. Однако при этом происходит потеря части информации о близости точек в многомерном пространстве, так как точка $x\in[0,1]$ имеет лишь левых и правых соседей, а соответствующая ей точка $y(x)\in R^N$ имеет соседей по $2^N$ направлениям. Как результат, в одномерной задаче может иметь место эффект расщепления глобального оптимума, что, естественно, ухудшает свойства одномерной задачи.

Сохранить часть информации о близости точек позволяет использование множества отображений вместо применения единственной кривой Пеано [3, 5]. Каждая кривая Пеано может быть получена в результате поворота на угол $\pm\pi/2$ в каждой из координ-

натных плоскостей. Число подобных поворотов определяется числом координатных плоскостей пространства, а общее число преобразований будет равно $N(N-1)$.

**Развёртки растущего уровня детализации.** Изложенные ранее подходы, сохраняющие часть информации о близости точек, используют множественные отображения, основанные на аффинных преобразованиях (смещениях, поворотах) исходного отображения. Однако следует отметить, что потеря информации о близости точек усугубляется и вместе с ростом детализации развёртки.

Предлагаемая в данной работе модификация способна дополнять подходы на основе множественных отображений. Модификация состоит в том, чтобы выполнять поиск сначала на «грубой» сетке и постепенно увеличивать детализацию развертки, тем самым усложняя получаемую одномерную целевую функцию $\varphi(y_m(x)) \rightarrow \varphi(y_{m+1}(x))$ и увеличивая предельную точность поиска. При этом необходимо определить правило перехода на следующий уровень детализации. Очевидными условиями перехода служат:

1) достижение такой точности поиска, что при текущем $m$ две различные точки гиперкуба сливаются в одну на отрезке;

2) остановка по общей заданной точности $\varepsilon$ для всех подзадач.

Процедура перехода от задачи с порядком развёртки $m$ к задаче с порядком развёртки $m+1$ требует также пересчёта координат каждой точки испытания на отрезке и вычисления оценки константы Гёльдера для новой целевой функции $\varphi(y_{m+1}(x))$. Из способа построения фрактального преобразования на основе кривой Пеано вытекает такое его свойство, что порядок точек испытаний на отрезке не меняется, поэтому переупорядочивание их в памяти не требуется, что существенно упрощает процедуру перехода на новый уровень детализации.

**Результаты экспериментов.** Для наиболее полного сравнения методов может быть использован аппарат операционных характеристик, предложенный В.А. Гришагиным [1]. Операционная характеристика метода – это кривая, показывающая зависимость числа решённых задач из определённого класса от числа испытаний.

Рисунок показывает операционные характеристики, построенные на функциях Гришагина (размерность которых $N = 2$), для последовательного глобального метода с фиксированным и растущим уровнями детализации, изображённые пунктирной и сплошной линиями соответственно.

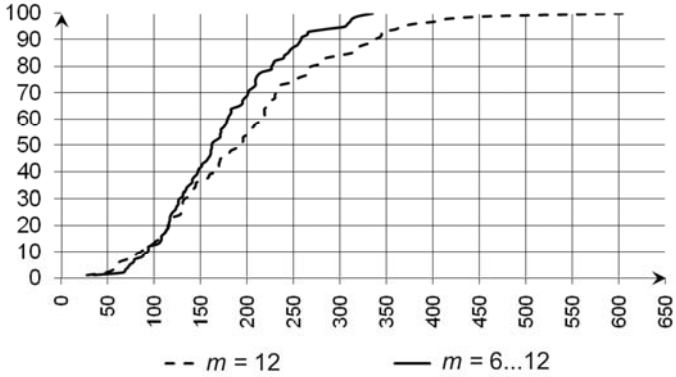



Рис. Операционные характеристики на функциях Гришагина

Значительная часть операционной характеристики для метода с растущим уровнем детализации лежит выше и левее, что убедительно доказывает эффективность предложенного подхода: при решении 100 % задач (верхние точки операционных характеристик) выигрыш по числу испытаний составляет 604/335 = 1,8 раза.

В рамках данной работы реализован прототип модификации последовательного индексного метода с растущим уровнем детализации развёрток, который дополняет другие подходы, улучшающие сходимость. Проведено исследование эффективности предложенной модификации. Результаты наглядно подтверждают ускорение сходимости. Эксперименты проведены с помощью системы глобальной оптимизации Global Expert, разработаной в ННГУ им. Н.И. Лобачевского.

## Список литературы

1. Стронгин Р.Г. Численные методы в многоэкстремальных задачах (Информационно-статистические алгоритмы). – М.: Наука, 1978.

2. Стронгин Р.Г. Поиск глобального оптимума. – М.: Знание, 1990.

3. Стронгин Р.Г. Параллельная многоэкстремальная оптимизация с использованием множества разверток // Вычисл. матем. и матем. физ. – 1991. – Т.31. № 8. – С. 1173–1185.

4. Стронгин Р.Г., Баркалов К.А. О сходимости индексного алгоритма в задачах условной оптимизации с ε-резервированными решениями // Математические вопросы кибернетики. – М.: Наука, 1999. – С. 273–288.

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

6. Баркалов К.А. Ускорение сходимости в задачах условной глобальной оптимизации. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2005.

7. Баркалов К.А., Сидоров С.В., Рябов В.В. Параллельные вычисления в задачах многоэкстремальной оптимизации // Вестн. ННГУ. Математическое моделирование и оптимальное управление. – Н. Новгород: Изд-во Нижегород. гос. ун-та. – № 6(1), 2009. – С. 171–177.

**В.А. Сапрыкин**

Нижегородский государственный университет им. Н.И. Лобачевского

## РЕАЛИЗАЦИИ АДАПТИВНОГО АЛГОРИТМА ВЫЧИТАНИЯ ФОНА: ПОСЛЕДОВАТЕЛЬНАЯ, ПАРАЛЛЕЛЬНАЯ, OpenCL-РЕАЛИЗАЦИЯ ДЛЯ GPU

С задачи отделения фона от переднего плана в видеопотоке начинаются многие алгоритмы машинного зрения [3]. Адаптивный алгоритм вычитания фона [1, 2], использующий смесь гауссианов, способен достаточно эффективно сегментировать изображение в условиях изменчивого заднего плана. Однако его производительность на современных настольных компьютерах зачастую не позволяет работать в реальном времени. Цель на-

стоящей работы – исследовать три направления повышения производительности: за счет оптимизации последовательной версии алгоритма, распараллеливания и реализации на графическом процессоре. При этом рассматриваются только такие способы реализации, которые не требуют внесения значительных изменений в существующий программный код, реализующий последовательную версию алгоритма. Такой подход упрощает поддержку нескольких реализаций одного алгоритма для различных вычислительных конфигураций. Кроме того, он позволят нам распространить полученные результаты на ряд других алгоритмов обработки изображений в виде методики повышения производительности готовой реализации.

**Постановка задачи.** Пусть есть исходная последовательная реализация некоторого алгоритма, опирающаяся на базовые функции библиотеки OpenCV [3]. Пусть такая реализация удовлетворяет следующим ограничениям:

1. Входная $\{A_1, A_2, ..., A_p\}$ и выходная $\{B_1, B_2, ..., B_q\}$ информация представляет собой множества двумерных изображений.

2. Все обрабатываемые изображения (входные, выходные, промежуточные) имеют одинаковый размер.

3. Каждый пиксель обрабатывается единообразно и независимо от остальных $(\forall i = 1, ..., q)(\forall x, y)\ B_i^{x,y} = g_i\left(A_1^{x,y}, A_2^{x,y}, ..., A_p^{x,y}\right)$.

Требуется разработать методы повышения производительности такого алгоритма и опробовать их на примере адаптивного вычитания фона. В частности, требуется разработать инструмент, позволяющий на основе последовательности вызовов функций OpenCV автоматически генерировать и запускать программы для GPU.

**Схема данных алгоритма вычитания фона.** Каждый пиксель изображения моделируется *n*-мерной (по числу цветовых каналов) случайной величиной, распределение которой является смесью *k* (обычно от 3 до 5) нормальных распределений [1, 2].

$$f(x) = \sum_{i=0}^{k-1} w_i N(x, \mu_i, Q_i) = \sum_{i=0}^{k-1} w_i \frac{1}{\sqrt{(2\pi)^n |Q_i|}} e^{-\frac{1}{2}(x-\mu_i)^T Q_i^{-1}(x-\mu_i)}, \sum_{i=0}^{k-1} w_i = 1, \quad (1)$$

где $w_i$ – вес, $\mu_i$ – вектор математического ожидания, $Q_i$ – матрица ковариации $i$-го $n$-мерного распределения.

Предполагая независимость одномерных случайных величин [8] для каждого цветового канала, имеем

$$Q_i = diag\left(\sigma_{i,1}^2, \sigma_{i,2}^2, ..., \sigma_{i,n}^2\right). \quad (2)$$

Таким образом, для хранения текущих оценок параметров распределения (1) нам потребуется $3k$ изображений: $M_0, ..., M_{k-1}$ – для хранения оценок матожиданий, $D_0, ..., D_{k-1}$ – для хранения оценок дисперсий, $W_0, ..., W_{k-1}$ – для хранения весов (рис. 1).

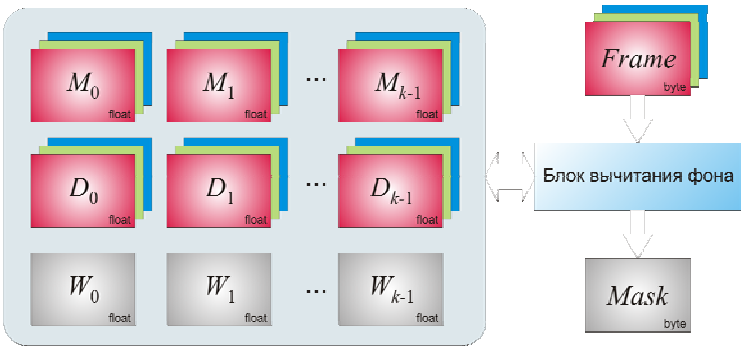



Рис. 1. Схема данных алгоритма вычитания фона

Время обработки алгоритмом каждого пикселя очередного кадра размера $a \times b$ постоянно. Оценим объем памяти, необходимый для работы алгоритма. Основная часть изображений хранится в формате с плавающей точкой, одинарной точной, по одному или по три цветовых канала. Маски хранятся в одноканальных изображениях с глубиной цвета 8 бит на пиксель. Последовательная реализация требует следующего объема памяти (байт) с учетом исходного, выходного и промежуточных изображений:

$$M = (29k + 60)ab. \quad (3)$$

**Автоматизация разрезания изображения на полосы.** В результате проведенных экспериментов в вычислительной системе на базе процессора Intel Core 2 Duo E8200 (2,6 ГГц, 6 Мб L2) было выявлено, что исходная последовательная реализация алгоритма, использующая смесь трех гауссианов, способна обрабатывать цветное изображение размером 640×480, поступающее с видеокамеры, со скоростью примерно 10,5 кадров/с.

Кэш-память процессора зачастую не в состоянии вместить целиком все основные и вспомогательные изображения, участвующие в обработке текущего кадра. Это дает возможность получения большей производительности, если обрабатывать изображение не целиком, а по частям.

Для автоматизации этого процесса был создан набор макросов, позволяющий легко превратить код, работающий с целыми изображениями, в код, работающий с полосой. Так, макрос CutImage (листинг 1) создает экземпляр структуры IplImage и подменяет указатель на целое изображение указателем на полосу этого изображения:

```
struct CStrip
{
  int fStartLine;
  int fEndLine;
};

#define CutImage(s, pImage, pSrtip) \
  IplImage Var_##pSrtip = *pImage;  \
  IplImage *pSrtip = &Var_##pSrtip; \
  Var_##pSrtip.imageData +=
s.fStartLine*Var_##pSrtip.widthStep; \
       Var_##pSrtip.height = s.fEndLine - s.fStartLine;
```

Есть макрос, делающий то же самое для массива изображений. Эти макросы можно использовать совместно с заключением нужного участка кода в составной оператор (блок), с тем чтобы получить возможность локально перекрывать имена исходных переменных. В итоге код последовательной реализации остается без изменений.

Минимальное количество полос, на которое следует разрезать изображение, можно попробовать оценить теоретически, воспользовавшись значением объема кэша процессора и формулой оценки требуемой памяти (3). Например, для описанного выше случая имеем $M = (29 \cdot 3 + 60)640 \cdot 480 \approx 50{,}9$Мб, что при объеме кэша в 6Мб говорит о необходимости разрезать изображение как минимум на 8–9 полос. Экспериментальные данные зависимости скорости обработки (кадров в секунду) от числа горизонтальных полос, на которые разбивается каждое изображение, подтверждают это значение (рис. 2). Кроме того, они позволяют оценить максимальное число полос, после превышения которого из-за возросших накладных расходов наступает некоторое снижение производительности. Таким образом, из графика легко видеть, что разбиение изображения на 10–30 горизонтальных полос позволяет достичь производительности 14,5 кадров в секунду.

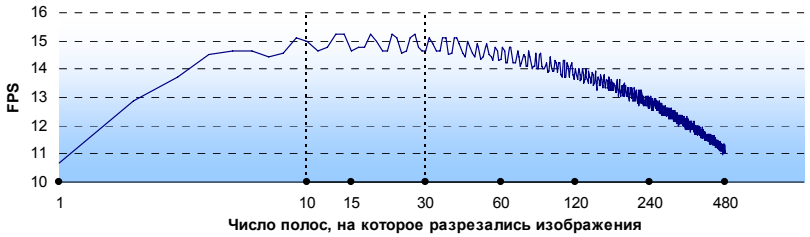



Рис. 2. Зависимость скорости обработки от числа полос

**Распараллеливание алгоритма.** Рассматриваемый алгоритм вычитания фона легко распараллеливается по данным. Наиболее удобно воспользоваться уже имеющимся у нас инструментом – разрезать очередной кадр на полосы и обрабатывать их параллельно на различных вычислительных ядрах. Из библиотеки Intel TBB [4] для этой цели удобно воспользоваться алгоритмом parallel_for, разбивая пространство номеров строк изображения. Существенно, что функтор, требуемый алгоритмом parallel_for, записывается довольно компактно вне зависимости от сложности алгоритма.

Размер порции данных (grain size) для parallel_for достаточно сильно влияет на итоговую производительность. Слишком большое значение скажется в худшую сторону на масштабируемости, а слишком маленькое – приведет к высоким накладным расходам [7]. Оптимальное значение можно выявить экспериментально. На следующем графике (рис. 3) в логарифмическом масштабе показана зависимость производительности (кадров в секунду) от величины grain size при распараллеливании вычислений на два потока. Таким образом, значение grain size из диапазона 10–60 дает возможность достичь производительности порядка 25 кадров в секунду.

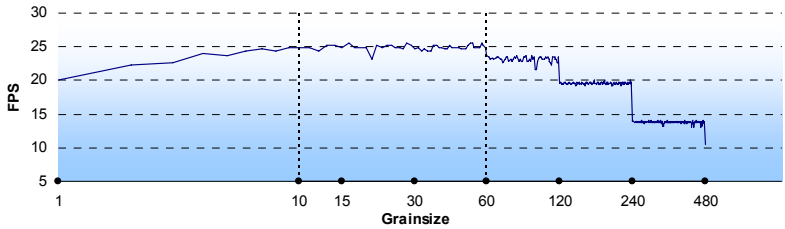



Рис. 3. Зависимость скорости обработки от величины grain size

Следует отметить, что библиотека TBB позволяет установить число создаваемых потоков в единицу. Тогда, указывая в качестве grain size полную высоту изображения, мы получим обыкновенную последовательную реализацию, а указывая меньшее значение – более производительную последовательную реализацию с разрезанием на полосы.

**Автоматизация реализации на GPU.** Для облегчения реализации алгоритма на графическом процессоре был создан следующий набор классов: CGpuContext, CVarList, CProgram. Класс CGpuContext является простой оберткой над функциями OpenCL, служащими для создания контекста и получения описателей устройств.

Класс CVarList позволяет сформировать список используемых в вычислении изображений. Для каждого изображения указывается тип памяти: частная/глобальная, при этом во втором случае в рамках контекста создаются соответствующие

объекты памяти (cl_mem). Все входные, выходные изображения и изображения, которые требуется хранить в памяти GPU между запусками kernel-а OpenCL, должны быть глобальными. Формирование списка происходит на основе указателей на структуры IplImage, описывающие каждое изображение. Причем обязательное выделение памяти под хранение данных требуется лишь для входных и выходных изображений.

За непосредственную генерацию текста OpenCL-программы отвечает класс CProgram. Он содержит ряд методов, сигнатуры которых совпадают с базовыми функциями OpenCV. В эти методы следует передавать те же указатели на IplImage, что использовались при формировании списка переменных. Ниже показан пример использования полученного набора классов для автоматической генерации OpenCL-кода, выполняющего взвешенное сложение изображений:

```
CGpuContext context(init);
CVarList vars(context);
vars.AddVar(A, atGlobalRead);
vars.AddVar(B, atGlobalRead);
vars.AddVar(C, atGlobalWrite);
vars.AddVar(Temp1, atPrivate);
vars.AddVar(Temp2, atPrivate);
vars.AddVar(Mask,
atGlobalRead);

CProgram prog1(vars);
prog1.BeginProgram("MyProg1");
prog1.cvScale(A,          Temp1,
alpha);
prog1.cvScale(B,   Temp2,   1.0-
alpha);
prog1.cvAdd(Temp1,   Temp2,   C,
Mask);
prog1.EndProgram();
```

```
__kernel void
MyProg1(
  __read_only
__global float4
*V0,
  __read_only
__global float4
*V1,
  __write_only
__global float4
*V2,
  __read_only
__global uchar
*V5)
{
  int id =
get_global_id(0);
  float4 V3;
  float4 V4;

  V3 = V0[id]*0.2;
//cvConvertScale
  V4 = V1[id]*0.8;
//cvConvertScale
  if (V5[id])
    V2[id] = V3+V4;
//cvAdd
}
```

В результате проделанной работы была создана методика повышения производительности адаптивного алгоритма вычитания фона, которая работает без внесения существенных изменений в исходный код реализации. На GPU удалось получить производительность в среднем 112 FPS (рис. 4) для цветного изображения 640×480 (Intel Core 2 Duo E8200 (2,6 ГГц, 6Мб L2), ОЗУ 2Гб, NVidia GeForce GTS 250). На рис. 4 также показаны частоты кадров, достигнутые исходной последовательной программой (1), последовательной программой с разрезанием изображения на полосы (1+) и параллельной программой, запущенной на двух потоках (2).

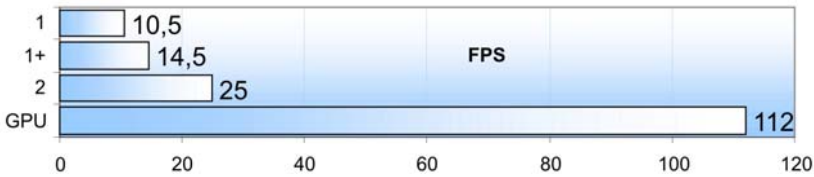



Рис. 4. Производительность при различных методах обработки изображений

В будущем планируется расширить сферу применимости созданных инструментов на более широкий класс алгоритмов. Для этого, в частности, потребуется научиться автоматически генерировать эффективный OpenCL-код на основе рекомендаций изложенных в статье [5].

Работа выполнена при финансовой поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

## Список литературы

1. Stauffer C., Grimson W. Learning Patterns of Activity Using Real-Time Tracking // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000. – Vol. 22, No. 8.

2. Power P.W., Schoonees J.A. Understanding background mixture models for foreground segmentation //Proceedings Image and Vision Computing New Zealand, 2002.

3. Bradski Gary, Kaehler Adrian Learning OpenCV. – O'Reilly, 2008.

4. Intel® Threading Building Blocks. Tutorial. Revision: 1.13. – Intel Corporation, 2007.

5. OpenCL Best Practices Guide. Version 1.0. – NVIDIA Corporation, 2009.

6. The OpenCL specification. Version: 1.0. – Khronos OpenCL Working Group, 2009.

7. Мееров И.Б., Сиднев А.А., Сысоев А.В. Библиотека Intel Threading Building Blocks – краткое описание. – Н. Новгород, 2007.

8. Федоткин М.А. Основы прикладной теории вероятностей и статистики. – М.: Высшая школа, 2006.

9. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. – М.: Вильямс, 2003.

**Б.Г. Севрюков, А.А. Лукьяница**

Московский государственный университет им. М.В. Ломоносова

## ИСПОЛЬЗОВАНИЕ GPU В ЗАДАЧАХ ТРЕХМЕРНОЙ РЕКОНСТРУКЦИИ СЦЕН

Задачи определения пространственной структуры реальных объектов или сцен по двумерным изображениям ставятся в разнообразных областях, таких как: компьютерное моделирование, навигация роботов по заранее неизвестной местности, кино и телевидение, медицина, архитектура. За последнее десятилетие методы решения этого класса задач получили существенное развитие, однако значительный объем вычислений, необходимый для получения качественной реконструкции, не позволяет использовать их в системах реального времени.

С другой стороны, вычислительная мощь современных видеокарт, невысокая цена и общая доступность делают их привлекательными для решения указанных задач. Поэтому нами было проведено исследование возможности переноса вычислений на GPU. Устройство, предназначенное для синтеза двумерных изображений, мы будем использовать для решения обрат-

ной задачи. Мы рассматриваем класс методов реконструкции, использующих два и более изображения. Общий процесс вычислений может быть представлен в виде конвейера, схема которого, а также методы и оценка возможности их эффективной реализации на видеокарте, приведены в следующей таблице.

**Общий процесс вычислений**

| Номер этапа | Название этапа | Методы | GPU |
|---|---|---|---|
| 0 | Формирование входного набора изображений, $n \geq 2$ | | |
| 1 | Выделение особых точек | SIFT | + |
| | | SURF | + |
| | | Детектор углов Харриса | + |
| 2 | Поиск соответствий между изображениями | Трекер особых точек Лукаса-Канаде | + |
| | | Оптический поток | + |
| | | KD-дерево | + |
| 3 | Вычисление фундаментальных матриц, факторизационные методы | SVD | + |
| | | QR | + |
| | | RANSAC | +/– |
| 4 | Самокалибровка в случае неизвестного фокусного расстояния | SVD | + |
| 5 | Выравнивание пучков (Bundle adjustment) | – | +/– |
| 6 | Создание плотной реконструкции | Триангуляция Делоне | + |
| | | Ректификация и сопоставление в случае стерео | + |
| | | Поиск минимального сечения на графе | + |
| 7 | Наложение текстуры и отрисовка | – | + |

В зависимости от условий процесс получения решения может меняться – некоторые этапы конвейера могут отсутствовать либо быть заменены другими. Например, при отсутствии информации о камерах для получения евклидовой реконструкции за этапом 3 должен следовать этап самокалибровки.

Отметим, что в данной задаче параллелизм может быть двух типов: массивный параллелизм по данным – когда распараллеливаются операции внутри каждого из блоков реконструкции для фиксированного набора ракурсов и параллелизм по наборам ракурсов, т.е. когда весь конвейер или отдельные его блоки выполняются параллельно для наборов ракурсов, а затем результаты реконструкции «склеиваются». Яркий пример второго случая – последовательная реконструкция по видео. Естественно, при должной организации процесса возможно комбинирование.

Основным препятствием на пути эффективного ускорения метода RANSAC является то обстоятельство, что он требует случайной выборки значений из памяти видеокарты, в результате чего доступ к памяти перестает быть согласованным между нитями. Для борьбы с этим можно предложить заранее помещать в память $n \cdot k$ значений, где $n$ – количество итераций алгоритма, $k$ – длина вектора значений (например, 7 двумерных точек в случае вычисления фундаментальной матрицы).

Процедура выравнивания пучков – процесс, построенный на методе Левенберга-Марквардта. Предположительно, он может быть достаточно эффективно распараллелен на GPU, однако довольно сложен и достоин отдельного исследования. Пока же стоит отметить, что он не является необходимым этапом конвейера реконструкции.

Результаты наших исследований показали, что состояние дел в области вычислений общего GPU позволяет использовать видеокарты для достижения эффективной производительности при решении задач реконструкции трехмерных сцен по изображениям.

**С.И. Соболев**

Научно-исследовательский вычислительный центр МГУ им. М.В. Ломоносова

# ТЕХНОЛОГИИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ: ТЕНДЕНЦИИ И ПЕРСПЕКТИВЫ

Технологии распределенных вычислений активно развиваются с середины 90-х годов прошлого века. Предпосылками к такому развитию стал, с одной стороны, бурный рост числа компьютерных систем самого разного уровня производительности, а также повышение качества и проникновения каналов связи, а с другой стороны – потребность решать все более сложные задачи, для которых мощности доступных вычислительных установок оказывалось недостаточно. Собственно технологии распределенных вычислений подразумевают использование для проведения расчетов уже имеющихся компьютерных ресурсов, соединенных между собой каналами связи общего назначения. Распределенные вычисления стали отчасти дополнением традиционных суперкомпьютерных технологий, позволяя объединять несколько суперкомпьютеров для решения задач высокой вычислительной сложности, а отчасти и альтернативой им, позволив для решения ряда задач обойтись и вовсе без суперкомпьютеров, подключив к расчетам только доступные офисные и домашние компьютеры, учебные классы и т.д.

Интересно, что распределенные вычисления в широком смысле можно рассматривать и как технологию параллельного программирования. Задача не распараллеливается традиционными средствами, такими как MPI или OpenMP, вместо этого создается «обвязка» для выбранной программной платформы распределенных вычислений, которая отвечает за разбиение задачи на независимые порции и компоновку результатов. При этом распределение заданий и все транспортные функции берет на себя выбранная платформа.

Прежде чем перейти к обзору состояний технологий распределенных вычислений, а также тенденций и перспектив их развития, стоит отметить еще два тесно связанных с ними комплекса технологий: грид-технологии и cloud computing («облач-

ные вычисления»). В настоящее время эти понятия зачастую смешивают, заменяя одно другим, однако, с нашей точки зрения, такое использование некорректно. Рассмотрим вкратце каждое из них.

Грид, согласно определению Я.Фостера и К. Кессельмана – согласованная, открытая и стандартизованная среда, которая обеспечивает гибкое, безопасное, скоординированное разделение ресурсов в рамках виртуальной организации [1]. Грид-технологии включают в себя все множество вопросов, решаемых при создании масштабных проектов по распределенной обработке и хранению данных – обеспечение безопасности доступа, поиск, учет и планирование использования ресурсов и т.д. Вследствие своей многофункциональности программное обеспечение для грид-сред (дефакто это Globus Toolkit [2] и gLite [3]) достаточно трудно в установке, настройке и использовании. Его применение не очень оправданно в тех случаях, когда необходимо достаточно оперативно развернуть распределенную среду и провести расчет на доступных ресурсах. Более того, в базовом варианте запуск задачи в грид-среду, по сути, означает ее запуск на одном из входящих в нее ресурсов; в случае когда необходимо выполнить серию таких запусков, т.е. фактически реализовать распределенный расчет, приходится использовать дополнительные средства. Стоит отметить, что такую функциональность предоставляет ПО промежуточного уровня, например, Unicore [4].

Cloud computing – технология обработки данных, обеспечивающая абстракцию между компьютерным ресурсом и нижележащей технической архитектурой, предоставляющая удаленный доступ к разделяемому массиву конфигурируемых компьютерных ресурсов, которые по запросу могут быть оперативно зарезервированы и задействованы с минимальными затратами на организацию и взаимодействие с непосредственно владельцами ресурсов. Если в грид-технологиях акцент делается на скоординированном использовании распределенных ресурсов, то в облачных вычислениях цель другая – предоставление компьютерных ресурсов в аренду. Облачные сервисы эффективно используются для обработки больших массивов данных или организации удаленного доступа к определенному программному обеспечению, однако их нельзя рас-

сматривать как готовые платформы для поддержки распределенных вычислений. Лимитирующими факторами, как правило, являются жестко стандартизированное программное обеспечение виртуальных платформ, не очень высокая пропускная способность каналов связи внутри облаков, а также принципиально платный характер использования сервисов.

Что же остается для распределенных вычислений в их классическом понимании, в каких направлениях будут развиваться эти технологии? Попробуем выделить наиболее важные, интересные и перспективные направления.

Инфраструктура больших проектов. Это те самые грид-среды, построенные для решения задач конкретной предметной области. В качестве одного из ведущих проектов такого класса можно назвать EGEE – Enabling Grids for E-sciencE [5] (ныне проект продолжается как EGI – European Grid Infrastructure [6]). С помощью этого проекта, в частности, реализуется среда для обработки и хранения данных, получаемых с Большого адронного коллайдера.

Организация масштабных расчетов путем привлечения максимально доступного числа вычислительных ресурсов различного плана – суперкомпьютеров, мощных серверов, рабочих станций. Такие вычислительные эксперименты могут строиться на основе инфраструктуры, подобной созданной в рамках программы СКИФ-ГРИД [7], объединяющей ряд крупнейших российских суперкомпьютерных центров и использующей ПО промежуточного уровня Unicore, либо на базе системы метакомпьютинга X-Com [8], позволяющей подключать к расчетам практически любые вычислительные ресурсы.

Volunteer computing – вычисления с использованием ресурсов компьютеров добровольных участников. Самый известный проект такого рода – SETI@home [9], цель которого – распределенная обработка сигналов, получаемых с радиотелескопа. В ходе развития этого проекта была разработана программная платформа BOINC [10], позволяющая на своей основе строить аналогичные проекты для различных научных областей. В настоящее время насчитывается более сотни подобных проектов, большинство из которых построены на платформе BOINC [11].

Распределенные вычисления в рамках веб-браузера. В отличие от предыдущего направления, где для участия в распределенном проекте на компьютере требуется установить и настроить специализированное ПО, в данном случае участнику будет достаточно зайти на специализированный веб-сайт для того, чтобы подключиться к расчету. Расчет будет проходить в контейнере веб-браузера, клиентская вычислительная часть может быть написана на Java/JavaScript, Microsoft Silverlight или даже Adobe Flash. Преимущества такого подхода исключительно весомые – крайне широкая аудитория охвата, стандартизированные платформы для написания приложений, автоматическое решение существенной части вопросов, связанной с безопасностью и изоляцией работы приложения на клиентской машине. Безусловно, есть и недостатки, наиболее существенный из которых – слабая пригодность упомянутых технологий программирования для создания именно вычислительных задач. Интерес к подобному направлению возник буквально несколько лет назад, однако уже сейчас существуют компании, продающие подобные вычислительные ресурсы [12].

Вычисления на мобильных устройствах. Достаточно простая и красивая идея – на руках у пользователей появляется все больше мобильных сетевых устройств – телефонов, коммуникаторов, интернет-планшетов и др., процессоры которых уже могут сравниться по мощности с процессорами настольных компьютеров, скажем, 10-летней давности. Процессоры этих устройств в основном находятся в состоянии простоя, следовательно, их вполне можно было бы использовать для проведения расчетов в те моменты времени, когда они не заняты своей прямой работой. Безусловно, на этом пути вопросов пока больше, чем ответов. Как разработать переносимое приложение с учетом разнообразия имеющихся мобильных платформ? Как обеспечить надежное и безопасное сетевое соединение? Как быть с повышенным расходом аккумуляторной батареи мобильного устройства из-за постоянной загрузки процессора? Эти проблемы еще предстоит решить, и тем не менее публикации на эту тему появляются уже сейчас.

В заключение еще раз хотелось бы подчеркнуть, что концепция распределенных вычислений, появившаяся свыше 20 лет назад, не теряет актуальности и сейчас, с увеличением мощности и доступности традиционных вычислительных систем. Наряду с изначально возникшими направлениями (объединение отдельных вычислительных систем в единое целое, утилизация простаивающих компьютерных ресурсов) в этой области появляются и новые технологии, а следовательно, прикладным специалистам становятся доступны новые методы решения вычислительно сложных задач.

## Список литературы


1. GRIDCLUB.RU: Интернет-портал по грид-технологиям. – URL: http://www.gridclub.ru.

2. The Globus Alliance. – URL: http://www.globus.org.

3. gLite: Lightweight Middleware for Grid Computing. – URL: http://glite.web.cern.ch/glite.

4. UNICORE: Distributed computing and data resources. – URL: http://www.unicore.eu.

5. EGEE Portal: Enabling Grids for E-sciencE. – URL: http://www.eu-egee.org.

6. EGI – European Grid Infrastructure. – URL: http://www.egi.eu.

7. СКИФ-ГРИД: научно-техническая программа Союзного государства. – URL: http://skif-grid.botik.ru.

8. Система метакомпьютинга X-Com. – URL: http://x-com.parallel.ru.

9. SETI@home. – URL: http://setiathome.berkeley.edu.

10. BOINC. – URL: http://boinc.berkeley.edu.

11. Distributed Computing. – URL: http://distributedcomputing.info.

12. Plura Processing. – URL: http://www.pluraprocessing.com.

**[1]Р.А. Степанов, [1,2]А.В. Чупин, [1]П.Г. Фрик**

[1]Институт механики сплошных сред УрО РАН, г. Пермь
[2]Пермский государственный университет

## ГЛОБАЛЬНАЯ ДИНАМО-ВОЛНА В ТОЛСТОМ ТОРЕ

Одним из возможных кандидатов на лабораторное получение эффекта генерации магнитного поля потоками проводящей жидкости является нестационарное динамо в тороидальном канале [2]. Порог генерации зависит от геометрии канала и свойств потока. Так, условия генерации в прямом круговом цилиндре были подробно изучены Пономаренко в 1968 году [1], а в работах, изучающих генерацию в криволинейном канале, отмечались некоторое повышение порога [3, 4] и немонотонная зависимость его от кривизны канала [3, 4].

Рассмотрим тор с конечной магнитной диффузией $\eta$, находящийся в бесконечной среде с такими же свойствами. Тор характеризуется радиусом $r_c$ – расстоянием от оси тора $z$ до центральной окружности и $R$ – радиусом кругового сечения канала.

При выборе внутреннего радиуса тора $R$ в качестве единицы длины, некоторой единицы скорости $U$, а диффузионного времени $R^2/\eta$ в качестве единицы времени безразмерное уравнение для вектора магнитной индукции выглядит так:

$$\frac{\partial \vec{B}}{\partial t} = \mathrm{Rm}\nabla\left(\vec{V}\times\vec{B}\right) - \nabla\left(\nabla\times\vec{B}\right), \qquad (1)$$

где $\mathrm{Rm} = UR/\eta$ – магнитное число Рейнольдса.

Магнитное поле должно быть соленоидальным, т.е. удовлетворять условию $\nabla\vec{B} = 0$, что создаёт значительные сложности при численном решении уравнения (1). В данной работе эта проблема обходится переходом к векторному магнитному потенциалу, определяемому формулой

$$\vec{B} = \nabla\times\vec{A}. \qquad (2)$$

Тогда уравнение (1) преобразуется к виду

$$\frac{\partial \vec{A}}{\partial t} = \mathrm{Rm}\vec{V}\left(\nabla \times \vec{A}\right) + \Delta\vec{A} - \nabla\nabla \times \vec{A} + \nabla\Phi\,, \qquad (3)$$

где $\Phi$ – электрический потенциал. Поскольку $\vec{A}$ определяется с точностью до произвольного потенциального векторного поля, то можно использовать калибровку $\nabla \cdot \vec{A} = \Phi$, что взаимно уничтожает последние два слагаемых в уравнении (3). Введение векторного потенциала позволяет решить также проблему разрыва поля скорости на границе тора. При решении уравнения (3) не возникает необходимости вычисления пространственных производных от поля скорости. Решение для $\vec{A}$ получается гладким, а для $\vec{B}$ – непрерывным. В качестве приближения к граничным условиям для магнитного поля на бесконечности используется условие непротекания электрического тока, которое формулируется в виде

$$A_{\perp} = 0,\ \frac{\partial A_{||}}{\partial n} = 0\,.$$

Конфигурация поля скорости является решающим фактором для возможности динамо-эффекта. Двумерные и осесимметричные поля скорости вообще не могут поддерживать генерацию [4], поэтому в динамо-задачах скорость $\vec{v}$ проводящей жидкости обладает ненулевой спиральностью $H = \vec{v} \cdot \mathrm{rot}\,\vec{v}$. В данной работе рассматривается кинематическая постановка задачи, когда поле скорости считается фиксированным и не возмущается магнитным полем. Такой подход оправдан для определения порога генерации, когда магнитное поле лишь формируется и по амплитуде мало́. Классическим для динамо полем скорости является винтовое поле [1], которое в случае криволинейного канала может задаваться по следующей формуле:

$$\vec{V} = \left\{0, \frac{U\rho\lambda}{(\lambda - \rho\cos\varphi)\sqrt{1+\chi^2}}, \frac{U\chi}{\sqrt{1+\chi^2}}\right\}, \qquad (4)$$

где $\lambda = r_c / R$ – безразмерный параметр кривизны, а $\chi$ – параметр, характеризующий степень закрученности поля скорости.

Множитель $\sqrt{1+\chi^2}$ нормирует поле скорости так, чтобы скорость на границе канала $(\rho=1)$ была порядка $U$ (после обезразмеривания – 1) в предельном случае малой кривизны ($\lambda>>1$). Уравнение (1) записано в координатах $\{\rho,\varphi,\zeta\}$, где $\{\rho,\varphi\}$ – полярные координаты в сечении, а $\zeta$ – линейная координата вдоль канала.

Винтовое динамо представляет собой гармоническую волну

$$\vec{B}(\rho,\varphi)\exp\left(k\zeta+(\gamma+i\omega)t\right), \qquad (5)$$

бегущую вдоль канала и вращающуюся вокруг него. Динамо-волна характеризуется волновым числом вдоль канала $k$, инкрементом роста $\gamma$ и фазовой скоростью $\omega/k$. В нашем случае $\gamma$ является функцией $\mathrm{Rm}$, $\lambda$ и $\chi$; минимальное значение магнитного числа Рейнольдса, при котором будет существовать незатухающее возмущение, т.е. $\gamma\geq 0$, и называется порогом генерации $\mathrm{Rm}^*$. Начальное распределение должно содержать весь спектр возмущений, поэтому в качестве начального условия можно взять случайное распределение магнитного поля, из которого со временем выделяется волна с наибольшим инкрементом роста. В качестве оценки зависимость $\mathrm{Rm}^*(\lambda)$ можно аналитически вычислить, рассматривая динамо в цилиндре с наложенными ограничениями на спектр волновых чисел (рис. 2).

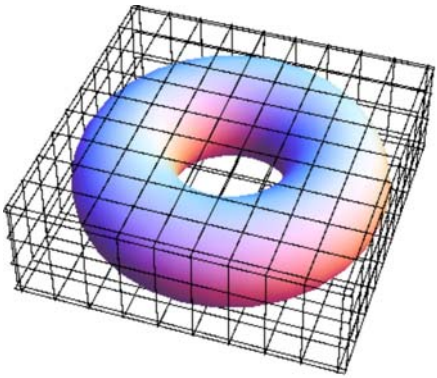


Рис. 1. Схема конечно-разностной сетки

При заданных управляющих параметрах задачи $\mathrm{Rm}$, $\lambda$ и $\chi$ уравнение (2) решалось численно сеточным методом. Для этого тор погружался в прямоугольную сеточную область со сторонами $L\times L\times H$ (рис.1). Размер области выбирался из тех соображений,

что граница не должна оказывать влияния на полученное решение. Это достигалось выбором $L = 2(\lambda + 4)$ и $H = 8$.

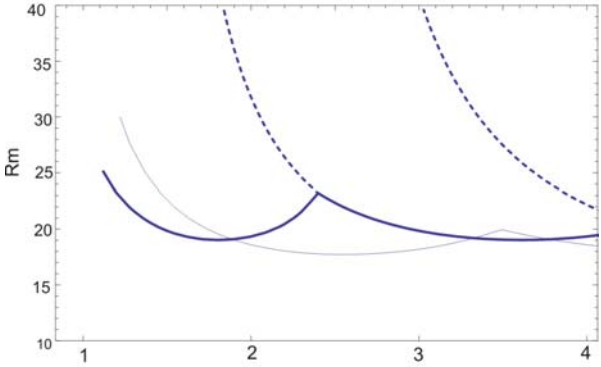
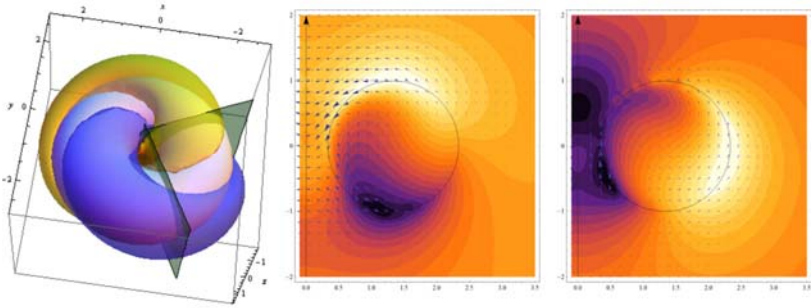



Рис. 2. Зависимость Rm*(λ) для цилиндра при χ = 1 (толстая и пунктирные линии) и χ = 1,3 (тонкая линия)

Интегрирование по времени выполнялось с использованием явной схемы Рунге-Кутта 6-го порядка с адаптивным выбором шага. Пространственные производные аппроксимированы центральными разностями 7-го порядка точности. При постановке граничных условий применялись фиктивные узлы. В расчётах использовались сетки с разрешением до $320 \times 320 \times 160$ узлов. Задача решалась на установление, т.е. до тех пор, пока из начального случайного распределения не выделится мода, амплитуда которой будет меняться по закону [5]; ожидается, что она имеет максимальное $\gamma$. В зависимости от параметров задачи расчёт длится от 10 до 20 единиц времени. Многопроцессорные вычислительные комплексы являются очень эффективным инструментом для решения задач подобного масштаба. Численная схема была адаптирована для выполнения параллельных вычислений с использованием библиотеки MPI. Тестирование алгоритма проводилось на 16-процессорном кластере Института механики сплошных сред УрО РАН, а основные расчёты были выполнены на 1664-процессорном кластере Института математики и механики УрО РАН. При запуске задачи на 32 вычислительных ядрах эффективность использования кластера составляла порядка 75 %,

что соответствует ускорению в 25 раз по сравнению со временем вычисления на одном процессоре.



*а* *б*

Рис. 3. Пространственная конфигурация глобальной динамо-волны (*а*), изолинии магнитного поля в сечениях (*б*)

В работе [4] было показано, что в достаточно толстом торе ($\lambda < 2$) критическим является магнитное поле, имеющее одну гармоническую волну вдоль канала. Как и для динамо в цилиндре, максимум поля наблюдается на границе тора, где сдвиг скорости максимален. Однако в толстом торе появляется горизонтальное магнитное поле, не затухающее при приближении к его внешней оси и имеющее достаточно высокую напряжённость (рис. 3). Если для тонкого тора, как и для цилиндра, характерным масштабом магнитного поля является диаметр сечения, то для моды $n = 1$ в толстом торе масштаб поля становится порядка максимального геометрического размера тора ($\sim 2r_c$). Крупномасштабное поле возникает в результате того, что диаметрально противоположные части тора генерируют магнитные поля, согласованные по пространственной структуре. Отметим, что такая перемычка может наблюдаться только для $n = 1$, так как из соображений симметрии для всех прочих значений $n$, включая $n = 0$ (осесимметричный случай), горизонтальная компонента поля на оси должна быть равна нулю.

Данный вид динамо-волны обладает тем свойством, что её порог генерации остаётся практически постоянным при изменении кривизны тора в пределах $1<\lambda<1{,}8$ (рис. 4). В то время как

увеличение кривизны канала от $\lambda = 3$ до $\lambda = 2$ (когда генерируется волна с $n = 2$) приводит к увеличению разницы между реальным порогом и значением, определяемым из цилиндрического приближения, дальнейшее изменение толщины тора не ухудшает генерацию – критическое магнитное число Рейнольдса, наоборот, стремится к теоретическому значению. Это можно объяснить следующим образом. В случае описанного обезразмеривания максимальный скачок скорости, который и оказывает главное влияние на эффект динамо, находится в точке поверхности канала, определяемой координатами $\varphi = \pi$, $\rho = R$. Поэтому характеристический масштаб, влияющий на генерацию, порядка $r_c - R$, что убывает при увеличении толщины тора как $\lambda$–1. Это уменьшение компенсируется множителем $(\lambda - 1)^{-1}$ в азимутальной компоненте скорости (4), необходимым для обеспечения соленоидальности поля скорости. Таким образом, порог генерации Re, зависящий от их произведения, практически постоянен.

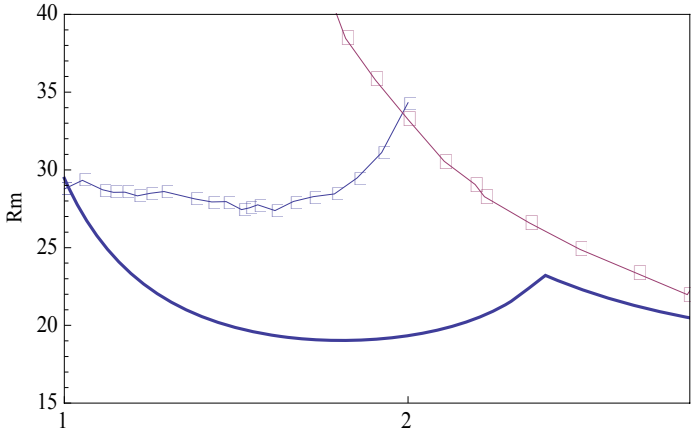



Рис. 4. Пороги итерации при $\chi = 1$, полученные с помощью прямого численного моделирования

Заметим, однако, что это свойство зависит от выбора характерных величин, входящих в безразмерный комплекс Rm. Есть несколько вариантов выбора единицы для скорости. Кроме

используемой в данной работе это могут быть: среднеквадратическая по каналу скорость, максимальный модуль вектора скорости, скорость вдоль канала. Если первый и третий вариант практически не отличаются от рассмотренного, то второй вариант обезразмеривания приводит к неограниченному росту порога генерации при приближении к $\lambda$=1, так как на внутренней кромке канала скорость соответственно растёт.

Представляет интерес зависимость порога генерации от $\chi$, поскольку это единственный параметр, который можно менять при проведении лабораторного эксперимента. Теоретическая зависимость, полученная для цилиндра с периодическими граничными условиями, указывает на возможность снижения порога при заданной геометрии канала путём варьирования данного параметра.

Поскольку именно глобальная динамо-волна обладает более низким порогом генерации, чем магнитные поля с большим количеством волн вдоль канала для толстого тора, имеет смысл определить, в каких пределах кривизны канала она имеет место. Так, при $\chi = 1$ смена мод происходит при $\lambda \approx 2$. Эта граница зависит в первую очередь от закрученности потока $\chi$. Численные эксперименты показывают, что $\chi$=1 не является экстремумом данной зависимости, поэтому, изменяя этот параметр, можно добиться появления низкопороговой глобальной волны и при достаточно тонком торе.

## Список литературы

1. Пономаренко Ю.Б. К теории гидродинамического динамо // ПМТФ. – 1973. – № 6. – С. 47–51.

2. Non-stationary screw flow in a toroidal channel: way to a laboratory dynamo experiment / P. Frick [et al.] // Magnetohydrodynamics. – 2002. – Vol. 38, No. 1–2. – P. 136–155.

3. Dobler W., Frick P., Stepanov R. The screw dynamo in a time-dependent pipe flow // Physical Review E. – 2003. – Vol. 67. – 056309.

4. Степанов Р.А., Фрик П.Г., Чупин А.В. Винтовое динамо в торе // Вычислительная механика сплошных сред. – 2008. – № 1. – С. 109–117.

**В.Н. Стрюков, Ю.Г. Бартенев, В.Г. Басалов, А.М. Варгин, В.М. Вялухин, Н.А. Дмитриев, Д.А. Жуков, В.И. Игрунов, Ю.Н. Корзаков, В.В. Кошелев, В.Н. Лашманов, Ю.В. Логвин, А.Н. Петрик, Г.П. Семенов, Р.М. Шагалиев, А.В. Шатохин, С.Н. Шлыков, Е.Л. Шмаков, С.О. Черных**

РФЯЦ-ВНИИЭФ ИТМФ, г. Саров

## УНИВЕРСАЛЬНАЯ КОМПАКТНАЯ СУПЕРЭВМ

Компактные суперЭВМ – это суперкомпьютеры настольного размера (персональные суперкомпьютеры), не требующие для своей эксплуатации специальных инженерных систем.

В докладе представлено описание компактной супер-ЭВМ (далее КС-ЭВМ), разработанной в РФЯЦ-ВНИИЭФ. Разработка выполнена в рамках проекта «Развитие суперкомпьютеров и грид-технологий», утвержденного решением Комиссии при Президенте Российской Федерации по модернизации и технологическому развитию экономики России (протокол от 02.03.2010 № 9).

КС-ЭВМ является сложным программно-аппаратным комплексом, включающим в себя: вычислительную подсистему, дисковую подсистему, подсистему электропитания, подсистему охлаждения, сервисную подсистему, системное и прикладное программное обеспечение.

Вычислительная подсистема состоит из 3 вычислительных модулей, реализованных на базе многоядерных процессоров AMD Opteron. Суммарная пиковая производительность вычислительной подсистемы составляет 1094 ГФлоп/с. Максимальная ёмкость подсистемы оперативной памяти составляет 1536 ГБ, максимальный объём дисковой подсистемы КС-ЭВМ составляет 24ТБ. Сервисная подсистема КС-ЭВМ выполнена с использованием выделенного сервисного процессора. Разработана специализированная система мониторинга, позволяющая контролировать наиболее важные параметры работы КС-ЭВМ. Для размещения комплектующих КС-ЭВМ разработан корпус, габариты которого составили 325 х 645 Х 725 (Ш х В х Г).

В ходе разработки КС-ЭВМ применены уникальные технические решения, связанные с реализацией высокоскоростной коммуникационной среды Infiniband по бескоммутаторной схеме и подсистемы жидкостного охлаждения процессоров КС-ЭВМ. Это позволило существенно сократить потребляемую мощность КС-ЭВМ (2,2кВт на тесте Linpack), снизить уровень акустического шума (47дБА) и расширить температурный диапазон использования КС-ЭВМ.

**Д.Н. Сузанский**

Военный учебный научный центр ВВС, г. Москва

## СПОСОБ РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ НА МНОГОМАШИННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

Для нахождения оптимальных решений различных задач оптимизации очень часто используется метод ветвей и границ, который является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений [1, 2]. С особенным успехом данный метод применяется для решения задач комбинаторной оптимизации, позволяя уменьшать пространство допустимых решений с помощью эффективной процедуры поиска. Применительно к данному классу задач идея метода может быть описана как нахождение минимума или максимума заданной целевой функции **F(p)** на множестве всех перестановок **P**, при этом множество всех перестановок представляется в виде дерева и поиск оптимального решения идет по ветвям этого дерева вариантов.

Таким образом, при использовании этого метода важно выбрать:

– процедуру ветвления;

– процедуру нахождения оценки (границ).

Процедура ветвления состоит в разбиении области допустимых решений на подобласти меньших размеров. Для ее реализации

удобно использовать лексикографическое представление [1, 2] множества всех перестановок **P**. Процедура нахождения оценок («отбраковка» элементов **P**) заключается в поиске верхних и нижних границ для оптимального значения на подобласти допустимых решений. Поиск происходит с использованием функции **F(p)**, вид которой определяется конкретной прикладной задачей.

Лексикографическое представление множества всех перестановок удобно представить в виде дерева глубиной **N** (количество ярусов), где **N** – размерность конкретной прикладной задачи, у которого ветвь – это отдельная перестановка. Количество ветвей у этого дерева – **N!**. Кроме этого, процедура нахождения оценки (границы) для любой ветви дерева одинаковая и не изменяется.

Описанный способ организации ветвления позволяет успешно реализовать данный метод на многомашинной вычислительной системе, разбив все дерево перестановок на количество машин в системе. При этом на каждой машине в системе решается однотипная задача. Так как области поиска (части дерева перестановок) у машин не пересекаются, все машины могут работать практически в асинхронном режиме. Для увеличения быстродействия всей системы в целом необходимо ввести обмен данными (минимумом или максимумом целевой функции **F(п)**) между машинами.

Таким образом, на каждой машине вычислительной системы необходимо иметь процедуры:

– нахождения границы;

– отправки минимума (максимума) целевой функции;

– порождения очередной ветви дерева перестановок;

– приема минимума (максимума) целевой функции.

При реализации упомянутых процедур возникают тонкие моменты только при программировании четвертой процедуры (приема минимума (максимума) целевой функции) и как следствие – третьей процедуры (порождения очередной ветви дерева перестановок). Так, при приеме минимума (максимума) целевой функции необходимо вводить синхронизацию при доступе к переменной, которая содержит этот минимум (максимум), а при

порождении очередной ветви дерева перестановок – необходимо точно определить ярус дерева, на котором происходит превышение минимума (максимума) целевой функции.

Четкая и правильная реализация перечисленных процедур позволяет добиться высокой производительности многомашинной вычислительной системы.

Приведенные выше положения были реализованы при решении задачи комбинаторной оптимизации – «задачи о назначениях» [1, 2]. Суть данной задачи заключается в том, чтобы при известной матрице затрат $\mathbf{C} = \left\| \mathbf{c_{i,j}} \right\|$ ($\mathbf{c_{i,j}}$ – затраты, связанные с назначением *i*-го ресурса на *j*-й объект), необходимо найти такую перестановку ($p_1, p_2, \ldots, p_N$), которая минимизирует сумму

$$\sum_{i=1}^{N} \mathbf{C}_{\mathbf{ip_i}}. \qquad (1)$$

Данная задача была реализована на многомашинной вычислительной системе при помощи которой исследовались вычислительные характеристики метода ветвей и границ. Для этого использовался коэффициент повышения быстродействия $\boldsymbol{\sigma}$ многомашинной вычислительной системы [3], который определяется как

$$\boldsymbol{\sigma} = \frac{\boldsymbol{t}_1}{\boldsymbol{t}_n}, \qquad (2)$$

где $t_1$ – время вычисления задачи на одной вычислительной машине, $t_п$ – время вычисления задачи на многомашинной вычислительной системе.

Указанный коэффициент позволяет оценить скорость выполнения задачи на многопроцессорной вычислительной системе.

При исследовании быстродействия рассматривалась задача о назначениях размерности $N = 12$. Матрицы стоимости заполнялись случайными числами, подчиненными нормальному закону распределения. Использовалась вычислительная система, состоящая из трех машин. Засекались соответствующие времена выполнений. В результате можно сказать, что коэффициент повышения быстродействия для системы из трех машин в основном варьируется в пределах $\boldsymbol{\sigma}$ = 1,9 ... 2,8. При этом на значение

данного коэффициента сильно влияет дисперсия элементов матрицы стоимости.

### Список литературы

1. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. – М.: Мир, 1980.

2. Романовский И.В. Алгоритмы решения экстремальных задач. – М.: Наука, 1977.

3. Головкин Б.А. Расчет характеристик и планирование параллельных вычислительных процессов. – М.: Радио и связь, 1983.

**А.В. Сысоев**

Нижегородский государственный университет им. Н.И. Лобачевского

## МОДЕЛИРОВАНИЕ ПРОЦЕССА ПАРАЛЛЕЛЬНОГО ГЛОБАЛЬНОГО ПОИСКА

Специалист-прикладник, ставящий задачу рационального выбора, имеет дело с интересующим его объектом исследований, характеризуемым некоторым выделенным набором параметров и свойствами, которые определенным образом зависят от этих параметров, и целью рационального выбора. Специфицировав характер этих зависимостей в виде набора функциональных характеристик, заданных на пространстве параметров, постановщик получает *модель объекта*, с которой далее и работает специалист по решению задач глобально-оптимального выбора. Задача последнего – формализовать цель рационального выбора, построив на множестве функционалов критерий эффективности и определив, каким ограничениям должны удовлетворять функционалы. При этом типичной является ситуация, когда выбор рационального варианта многокритериален, а значит, критерий эффективности будет векторным. Ситуация усложняется тем, что отнюдь не всегда очевидно, какие функционалы взять в качестве критериев, а сами частные критерии обычно противоречивы. Априорной информации для принятия такого

решения обычно недостаточно, она появляется и накапливается в процессе исследований, приводя к необходимости уточнения постановок задач (смене выбора критериев и ограничений), то есть возникновению *набора задач оптимального выбора*, решать которые требуется *совместно*.

**Модель объекта исследований.** Как отмечалось выше, формализация исходной задачи рационального выбора ведет, в общем случае, к появлению набора задач оптимального выбора, построенных на основе выбранной и зафиксированной модели объекта исследований.

В процессе постановки конкретных задач оптимального выбора может изменяться векторный критерий эффективности (некоторые частные критерии могут переводиться в функциональные ограничения и наоборот), могут меняться допуска на значения функциональных характеристик, часть параметров может быть зафиксирована или сведена к дискретному набору значений, может быть изменена рассматриваемая область изменения параметров.

В работе предложен способ формального описания указанных постановок на основе математической модели объекта глобально-оптимального выбора, построить которую означает задать:

– вектор параметров $\overline{y}$ и его *размерность* $N$;

– векторы $\overline{a}$ и $\overline{b}$, определяющие гиперинтервал $\overline{D}$ возможных значений вектора $\overline{y}$;

– вектор-функцию характеристик (*функционалы*) $w(\overline{y})$ и ее размерность $n$.

При этом для численного решения на основе построенной модели объекта формируются постановки задач оптимального выбора. Формирование таких постановок происходит в два этапа.

На первом этапе определяются множество $F$, задающее векторный критерий эффективности $f$, и множество $G$, определяющее ограничения $g$.

На втором этапе для сформированного варианта множеств $F$ и $G$ уточняется область поиска и, при необходимости, выделяются параметры, принимающие дискретные значения, то есть

вектор $\bar{y}$ записывается в двухкомпонентном виде $\bar{y} = (y, u)$, а также задаются:

– векторы *a* и *b*, определяющие гиперинтервал *D* возможных значений вектора *y*;

– множество наборов значений дискретных параметров *U* и его размерность *M*;

– правило $u(\tau)$, сопоставляющее индексу $\tau$, $1 \le \tau \le S$, допустимый кортеж *u*.

Как следствие, для отражения всех возможных вариантов в работе предложено иерархическое описание объекта исследований (рис. 1), включающее в себя три базовых элемента:

*объект оптимизации*

$$O = < \bar{y}, \bar{D}, w(\bar{y}) > - \quad (1)$$

представляет модель объекта исследований, описывается набором параметров, областью изменения их значений и функционалами;

*задание оптимизации*

$$Z = < F, f, G, g, q > - \quad (2)$$

представляет первую часть постановки задачи оптимального выбора, описывается множеством критериев и ограничений;

*задача оптимизации*

$$z = < y, D, u, U > - \quad (3)$$

представляет вторую часть постановки задачи оптимального выбора, описывается областью поиска.

Расширим представленную в (1) модель объекта исследований на случай, когда некоторые входящие в нее функционалы могут быть вычислимы не во всех точках области $\bar{D}$. Указанное обстоятельство нередко встречается в практических задачах, когда при определенных сочетаниях параметров модели, задаваемых, например, компоновочными ограничениями, объект исследований становится неработоспособным или нереализуемым. В этом случае, подав «на вход» модели точку $\bar{y}$, «на выходе» мы получим только часть вычисленных компонент вектор-функции $w(\bar{y})$, таким образом, описание объекта O должно

быть дополнено вектором признаков $\xi^w(\bar{y})$, указывающим для каждого функционала $w_i(\bar{y})$ вычислен он или нет. Назовем такую модель *частично вычислимой*, а модель, для описания которой достаточно (1), – *полностью вычислимой*.

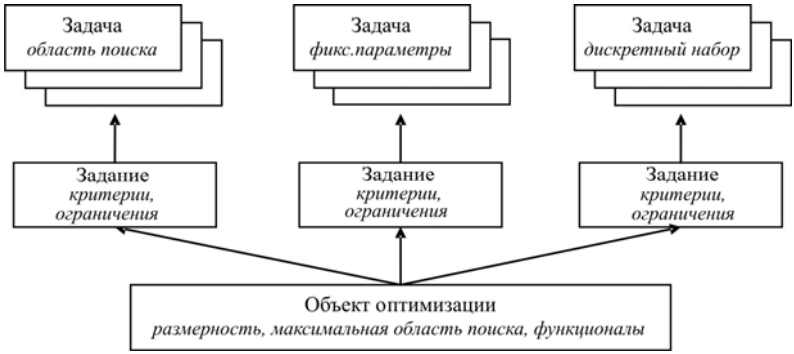



Рис. 1. Иерархическое описание объекта исследований

Для частично вычислимой модели также необходимо доопределить понятие задания оптимизации, дополнив его векторами признаков $\xi^f$ для описания вычисленных критериев и $\xi^g$ – для описания вычисленных ограничений.

**Поисковая информация и оптимизационные данные.** В процессе численного решения конкретной постановки задачи оптимального выбора выполняются итерации, приводящие в конечном счете к выбору очередной точки испытания $\bar{y} = (y, u)$ и вычислению в этой точке значений функционалов $w_1(\bar{y}), \ldots, w_n(\bar{y})$. Таким образом, для полностью вычислимой модели происходит накопление множества

$$\Omega = \left\{ (\bar{y}^i, w(\bar{y}^i)) : \ 0 \le i \le k \right\}, \tag{4}$$

а для частично вычислимой – множества

$$\Omega = \left\{ \left( \bar{y}^i, w(\bar{y}^i), \xi^w(\bar{y}^i) \right) : \ 0 \le i \le k \right\}, \tag{5}$$

где $k$ – число выполненных итераций.

Будем называть множество $\Omega$ *поисковой информацией*.

Как уже отмечалось, конкретная постановка задачи оптимального выбора строится на основе модели объекта формированием во введенных терминах *задания оптимизации* (выделением в множестве функционалов критериев эффективности и ограничений) и *задачи оптимизации* (определением области поиска, фиксацией или сведением к дискретному набору значений части параметров).

Переход от модели объекта к заданию оптимизации означает преобразование множества Ω в множество

$$\omega = \left\{ \left( \overline{y}^i, f(\overline{y}^i), \xi^f(\overline{y}^i), g(\overline{y}^i), \xi^g(\overline{y}^i) \right) : \ 0 \le i \le k \right\}. \qquad (6)$$

Множество ω будем называть *оптимизационными данными*. При необходимости может быть обеспечено как одновременное хранение множеств Ω и ω, так и хранение только множества ω, поскольку по значениям критериев *f*, ограничений *g* и признакам $\xi^f$ и $\xi^g$ значения функционалов *w* и признаки $\xi^w$ могут быть восстановлены.

Дальнейший переход от задания к задаче оптимизации меняет лишь область допустимых значений для точек $\overline{y}^i$ и преобразует оптимизационные данные к виду

$$\omega = \left\{ \left( y^i, u^i, f(y^i, u^i), \xi^f(y^i, u^i), g(y^i, u^i), \xi^g(y^i, u^i) \right) : 0 \le i \le k \right\} \quad (7)$$

Введенные понятия поисковой информации и оптимизационных данных позволяет дополнить иерархическое описание объекта исследований и построить *информационную модель объекта исследований* (рис. 2).

При этом оптимизационные данные, соответствующие разным задачам/заданиям, взаимосвязаны и могут, как отмечено выше, быть использованы при совместном решении набора постановок задач оптимизации.

**Общая схема процесса глобального поиска.** Построенное *информационное описание* задачи рационального выбора в терминах объекта оптимизации, набора заданий и задач опти-

мизации, а также поисковой информации и оптимизационных данных позволяет сформулировать общую схему процесса глобального поиска, включающую следующие этапы:

1) построение задания оптимизации, подготовка оптимизационных данных на основе решенных ранее постановок;

2) построение задачи оптимизации, подготовка оптимизационных данных на основе решенных ранее постановок;

3) выбор метода глобального поиска;

4) выполнение численного решения до получения оценки глобального оптимума, накопление поисковой информации по объекту и оптимизационных данных по текущей постановке;

5) анализ полученного решения и, при необходимости, смена постановки либо переходом к этапу 2 (изменение области поиска), либо переходом к этапу 1 (изменение ограничений и критериев).

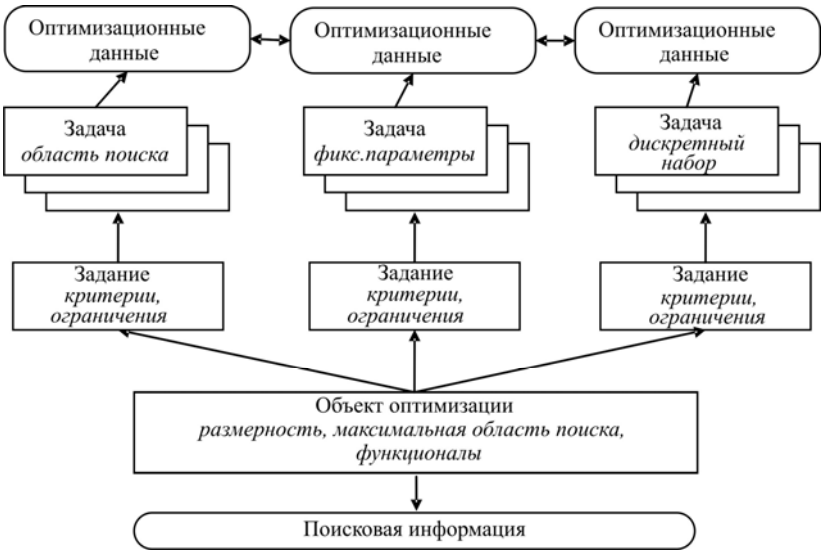



Рис. 2. Информационная модель объекта исследований

Процесс глобального поиска решения исходной задачи рационального выбора итерационен и на практике включает в себя неоднократные уточнения постановок конкретных задач оптимизации по мере накопления информации. Возможность повторного использования накопленных оптимизационных данных при смене постановок означает существенное повышение эффективности поиска решения не только в них, но и во всей задаче рационального выбора в целом.

**Параллельные вычисления в процессе глобального поиска.** Представленная в работе схема процесса глобального поиска обладает принципиальным параллелизмом на этапах с 1-го по 4-й.

Построив на этапе 1 набор заданий оптимизации и на этапе 2 для каждого из них набор задач оптимизации, далее на этапе 3 все эти постановки можно решать одновременно (параллельно) при наличии достаточного числа исполняющих устройств (процессоров, ядер, ускорителей). Сам процесс численного решения выбранным методом глобального поиска (этап 4) также может быть распараллелен на большинстве входящих в него шагов.

## Список литературы

1. Strongin R.G., SergeyevYa.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000.

2. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация: учеб. пособие. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2007.

**М.А. Ташкинов**

Пермский государственный технический университет

# ВЫЧИСЛЕНИЕ СТАТИСТИЧЕСКИХ ХАРАКТЕРИСТИК ПОЛЕЙ НАПРЯЖЕНИЙ И ДЕФОРМАЦИЙ В КОМПОЗИЦИОННЫХ МАТЕРИАЛАХ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ПРОГРАММНОЙ СРЕДЕ MATHEMATICA 7

Механика композиционных материалов является развивающейся научной отраслью. Возможность создавать конструкции и материалы с заданными свойствами для конкретных приложений является крайне важной и актуальной для промышленности в наше время. Адекватное описание реальных стохастических структур, исследование полей деформирования с последующей оценкой механизмов разрушения являются актуальными проблемами механики композитов, которые сопровождаются необходимостью ёмких и сложных вычислений.

Структурная модель исследуемого материала представляет собой двухфазный композит, состоящий из матрицы и хаотически расположенных включений. В качестве включений рассматриваются сферы или эллипсоиды (рисунок). Синтез структур производится с помощью различных методик, позволяющих получать случайные структуры с такими заданными заранее характеристиками, как объемная доля включений, размер включений и матрицы, количество включений.

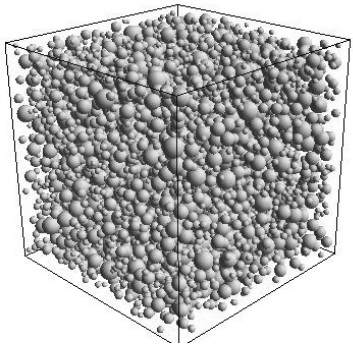


Рис. Синтезированная структура композита. Объемная доля включений –30 %

Для исследования структурных полей деформирования и расчета эффективных характеристик микронеоднородных сред решается стохастическая краевая задача, уравнения и граничные

условия которой содержат случайные величины. Статистическая информация о структуре в виде многоточечных моментных функций получается с использованием имитационного моделирования.

Краевая задача сводится к интегродифференциальному уравнению, решение которого ищется с использованием метода функций Грина. Решение ищется относительно пульсаций перемещений и имеет вид

$$\frac{\partial U_i'(\vec{r})}{\partial x_\alpha} = \int_{V_1} \frac{\partial G_{ij}(\vec{r},\vec{r}_1)}{\partial x_\alpha} \frac{\partial}{\partial x_{1n}} \left( \left( < C_{jnkl}(\vec{r}_1) > + C'_{jnkl}(\vec{r}_1) \right) e_{kl} + C'_{jnkl}(\vec{r}_1) U'_{k,l}(\vec{r}_1) \right) dV_1. \quad (1)$$

Так как искомая величина $U_i'$ содержится как в левой, так и в правой части уравнения, для ее отыскания используется метод последовательных приближений. В данной работе исследуется метод построения приближений высших порядков данной задачи.

Статистические характеристики полей напряжений и деформаций, а именно моменты $< \varepsilon_{ij} >$, $< \sigma_{ij} >$, $< \varepsilon'_{ij}(\vec{r}) \varepsilon'_{\alpha\beta}(\vec{r}) >$, $< \sigma'_{ij}(\vec{r}) \sigma'_{\alpha\beta}(\vec{r}) >$, записываются через аналитические выражения, содержащие функцию Грина, смешанные моменты типа $< \lambda'(\vec{r}) \lambda'(\vec{r}_1) U'_{m,n}(\vec{r}) >$, $< \lambda'(\vec{r}) \varepsilon'_{ij}(\vec{r}) >$, $< \lambda'(\vec{r}) \sigma'_{ij}(\vec{r}) >$ и в случае второго приближения представляют собой сумму шестимерных интегралов. Интегрирование проводится с помощью методов Симпсона или Монте-Карло, по всей области статистической зависимости в сферических координатах.

Все этапы решения задачи, а именно синтез структур, построение моментных функций и вычисление статистических характеристик автоматизированы и реализованы в среде Mathematica 7.

С увеличением порядка искомого приближения растет порядок моментных функций, необходимых для получения статистических характеристик, а также порядок интегралов, которые необходимо вычислять. Значительное увеличение времени счета в таком случае приводит к потребности в использовании параллельных вычислений. Для многомерных интегралов с помощью встроенных функций Mathematica 7 составлен алгоритм, позволяющий разбивать циклическое численное вычисление многомерных интегралов на части, каждая из которых вычисляется на отдельном процессоре. Каждая такая часть представляет собой

выражение с определенной комбинацией индексов подынтегральных тензоров.

Расчеты проводятся в Центре высокопроизводительных вычислительных систем Пермского государственного технического университета.

Таким образом, реализована модель, входными данными которой являются геометрия структуры, физические характеристики материала и условия нагружения, а выходными – статистические характеристики полей напряжений и деформаций.

**А.Б. Терентьев**

Научно-исследовательский центр специальных вычислительных технологий,
г. Нижний Новгород

## АППАРАТНО-ПРОГРАММНАЯ ПЛАТФОРМА CUBLIC ДЛЯ НАУЧНОГО МОДЕЛИРОВАНИЯ

В настоящее время всё сильнее ощущается необходимость в высокопроизводительных вычислениях над большими объёмами данных, а следовательно возрастает интерес к средствам, позволяющим реализовать такие вычисления. Одним из таких средств являются современные графические процессоры, имеющие наилучшее на сегодняшний день соотношение цены, производительности и энергопотребления. Однако сложность разработки соответствующего эффективного программного обеспечения не позволяет этой технологии получить широкое распространение.

Целью и содержанием проекта Cublic является создание вычислительной платформы, позволяющей решать вычислительные задачи практически любой сложности без привлечения профессиональных программистов.

Аппаратное обеспечение проекта базируется на графических процессорах. Программная платформа включает в себя три основных компонента: редактор алгоритмов, среда выполнения и система удалённого доступа через Internet. Редактор алгоритмов позволяет описывать любой вычислительный алгоритм как диаграмму, включающую предопределённые вычислительные примитивы (блоки), их свойства и связи. Блоки могут группиро-

ваться в модули, которые, в свою очередь, могут быть использованы наравне с блоками на любом уровне вложенности. Процесс низкоуровневого распараллеливания задачи скрыт от пользователя и выполняется автоматически в рамках среды выполнения. Таким образом, для решения прикладной задачи от специалиста-предметника требуется только владение основами математического моделирования.

В системе реализованы следующие виды элементарных блоков: блоки синхронизации, вектор-матричных операций, генерации, элементарных функций, решения СЛАУ, визуализации и блоки специализированных высокоуровневых алгоритмов. На базе вышеперечисленных блоков реализуются модули, которые можно применять в решении сложных предметных задач в том числе и в области биологии.

Совместно со специалистами, занимающимися изучением биологических процессов головного мозга, с помощью Cublic были решены следующие задачи: моделирование внутреннего взаимодействия в нейральных сетях, динамики роста нейральных сетей, межнейронного взаимодействия с учётом периода рефрактерности, длины аксона, пластичности, а также реализованы алгоритмы исследования зависимости параметров нейральной сети от структуры её внутренних связей.

С помощью вычислительной платформы Cublic к настоящему моменту был также решён ряд предметных задач: построение фрактальных множеств Жюлиа и Мандельброта, «пыли» Фату, смоделирована нейронная сеть с учетом различных параметров нейронов и их связей. В рамках совместных НИР проводятся реальные расчеты в областях физики живых систем, атмосферного анализа, акустики.

Одним из основных преимуществ разрабатываемой платформы является её эффективная работа на различных гибридных архитектурах, начиная с персонального компьютера с графическими картами и заканчивая новейшими разработками компании «Т-платформы» TB2-TL(tm) и суперкомпьютерами из первой десятки Top500 – Tianhe-1 (7-е место) на базе графических вычислителей ATI Radeon HD 4870 и Nebulae (2-е место) на базе NVidia Tesla C2050.

Текущая информация о проекте представлена на сайте http://hopcomp.net.

**М.В. Усанин, А.М. Сипатов, Л.Ю. Гомзиков**

ОАО «Авиадвигатели», г. Пермь

## ПРИМЕНЕНИЕ СХЕМ ВЫСОКОГО ПОРЯДКА ДЛЯ РЕШЕНИЯ ТРЕХМЕРНЫХ УРАВНЕНИЙ ЭЙЛЕРА И НАВЬЕ–СТОКСА

В настоящее время для решения проблем аэродинамического профилирования проточной части турбореактивного двигателя наряду с экспериментальным широко используют вычислительный подход. Применение вычислительного подхода позволяет значительно снизить материальные и временные затраты на разработку, а получаемая информация позволяет проводить более подробный анализ. Сегодня вычислительный подход является основным инструментом для аэродинамического проектирования турбомашин. С его помощью решают практически весь спектр задач динамики газа. Для решения задач газовой динамики существуют коммерческие программные продукты, например CFX, FLUENT и др. Как правило, в этих пакетах реализованы схемы не более чем второго порядка точности по пространству и времени, основанные на методах контрольного объема совместно с противопоточной схемой (Upwind) и неявной аппроксимацией по времени. Они с успехом могут использоваться для решения стационарных задач, но для расчета нестационарных и волновых процессов малопригодны. Это связано с относительно высокой численной диссипацией и дисперсией указанных выше расчетных схем. Важной и актуальной проблемой в вычислительной газовой динамике и аэроакустике является разработка и внедрение в процесс проектирования новых низкодиссипативных методов расчета, позволяющих получать более точные решения для нестационарных процессов. Данная работа является промежуточным этапом на пути создания высокоточного параллельного кода применимого для решения задач нестационарной газовой динамики и аэроакустики.

**Математическая модель.** Для упрощения реализации расчетной схемы в случае криволинейных координат, а также для более эффективной формулировки характеристических граничных условий будем рассматривать трехмерные уравнения Навье-Стокса записанные в неконсервативной форме:

$$\frac{\partial U}{\partial t}+A_x\frac{\partial U}{\partial x}+A_y\frac{\partial U}{\partial y}+A_z\frac{\partial U}{\partial z}=F^{\nu}, \qquad (1)$$

где $A_x=\begin{bmatrix} u & \rho & 0 & 0 & 0 \\ 0 & u & 0 & 0 & 1/\rho \\ 0 & 0 & u & 0 & 0 \\ 0 & 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & 0 & u \end{bmatrix}$, $A_y=\begin{bmatrix} v & 0 & \rho & 0 & 0 \\ 0 & v & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 1/\rho \\ 0 & 0 & 0 & v & 0 \\ 0 & 0 & \gamma\rho & 0 & v \end{bmatrix}$, $A_z=\begin{bmatrix} w & 0 & 0 & \rho & 0 \\ 0 & w & 0 & 0 & 0 \\ 0 & 0 & w & 0 & 0 \\ 0 & 0 & 0 & w & 1/\rho \\ 0 & 0 & 0 & \gamma p & w \end{bmatrix}$

$$F^{\nu}=\begin{bmatrix} 0 \\ \frac{\mu}{\rho}\left[\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\frac{1}{3}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 v}{\partial x\partial y}+\frac{\partial^2 w}{\partial x\partial z}\right)\right] \\ \frac{\mu}{\rho}\left[\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+\frac{1}{3}\left(\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 u}{\partial y\partial x}+\frac{\partial^2 w}{\partial y\partial z}\right)\right] \\ \frac{\mu}{\rho}\left[\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+\frac{1}{3}\left(\frac{\partial^2 w}{\partial z^2}+\frac{\partial^2 u}{\partial x\partial z}+\frac{\partial^2 v}{\partial y\partial z}\right)\right] \\ (\gamma-1)\left[k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)+\frac{4}{3}\mu\left(\frac{\partial u}{\partial x}\left(\frac{\partial u}{\partial x}-\frac{\partial v}{\partial y}\right)+\frac{\partial v}{\partial y}\left(\frac{\partial v}{\partial y}-\frac{\partial w}{\partial z}\right)+\frac{\partial w}{\partial z}\left(\frac{\partial w}{\partial z}-\frac{\partial u}{\partial x}\right)\right)+\right. \\ \left.\mu\left(\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)^2+\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)^2\right)\right] \end{bmatrix}$$

Заметим, что при нулевой правой части выражения (1) мы получим уравнения Эйлера для идеального газа. Будем считать, что молекулярная вязкость $\mu$ зависит от температуры по закону Сазерленда:

$$\mu(T)=C_1\frac{T^{3/2}}{T+C_2},$$

где

$$C_1=1{,}458\cdot10^{-6}\,\text{кг}/(\text{м}\cdot\text{с}\cdot\sqrt{K}),\ C_2=110{,}4\,K.$$

Для возможности проведения расчетов в областях с криволинейными границами, такими как межлопаточные каналы турбомашины, перейдем от физических координат $(x, y, z)$ к расчетным $(\xi, \eta, \zeta)$. В расчетной системе $(\xi, \eta, \zeta)$ сетка предполагается равномерная и шаги сетки постоянны и единичны ($\Delta\xi = \Delta\eta = \Delta\zeta = 1$). Чтобы получить вид системы (1) в расчетной системе координат $(\xi, \eta, \zeta)$, необходимо выразить производные от неизвестных функций в системе $(x, y, z)$ через производные в системе $(\xi, \eta, \zeta)$. Производные в правой части (1) заменяем, используя правило цепочки при вычислении частных производных от сложной функции. В результате получим

$$\frac{\partial U}{\partial \mathrm{t}} + \left( A_x \frac{\partial \xi}{\partial x} + A_y \frac{\partial \xi}{\partial y} \right) \frac{\partial U}{\partial \xi} + \left( A_x \frac{\partial \eta}{\partial x} + A_y \frac{\partial \eta}{\partial y} \right) \frac{\partial U}{\partial \eta} = \Phi^v, \qquad (2)$$

где $\Phi^v$ – правая часть $F^{\mathrm{v}}$ из выражения (1), где пространственные производные в системе $(x, y, z)$ выражены через производные в системе $(\xi, \eta, \zeta)$.

**Численный метод.** Для аппроксимации пространственной производной используется схема DRP (Dispersion Relation Preserving), предложенная C.K. Tam и J.C. Webb. На равномерной сетке с постоянным шагом $\Delta x$ она имеет вид

$$\left( \frac{\partial U}{\partial x} \right)_1 \approx \frac{1}{\Delta x} \sum_{j=-3}^{3} a_j U(x_1 + j\Delta x). \qquad (3)$$

Коэффициенты разностной схемы $a_j$ определяются не только из .условия достижения наибольшего порядка аппроксимации, но и из условия получения дисперсионного соотношения, максимально близкого к соотношению, которое имеет исходное дифференциальное уравнение. В случае симметричного семиточечного шаблона имеется три независимых коэффициента. Два из них выбираются из условий аппроксимации исходного дифференциального уравнения с четвертым порядком точности, а оставшийся ко-

эффициент выбирается из условий минимума дисперсионных ошибок. Коэффициенты схемы приведены в работе [1].

Для интегрирования по времени используется явная двухслойная шестишаговая оптимизированная схема Рунге-Кутты четвертого порядка точности [2].

При решении системы вида

$$\frac{\partial U}{\partial t} = R(U) \tag{4}$$

двухслойную схему можно записать так:

$$\left.\begin{aligned} W_i^j &= \alpha^j W_i^{j-1} + \Delta t R\left(t^j, Q_i^{j-1}\right) \\ Q_i^j &= Q_i^{j-1} + \beta^j W_i^j \end{aligned}\right\} \qquad j = 1,\ldots S, \tag{5}$$

т.е. в памяти ЭВМ хранится только два слоя переменных. Коэффициенты определяются с помощью процедуры оптимизации подобной для схемы по пространству. Используются условия аппроксимации с четвертым порядком и условия минимизация диссипативных и дисперсионных ошибок. Коэффициенты схемы приведены в [2].

Для подавления нефизичных высокочастотных пилообразных пульсаций, присущих центрально разностной схеме, в конце каждого шага по времени к рассчитанным газодинамическим полям применяется фильтр высокого порядка вида:

$$\bar{U}(x_0) = U(x_0) - \sigma_d D_U(x_0), \;\; D_U(x_0) = \sum_{j=-3}^{3} d_j U(x_0 + j\Delta x), \tag{6}$$

где константа демпфирования $\sigma_d \in [0,1]$, а для коэффициентов фильтра выполняются условия $d_j = d_{-j}$, чтобы не вносить дисперсионных ошибок.

При построении оптимизированных фильтров к $D_U(x_0)$ в (10) применяется пространственное преобразование Фурье и проводится оптимизация в пространстве волновых чисел. Коэффициенты $d_j$ приведены в работе [1].

При использовании низкодиссипативных схем для расчета течений со скачками вблизи них появляются нефизичные пульсации, связанные с неустойчивостью Гиббса. Чтобы их подавить

в окрестности точек разрыва, в схему необходимо ввести дополнительную вязкость. Для этого используется специальная процедура фильтрации, описанная в работе [3] (Shock-Capturing Filtering).

**Особенности численной реализации.** Алгоритм строится на адаптивных структурированных многоблочных сетках. Для хранения сетки используется открытый формат CGNS. Зависимые переменные: плотность, компоненты скорости и давление приписываются к центрам ячеек. При реализации алгоритма использовалась среда C++. Для ускорения вычислений на многоядерных и многопроцессорных системах при реализации вычислительного кода для параллельного выполнения циклов использовались директивы и функции надстройки OpenMP. Учет вязких слагаемых в алгоритме производится опционально. Он не включен в основную схему Рунге–Кутты, а реализован в виде дополнительного подшага по времени, который выполняется если во входном файле параметров присутствует соответствующий ключ, предписывающий учет в расчете вязкости. Приведем здесь также условие устойчивости Куранта для течений с доминирующей конвекцией. В расчетной системе координат его можно записать в виде

$$dt = \frac{CFL}{\max\left(|u| + c\sqrt{\left(\frac{\partial\xi}{\partial x}\right)^2 + \left(\frac{\partial\xi}{\partial y}\right)^2 + \left(\frac{\partial\xi}{\partial z}\right)^2}, |v| + c\sqrt{\left(\frac{\partial\eta}{\partial x}\right)^2 + \left(\frac{\partial\eta}{\partial y}\right)^2 + \left(\frac{\partial\eta}{\partial z}\right)^2}, |w| + c\sqrt{\left(\frac{\partial\zeta}{\partial x}\right)^2 + \left(\frac{\partial\zeta}{\partial y}\right)^2 + \left(\frac{\partial\zeta}{\partial z}\right)^2}\right)}, \quad (7)$$

$$CFL \le 0{,}9.$$

Для реализации граничных условий используется метод фиктивных ячеек совместно с методом характеристик. На входе задаются: угол набегающего потока, полное давление и температура, на выходе – статическое давление. На стенках стандартные условия прилипания – для расчетов с вязкостью и условия скольжения – для идеального газа.

**Результаты расчетов тестовых задач.** Для верификации разработанного численного кода на реальной задаче были проведены расчеты течения в межлопаточном канале турбины 11-й стандартной конфигурации (STCF11). Результаты экспериментов и исходная геометрия лопаток свободно доступны на интер-

нет-странице Шведского Королевского Института технологии [4], где проводились эксперименты.

Основные геометрические параметры показаны на рис. 1, *а*. Хорда лопатки $c$ = 77,8 мм, период канала $\tau$ = 56,55 мм по среднему радиусу, угол установки лопаток $\gamma = 40{,}85^{o}$, расстояние до датчиков $e_1$ = 18 мм, $e_1$ = 35 мм, высота лопатки 40 мм, радиус втулки 160 мм, радиус периферии 200 мм. Угол между лопатками $18^{o}$.

Вид расчетной сетки показан на рис. 1, *б*. Сетка состоит из пяти блоков. Общее количество узлов сетки во всех блоках 234025. Сетка сгущалась к поверхности лопатки и к стенкам проточной части. Высота пристеночной ячейки 0,1 мм.

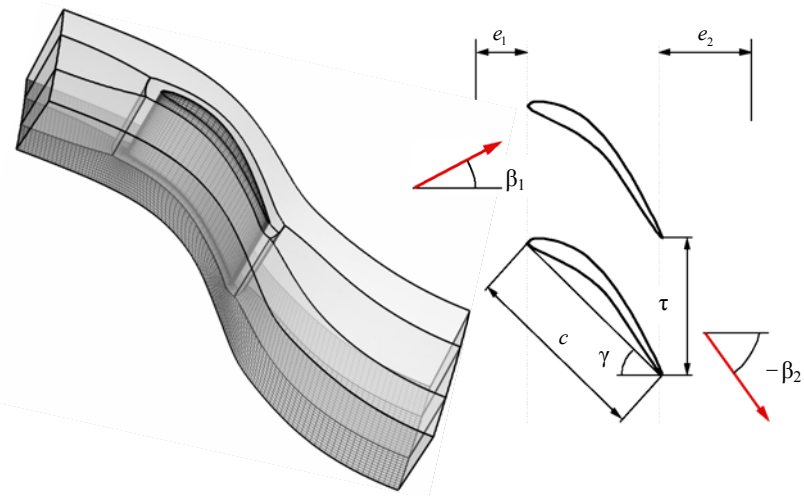
Параметры течения, задаваемые на входе и на выходе расчетной области приведены в таблице.

**Граничные условия для расчета 11-й стандартной конфигурации**

| Режим | 100 | 200 |
|---|---|---|
| Полное давление на входе, Па | 124 400 | 226 080 |
| Полная температура на входе, К | 300 | 400 |
| Угол набегающего потока $\beta_1$, град. | 16,1 | 31,8 |
| Статическое давление на выходе, Па | 91 700 | 115 060 |

Рассматривалось два режима течения. Первый (режим 100) соответствует расчетному режиму обтекания без появления скачков в межлопаточном канале, второй режим (режим 200) соответствует сверхзвуковому обтеканию профиля.

На рис. 2 представлены распределения изоэнтропического числа Маха на среднем радиусе лопатки, полученные в расчетах режима 100 без учета вязкости и при ее учете в сравнении с экспериментальными данными. Как можно видеть из графиков, полученные результаты близки к эксперименту. На корыте лопатки оба расчета дали практически одинаковые результаты, на спинке – результаты вязкого расчета ближе к экспериментальным данным.




*а* *б*

Рис. 1. Геометрия лопатки 11-й стандартной конфигурации (*а*) и вид расчетной сетки (*б*)

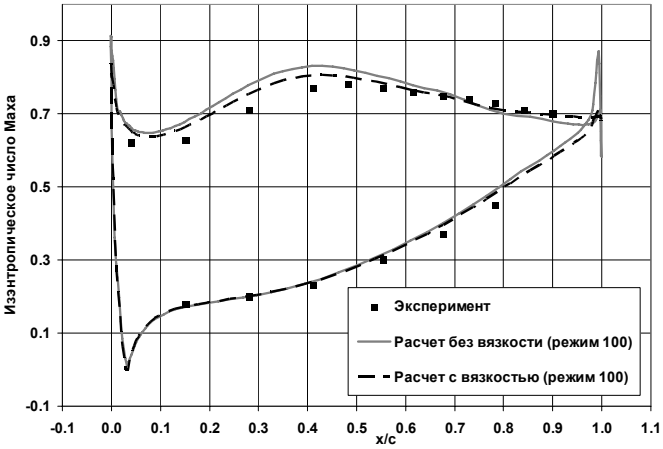



Рис. 2. Распределение изоэнтропического числа Маха на среднем радиусе лопатки

На рис. 3 представлены аналогичные результаты, полученные в расчетах режима 200. Из графиков опять можно увидеть, что полученные результаты близки к эксперименту. Одна-

ко соответствие несколько хуже, чем в дозвуковом случае. В вязком расчете отрывная зона вблизи входной кромки на спинке лопатки получилась больше, чем в эксперименте.

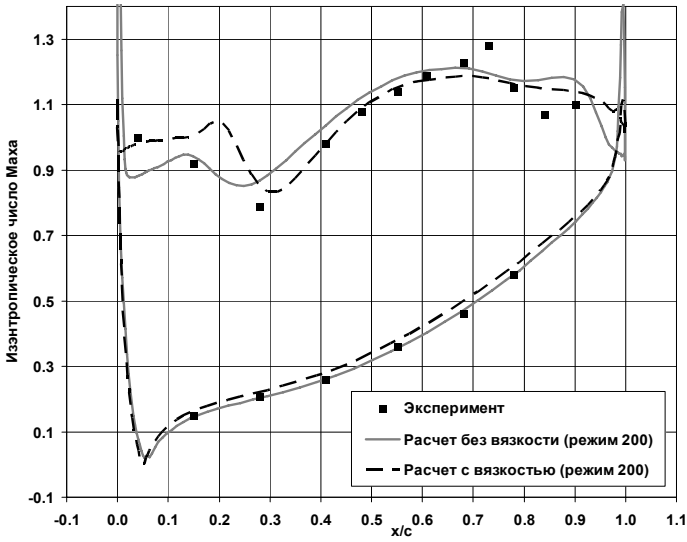



Рис. 3. Распределение изоэнтропического числа Маха на среднем радиусе лопатки

Таким образом, в результате проведенной верификации получено адекватное согласование расчетных и экспериментальных данных по статическому распределению изоэнтропического числа Маха вдоль профиля лопатки.

## Список литературы

1. Tam C. and Webb J., Dispersion–Relation–Preserving Finite Difference Schemes for Computational Aeroacoustics // Journal of Computational Physics. – 1993. – Vol. 107. – P. 262–281.

2. Berland J., Bogey C., Bailly C., Optimized explicit schemes: matching and boundary schemes and 4$^{th}$-order Runge-Kutta algorithm// AIAA 2004–2814.

3. Bogey C., de Cacqueray N., Bailly C., Self-adjusting shock-capturing spatial filtering for high-order non-linear computations // AIAA 2008–2968, 14th AIAA/CEAS Aeroacoustics Conference (29th AIAA Aeroacoustics Conference, 5–7 May 2008, Vancouver, British Columbia Canada.

4. http://www.energy.kth.se/proj/projects/Markus%20Joecker/STCF.

**А.Н. Фирсов**

Пермский государственный университет

## ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ В ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

В настоящее время все острее встают вопросы обеспечения отказоустойчивости информационных систем (ИС). Этому есть несколько причин. Во-первых, системы с каждым годом усложняются, включают в себя все большее количество компонентов, а следовательно, вероятность того, что в одном из компонентов системы произойдет сбой, увеличивается. Это может привести к отказу всей системы, если она не спроектирована с учетом такой возможности. Во-вторых, информационные системы проникают во все области жизнедеятельности, люди все больше полагаются на ИС, все больше зависят от них, поэтому последствия отказа ИС становятся все более и более критичными, если не катастрофическими.

Византийская модель сбоев [1] (или модель произвольных сбоев) является наиболее общей, включающей в себя все остальные, моделью. Она не накладывает никаких ограничений на действия сбойных процессоров и, благодаря этому, позволяет моделировать такие типы сбоев, как отказы оборудования, потеря связи, неисправность оборудования, программные сбои, ошибки операторов и действия злоумышленников.

Особый интерес представляют асинхронные распределенные системы, так как они не накладывают никаких временных ограничений и благодаря этому идеально подходят для работы в сети Интер-

нет. Наиболее универсальной моделью вычислительного узла является машина с конечным числом состояний (state machine).

В данной работе рассматриваются асинхронные распределенные системы, которые, во-первых, устойчивы к произвольным отказам, во-вторых, позволяют выполнять прикладные программы, требующие сохранять свое состояние между обращениями к ним, в-третьих, обеспечивают отказоустойчивость на программном уровне без необходимости в какой-либо специальной аппаратуре и, в-четвертых, делают все это прозрачно для прикладных программ.

В настоящее время существует несколько таких систем: Rampart [2], SecureRing [3], BFT [4] и т.д. Все они реализуют единственный известный на сегодня программный метод защиты от произвольных сбоев – активную репликацию. Суть этого метода заключается в том, что параллельно выполняются нескольких копий (или, как их еще называют, реплик) одной и той же программы. Отличие между этими системами в основном заключается в стеке протоколов, который обеспечивает согласованную работу всех реплик.

Существующие системы обладают несколькими недостатками, корнем которых является то, что эти системы плохо приспособлены для работы в гетерогенных сетях. Под гетерогенностью сети здесь понимается различная надежность включенных в эту сеть компьютеров, различная скорость и задержка при обмене информацией между компьютерами, различная производительность. Именно такой сетью является Интернет, для работы в котором и создавались приведенные выше отказоустойчивые системы, однако то, что они не учитывают неоднородность, приводит к неэффективной работе этих систем:

- Предоставляемый ими стек протоколов фиксирован, а как показано в работе [6,7], в разных ситуациях эффективным оказывается применение разных протоколов.

- Не предоставляется алгоритм для определения оптимального значения параметров работы системы (частота создания контрольных точек, время ожидания у ненадежных детекторов сбоев и т.д.). А следует заметить, что значения этих параметров существенно влияют на эффективность работы системы [6, 7].

• Не дается оценка надежности получаемого решения. А именно в этом показателе наиболее заинтересован пользователь отказоустойчивых систем, так как абсолютная надежность недостижима.

• Не определяется вклад каждого процессора в общий вычислительный процесс. Создатели приведенных выше систем такую задачу перед собой и не ставили, но, как продемонстрировано платформой BOINC [5], это может оказаться очень полезным.

**Подход к обеспечению отказоустойчивости в гетерогенных системах.** Корнем всех проблем, описанных выше, является то, что, существующие системы обеспечения отказоустойчивости не получают никакой информации о среде, в которой они работают. Под средой выполнения здесь подразумевается следующее:

**1. Сеть**, на основе которой работает отказоустойчивая система. Информация о сети включает:

а) производительность вычислительных узлов;

б) вероятности отказов вычислительных узлов;

в) задержки при передаче сообщений между вычислительными узлами;

г) скорость передачи информации по каналам связи.

**2. Прикладная программа**, бесперебойное выполнение которой требуется обеспечить. О ней необходима следующая информация:

а) время, требующееся на создание контрольной точки;

б) частота групповых рассылок.

**3. Относительная ценность ресурсов**. Работа алгоритма обеспечения отказоустойчивости требует ресурсов. Можно выделить три из них, которые являющиеся наиболее важными:$T_o$ – общее время работы программы, $T_p$ – количество единиц процессорного времени, потраченного на выполнение программы всеми процессорами, $V$ – количество переданных сообщений. Не существует алгоритма обеспечения отказоустойчивости, который является предпочтительным по всем параметрам. Как правило, каждый из протоколов позволяет оптимизировать один параметр за счет другого, например, сэкономить процессорное время за счет передачи большего числа сообщений. Поэтому вопрос об использовании того или иного протокола зависит от степени важности

того или иного ресурса. А эту степень важности может определить только пользователь или система балансировки.

В общем виде относительная степень важности ресурсов задается вектором $(r_1, r_2, ..., r_N)$, компоненты которого должны удовлетворять двум условиям:

1) $r_i \geq 0, i = \overline{1, N}$;

2) $\sum_{i=1}^{N} r_i > 0$.

Имея данные о сети и прикладной программе, можно оценить, сколько ресурсов каждого вида будет потреблять тот или иной алгоритм обеспечения отказоустойчивости в зависимости от параметров его функционирования [7]. Общую же ресурсоемкость $R$ алгоритма $A$ можно вычислить по формуле

$$R^A = \sum_i r_i \cdot R_i^A,$$

где $R_i^A$ – количество потребленных единиц ресурса $i$ при использовании алгоритма обеспечения отказоустойчивости $A$ ($T_o{}^A \equiv R_1^A$, $T_p{}^A \equiv R_2^A$, ...).

При имеющейся оценке ресурсоемкости протоколов обеспечения отказоустойчивости вопрос о поиске оптимальных значений параметров этих протоколов сводится к задаче многомерной глобальной оптимизация функции $R^* = \min_X(R(X))$, где $X$ – вектор параметров некоторого протокола.

Выбор же оптимального алгоритма обеспечения отказоустойчивости сводится к оценке ресурсоемкости всех имеющихся алгоритмов с оптимальными параметрами и выбору того, который обеспечивает минимальную ресурсоемкость.

Кроме как для определения наилучшего алгоритма обеспечения отказоустойчивости и его оптимальных параметров для каждой конкретной ситуации, данные о среде исполнения могут быть использованы в алгоритмах обеспечения отказоустойчивости напрямую. К таким алгоритмам относятся асинхронный протокол надежной групповой рассылки с переменным числом

активных реплик[8] и алгоритм группировки вычислительных узлов для повышения надежности [9].

**3. Реализация и использование.** С целью практического применения идей, методов и алгоритмов, изложенных выше, была реализована распределенная система SFT (Statistical Fault-Tolerance), обеспечивающая устойчивость к произвольным отказам. Более подробное описание архитектуры и некоторых деталей реализации приведено в [10].

При использовании системы SFT принципиально важной является среда, в которой она работает. В зависимости от среды выполнения преимущества от использования системы SFT по сравнению с другими системами могут быть как бесконечно большие (в теории), так их может и не быть вовсе.

В общем случае действует правило: чем больше неоднородность среды выполнения, тем больше получаемый выигрыш. Если же среда гомогенна, то выигрыша в ресурсоемкости не будет, а накладные расходы на сбор информации о среде выполнения могут привести и к отрицательным результатам. Отсюда следует вывод, что систему SFT следует использовать в гетерогенных сетях.

## Список литературы

1. Lamport L., Shostak R. and Pease M. The Byzantine Generals Problem. // ACM Transactions on Programming Languages and Systems –July 1982. –Vol. 4, No. 3. – P. 382–401.

2. Reiter M. The Rampart Toolkit for Building High-Integrity Services // Selected Papers from the International Workshop on Theory and Practice in Distributed Systems –Springer-Verlag, 1994. –P. 99–110

3. Kihlstrom K. P., Moser, L. E. and Mellar-Smith, P. M. The SecureRing group communication system // ACM Transactions on Information and System Security (TISSEC), –November 2001. –Vol. 4, No. 4. –P. 371–406.

4. Castro M. Practical Byzantine Fault Tolerance // Massachusetts Institute of Technology, 2001. PhD thesis. -172 p.

5. Anderson, D.P., Christensen, C. and Allen, B. Designing a runtime system for volunteer computing // Proceedings of the 2006 ACM/IEEE conference on Supercomputing –2006. –Article No. 126

6. Фирсов А.Н. Оптимизация на основе статистических данных асинхронной распределенной системы, устойчивой к произвольным отказам // Параллельные вычислительные технологии (ПаВТ-2009): тр. междунар. науч. конф. (Нижний Новгород, 30 марта – 3 апреля 2009 г.). – Челябинск: Изд-во ЮУрГУ, 2009. – С. 765–771.

7. Фирсов А.Н. Оценка эффективности некоторых оптимизаций протоколов надежной и атомарной групповой рассылки // Вестн. Перм. гос. ун-та. Математика. Механика. Информатика. 2009. – С. 161–168.

8. Фирсов А.Н. Асинхронный протокол надежной групповой рассылки с переменным числом активных реплик // Математика программных систем: межвуз. сб. науч. тр. / Перм. ун-т. – Пермь, 2008. – С. 166– 177.

9. Фирсов А.Н. Повышение надежности гетерогенных распределенных отказоустойчивых систем за счет группировки вычислительных узлов. Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: тр. междунар. суперкомпьютерной конф. (20–25 сентября 2010г., г. Новороссийск). – М.: Изд-во МГУ, 2010. – С. 628–634.

10. Фирсов А.Н. Архитектура распределенной системы устойчивой к произвольным отказам // Высокопроизводительные параллельные вычисления на кластерных системах: материалы IX Междунар. конф.-семинара. – Владимир : Изд-во Владимир. гос. ун-та, 2009. – С. 396–400.

[1*]Э.С. Фомин, [1]Н.А. Алемасов, [2]С.А.Матвиенко
[1] Институт цитологии и генетики СО РАН
[2] Новосибирский государственный университет

# СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ПРОЦЕССОРОВ POWERXCELL8I И INTEL X7560 НА ПРИМЕРЕ ЗАДАЧ МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Метод молекулярной динамики (МД) является методом компьютерного моделирования, позволяющим в течение заданного периода времени проследить эволюцию системы взаимодействующих частиц с помощью численного интегрирования уравнений движения Ньютона [1]. В молекулярной биологии метод МД широко используется для решения задач исследования термостабильности белков, конформационных переходов, транспорта молекул, белкового фолдинга и проч. Программы МД могут служить также тестами производительности различных вычислительных платформ благодаря следующим особенностям метода:

– алгоритмическая простота (используются общеизвестные уравнения Ньютона);

– естественный параллелизм высокой степени (силы, действующие на каждую частицу могут быть рассчитаны независимо, и возможная степень параллелизма по порядку величины сравнима с числом частиц в исследуемой системе);

– большой объем вычислений (запросы реальных задач).

Программы МД работают на всех существующих в мире вычислительных платформах, использующих как стандартные Intel, AMD, Alpha, так и специализированные Cell [2, 3] и GP GPU [4, 5] процессоры. Данная работа посвящена сравнительному анализу эффективности выполнения задач молекулярной динамики на сравнимых по архитектуре серверах, построенных на процессорах Cell8i и X7560. Доступ к серверу, вычислительные узлы которого состоят из двух PowerXCell8i-процессоров, каждый из которых включает 8 SPE ядер, был предоставлен компанией T-platforms [6]. Доступ к серверу, вычислительные узлы которого построены на четырех восьмиядерных Intel X7560-процессорах был предоставлен компанией Intel в рамках

Manycore Testing Lab [7]. Память вычислительных узлов для обоих кластеров организована по схеме NUMA.

## Инструменты и методы

Исходный алгоритм и тесты

Данная работа выполнена с использованием программы молекулярного моделирования MOLKERN [8], использующей силовое поле AMBER [9]. Для снижения вычислительной сложности с $O(N^2)$ до $O(N)$ при расчете парных взаимодействий N частиц область действия потенциалов ограничивается некоторым радиусом $r_{cut}$. Все пары атомов, для которых расстояния между атомами $r < r_{cut}$, помещаются в списки соседей. Для создания списка соседей используется улучшенный метод Верлет-таблицы [10], разделяющей пространственную область системы на ячейки с длиной ребра $r_{cut}$ и с поиском соседей для каждого атома только в ячейке самого атома и в 26 соседних ячейках. Для избежания излишних расчетов расстояний между атомами всевозможных пар соседних ячеек атомы в ячейках упорядочиваются согласно проекции их координат на вектор, соединяющий данные ячейки, как предложено в [11]. В расчетах используется третий закон Ньютона, и никакие взаимодействия не вычисляются дважды, поэтому Верлет-таблица хранит индексы $(i, j)$ только тех пар атомов, для которых выполняется условие $i<j$. Для минимизации числа промахов кэша все данные в памяти компьютера упорядочиваются согласно LCR-алгоритму [12]. Цель упорядочения – обеспечить выполнение принципа локальности, то есть данные, связанные с близкими в пространстве атомами, должны располагаться в памяти компьютера так же недалеко друг от друга.

В расчетах использовалась область 120 x 120 x 120 $A^3$, содержащая 59 тысяч молекул воды, которые помещались в нее случайным образом с плотностью 0,03 молекулы $A^{-3}$. Использовалась SPC-модель воды. Для тестирования производительности использовались кулоновский $1 / r$ и 6-12 потенциал Леннарда – Джонса $(\varepsilon/4) [(\sigma/r)^{12} – (\sigma/r)^6]$. Все потенциалы обрезались при 10 А. Дополнительно использовался потенциал $\mathrm{erfc}(\sqrt{\pi} * r / r_{cut}) / r$, называемый ближним кулоновским потенциалом, совпадающий с кулоновским потенциалом при $r \rightarrow 0$. В начале любого расчета

выполнялось 10 итераций оптимизации геометрии системы. Далее, согласно Больцмановскому распределению все молекулы получали случайные значения скоростей, и выполнялась МД в диапазоне 0,1 ps. Использовался Verlet leap-frog-алгоритм [13] для интегрирования уравнений движения с шагом в 1 *fs*. Во тестах использовался NVT ансамбль при температуре $T = 300$ К.

Процессор PowerXCell8i

Процессор PowerXCell8i [14] содержит 9 ядер. Одно ядро – PPE (PowerPC Processor Element) – процессор архитектуры PowerPC, предназначенный для взаимодействия с операционной системой. Остальные восемь ядер, SPE (Synergistic Processor Element) используются как основные вычислительные устройства. SPE могут напрямую работать только со своей локальной памятью (LS) размером в 256 Кб. Передачи данных происходят по кольцевой шине EIB (Element Interconnect Bus), соединяющей все вычислительные элементы процессора и имеющей пропускную способность более 200 Гб/с. В процессоре есть DMA-контроллер, позволяющий без участия PPE и SPE передавать данные между основной памятью и LS. Каждый SPE имеет 128 128-битных регистров с широким набором SIMD-инструкций.

Сервер PeakCell S содержит два процессора PowerXCell8i с частотой 3.2 ГГц и 16 Гб DDR2 оперативной памяти. Оба процессора системы соединены шиной FlexIO с пропускной способностью в 20 Гб/с.

Особенности реализации кода для процессора Cell

Для минимизации программистских затрат для Cell процессора была выбрана модель реализации, в которой большая часть кода выполнялась на PPE, а на SPE выполнялся только один, наиболее затратный по времени, алгоритм расчета ближних невалентных взаимодействий.

Реализация кода для SPE следовала всем рекомендациям из IBM SDK [15]. Загрузки данных и счет выполнялись на SPE, а PPE выступал в роли диспетчера ресурсов (SPE-центричная модель программирования). Для совмещения операций чтения данных и оперирования с ними использовалась техника двойной буферизации, для ускорения загрузки данных – выравнивание данных на гра-

ницу 128 байт как для приемника, так и для передатчика данных. Реализация доступа ядер SPE в основную память выполнена с помощью библиотеки libspe2, а векторизация и многопоточность – с помощью MASS и Pthreads, соответственно.

Процессор Xeon

Процессор Intel Xeon X7560 на данное время является последней разработкой в серии Intel Xeon-процессоров и основан на микроархитектуре Nehalem. Процессор включает: 8 ядер, работающих на частоте 2,26GHz, L2 кэш с 256KB памяти на ядро, 24MB на L3 кэше (разделяемом между ядрами), поддержку набора инструкций x86-64 и технологии Hyper Threading. Процессоры системы соединены высокоскоростной (до 25.6 GB/s) шиной Intel® Quickpath Interconnect, позволяющей объединять по схеме «каждый с каждым» до 8 процессоров и поддерживающей до 16 слотов 16GB DDR3 памяти на каждый процессор.

Особенности реализации кода для процессора Intel Xeon

Благодаря тому что разработка кода для данного процессора имеет меньшую трудоемкость, чем для Cell-процессора, векторизация и параллелизация делалась для большей части программы, что включало: расчет навалентных взаимодействий, построение списков атомов для ячеек, сортировку атомов в ячейках, фильтрацию пар атомов по расстояниям и по наличию/отсутствию химической связи между ними. Векторизация сделана с помощью SSE и SSE2 intricics.

Параллелизация пакета была сделана с помощью библиотеки BOOST threads. При чтении данных использовалась техника декомпозиции по индексам, то есть каждый процесс выполнял расчеты только с назначенным ему диапазоном рядов Верлет-таблицы. При записи промежуточных результатов расчета сил и избегания конфликтов памяти использовалась техника дублирования данных. В этом случае каждый поток записывал результаты в свой массив данных и в конце шага молекулярной динамики все эти массивы объединялись. В условиях небольшого числа потоков накладные расходы для данной техники находились в пределах 2 %.

**Результаты и обсуждение.** На рисунке представлена в логарифмическом масштабе диаграмма эффективности выполнения программы MOLKERN в зависимости от числа используемых потоков для разных процессоров. Эффективность измеряется в миллиардах пар ($10^9$), обрабатываемых за одну секунду. Результаты представлены для процессоров PowerXCell8i (Cell8i), Intel Quad (Q8400) и Intel Xeon (X7560). На всех процессорах выполнялся алгоритмически единый код, но оптимизированный под ту или иную платформу.

Ранее, в [16], мы сообщали результаты по оптимизации данного кода под Cell-платформу. В [16] была реализована PPE-центричная модель вычислений и максимальное ускорение оптимизированного PPE + SPE кода относительно выполнения неоптимизированного кода на одном PPE составляло 26 раз при расчетах с потенциалом $\mathrm{erfc}(\sqrt{\pi} * r / r_{\mathrm{cut}}) / r$. Данное ускорение достигалось при использовании 8 SPE для счета и 2 PPE для загрузки данных. Масштабируемость приложения была низкой, и производительность не увеличивалась даже при использовании всех 16 SPE. Основная причина была в том, что PPE были перегружены и не успевали подготавливать данные для всех SPE. В [17] была реализована SPE-центричная модель. Максимальное ускорение для расчетов с тем же потенциалом $\mathrm{erfc}(\sqrt{\pi} * r / r_{\mathrm{cut}}) / r$ было достигнуто при использовании всех 16 SPE и составляло 218 раз. При использовании стандартного кулона $1/r$ максимальное ускорение равнялось 63, что весьма близко к результатам других авторов. Например, в реализации CHARMM27 для Cell [2] максимальное достигнутое ускорение расчётов при использовании 8 SPE относительно одного PPE составило 35 раз, что соответствует до 70 раз при использовании сервера с 16 SPE. Для программы NAMD в версии для Cell [3] получено ускорение в 61 раз.

Из диаграммы видно, что масштабируемость кода для Cell-реализации, вплоть до 8 используемых SPE, является идеальной. И только при использовании 16 SPE наблюдаются потери в пределах ~15 % от идеального значения, что обусловлено перегрузкой процессорной шины EIB.

Для Intel-процессоров получается существенно большая производительность при выполнении кода при малом числе потоков. Так, разница в производительности при выполнении кода в одном потоке равна 2,75 раза в пользу процессора Intel. Однако рост производительности при увеличении числа потоков заметно меньше, чем для Cell-процессора, и у кривой существует эффект насыщения. Так, при выполнении кода на двух SPE вместо одного для Cell-процессора рост производительности приложения был равен 1.997, а для Intel процессора при переходе от использования одного ядра к двум ядрам рост составляет всего 1,82 раза. На диаграмме это приводит к разному наклону кривых, прекращению роста производительности при большом числе потоков и сближению эффективности процессоров Cell и Xeon при увеличении числа ядер. Так, разница в производительности счета при использовании 16 потоков равна всего 28 %. Можно предположить, что если бы существовал Cell-процессор с 32 ядрами, то он мог обойти Intel-процессор по производительности. В чем могут быть причины различной масштабируемости выполнения задачи для разных процессоров?

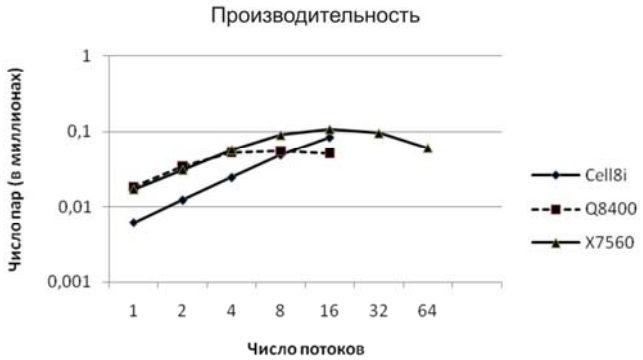



Рис. Диаграмма зависимости эффективности счета от числа потоков для разных процессоров

Возможной причиной низкой масштабируемости приложения при выполнении его на Intel-процессоре может являться специфика алгоритма. В алгоритме для хранения пар соседей

используется Верлет-таблица, объем которой составляет $(4\pi\rho/6)$ $R_{cut}^3$ N элементов, где $\rho$ – плотность частиц в системе, $R_{cut}$ – радиус обрезания потенциала, N – число атомов в системе. Используя параметры теста, указанные выше, можно оценить объем памяти Верлет-таблицы, который составляет величину ~47 Мб. Таким образом, данная таблица не может целиком поместиться в L3 кэш, объем которого вдвое меньше. Поскольку каждое ядро процессора в многопоточном выполнении работает со своей частью Верлет-таблицы и кроме идентификаторов атомов, хранимых в ней, для расчетов необходимо подгружать множество других данных, то ядра начинают «конфликтовать» друг с другом за то, чьи данные должны быть помещены в L3 кэш. Это неизбежно должно приводить к многочисленным промахам кэша и низкой масштабируемости приложения.

Заметим, что в Cell-процессоре ситуация иная и управление загрузкой LS (аналог кэша) целиком возлагается на программиста. В SPE-центричной модели каждое SPE извлекает данные из оперативной памяти независимо друг от друга. И, как показывает наш опыт, это позволяет добиться практически идеальной масштабируемости.

Работа выполнена при поддержке грантов Междисциплинарных интеграционных проектов фундаментальных исследований СО РАН № 26, № 113 и № 119, а также частично Министерства образования и науки РФ (Госконтракт № П857).

## Список литературы

1. Alder B. J., Wainwright T. E. J. Chem. Phys. 1957, 27, 1208.

2. Fabritiis G. D. Comput. Phys. Comm. 2007, 176, No. 11/12. P. 660–664.

3. Shi G., Kindratenko V. 9th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2008).

4. Anderson J. A., Lorenz C. D., Travesset A. J. Comput. Science. 2008, 227, 5342.

5. Liua W., Schmidt B., Vossa G., Müller-Wittig W. Comp. Phys. Comm. 2008, 179, No. 9, 634–641.

6. http://www.t-platforms.ru/

7. http://software.intel.com/en-us/articles/intel-many-core-testing-lab/

8. E. S. Fomin, N. A. Alemasov, A. S. Chirtsov and A. E. Fomin. Biophysics. 2006, 51, No. 7, 110–113.

9. J.W. Ponder and D.A. Case. Adv. Prot. Chem. 2003, 66, 27–85.

10. Yao, Z.; Wang, J.-S.; Liu, G.-R.; Cheng, M. Comp. Phys. Comm. 2004, 161, 27–35.

11. Gonnet, P.J. Comp. Chem. 2007, 28(2), 570–573.

12. Meloni S. and Rosati M. J. Chem. Phys., 2007, 126, 121102.

13. L. Verlet, Phys. Rev. 159, 98 (1967); Phys. Rev. 165, 201 (1967).

14. Kahle J.A., Day M.N., Hofstee H.P., Johns C.R., Maeurer T.R., Shippy D. // IBM Journal of Research and Development. – 2005. – Vol. 49, No. 4/5. – P. 589–604.

15. Programmer's Guide to the IBM SDK for Multicore Acceleration. Vol. 3,0. IBM. 2009.

16. Fomin E., Alemasov N. Parallel computing technologies 2009 / Malyshkin V; Springer. LNCS, Vol. 5698. 2009. P. 399–405.

**Д.А. Чарнцев, А.Н. Ефремов**

ОАО «НПО “Искра”», г. Пермь

## ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА КЛАСТЕРНЫХ СИСТЕМАХ ПРИ ИССЛЕДОВАНИИ ГАЗОДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК ШУМОТЕПЛОЗАЩИТНОГО КОЖУХА ГАЗОТУРБИННОЙ УСТАНОВКИ

В последнее время большую актуальность приобрели численные трехмерные газодинамические исследования проточных трактов газоперекачивающих агрегатов. Одним из наиболее сложных с точки зрения течения воздуха и распределения газодинамических характеристик является проточный тракт системы охлаждения газотурбинной установки (ГТУ), а именно про-

странство под шумотеплозащитным кожухом (КШТ), где располагается ГТУ. На рис. 1 представлена схема системы охлаждения ГТУ одного из газоперекачивающих агрегатов, разработанных НПО «Искра».

ГТУ размещается под КШТ с целью снижения шума на территории компрессорной станции. Кроме ГТУ под кожухом устанавливается датчиковая аппаратура (датчики систем газообнаружения, пожаротушения и др.), элементы которой имеют различную температуру эксплуатации. В результате вентиляции из-под КШТ выносится горячий воздух, и вследствие этого поддерживается приемлемый для датчиковой аппаратуры температурный режим. Главной проблемой при этом является существенно неравномерное распределение температур в пространстве под КШТ, там образуются зоны с высокотемпературным воздухом. В связи с этим важно знать поле температуры для правильного размещения датчиковой аппаратуры под КШТ.

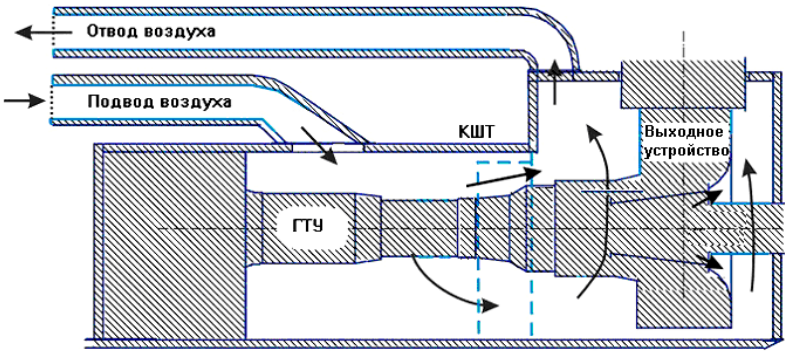



Рис. 1. Схема системы охлаждения ГТУ

Кроме того, при эксплуатации ГПА имеют место режимы работы с кратковременным (10–30 с) отключением системы охлаждения кожуха ГТУ при переключении вентиляторов на источник бесперебойного питания в случае несанкционированного отключения электроэнергии, а также при переходе с основного вентилятора на запасной при выходе вентилятора из строя. Прекращение подачи охлаждающего воздуха под кожух ГТУ при

продолжающем работать двигателе приводит к перестройке установившихся полей температуры и скорости и может повлечь существенное увеличение температуры воздуха в зонах установки аппаратуры. То есть возникает задача исследования эволюции полей газодинамических параметров с течением времени при остановке подачи воздуха под КШТ.

Для решения вышеперечисленных задач требуется трехмерное численное моделирование течения под КШТ. На кафедре ММСП ПГТУ для описания газодинамических процессов, происходящих под КШТ, была разработана математическая модель, основанная на системе уравнений Эйлера, а также программа для численного решения поставленных краевых задач газовой динамики. В качестве численного метода использовался метод крупных частиц Давыдова. Данный метод позволяет достаточно хорошо исследовать течения газа при сложной геометрии границ расчетной области в двух- и трехмерном случае. Для простоты использовалась явная конечноразностная схема. В качестве расчетной сетки использовалась однородная ортогональная сетка (рис. 2).

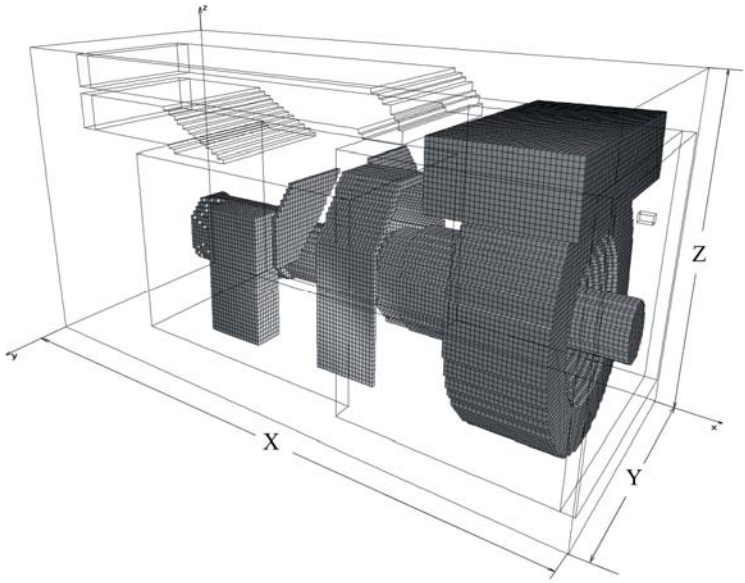



Рис. 2. Однородная ортогональная сетка

Использование такой сетки не позволяет описывать округлые поверхности так же хорошо, как в случае использования конечно-элементного триангуляционного разбиения. Однако этот недостаток можно компенсировать, с одной стороны, за счет уменьшения размеров сетки. С другой стороны, при исследовании течения воздуха под КШТ нет необходимости точно аппроксимировать округлую форму корпусов ГТУ, так как на поверхности ГТУ содержится много таких конструктивных элементов, как трубки, шланги, а также другое оборудование. Поэтому использование ортогональной сетки представляется вполне оправданным.

Нахождение полей скорости и температуры при номинальной работе системы охлаждения предусматривает стационарное решение задачи газовой динамики. Задача нахождения эволюции полей скорости и температуры при аварийном останове вентиляторов системы охлаждения ГТУ является нестационарной. Однако даже стационарное решение ищется путем установления нестационарных газодинамических процессов.

Расчетное время (время протекания газодинамических процессов), в течение которого поле температуры выходит на стационарный режим при номинальной работе системы охлаждения ГТУ, зависит от характера течения воздуха под КШТ. В среднем для конструкций систем охлаждения ГТУ, разрабатываемых ОАО НПО «Искра», время установления составляет около трех секунд. Однако некоторые конструкции требуют расчета в течение 10–12 секунд. Например, как показали расчеты, поле температуры для конструкции системы охлаждения, показанной на рис. 1, устанавливается к двенадцатой секунде. Эволюцию полей скорости и температуры при аварийном отключении вентиляторов системы охлаждения ГТУ необходимо просчитывать в течение минуты расчетного времени после остановки вентиляторов.

Вышеописанные расчеты требуют больших вычислительных ресурсов. Если учесть, что на этапе проектирования системы охлаждения ГТУ иногда требуется просчитать несколько вариантов конструкции, содержащих различные устройства для направления потоков воздуха, то поставленная задача является трудновыполнимой при решении ее на персональных компьютерах. При ограниченных вычислительных мощностях, с одной

стороны, приходится увеличивать размер ячеек расчетной сетки, что, в свою очередь, ведет к потере точности расчетов. С другой стороны, приходится сокращать количество рассматриваемых вариантов конструкций, что также приводит к потере качества проектирования системы охлаждения ГТУ.

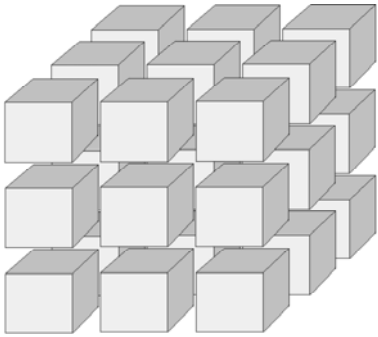


Рис. 3. Расчетная область

Ранее на персональных компьютерах с помощью разработанной программы проводились расчеты для нескольких наиболее характерных конструкций системы охлаждения ГТУ. В результате этих расчетов были выявлены их сильные и слабые стороны с точки зрения равномерного обдува поверхности ГТУ. Также были проведены довольно приблизительные расчеты для случая аварийного отключения вентиляторов системы охлаждения ГТУ. Однако более глубокие исследования того, как можно доработать ту или иную конструкцию системы охлаждения ГТУ, были трудновыполнимы. Аналогичные трудности были и при выполнении более точных исследований аварийных режимов работы системы охлаждения.

Поэтому для решения вышеописанных задач актуальным стало использование высокопроизводительных кластерных систем. Распараллеливание вычислительного процесса при решении такого рода задач может в десятки раз сократить время расчетов и повысить их точность вследствие уменьшения размеров ячеек расчетной сетки. Использование конечноразностных явных схем и однородной ортогональной сетки делает процедуру распараллеливания алгоритма метода крупных частиц довольно простой. Вся расчетная область делится на подобласти, расчет каждой из которых передается одному процессору кластера (рис. 3).

Для каждой подобласти организуется обмен данными граничных слоев расчетных ячеек между ней и подобластями, непосредственно к ней прилегающими.

В соответствии с вышеизложенной процедурой распараллеливания была доработана программа трехмерного газодинамического расчета, модернизирован алгоритм метода крупных частиц, а именно в него был введен обмен данными между узлами кластера с помощью процедур MPI. В настоящее время с помощью модернизированной программы ведутся расчеты на кластере ПГТУ. Проводятся более точные расчеты различных конструкций систем охлаждения ГТУ с направляющими воздушный поток устройствами. Это позволяет оптимизировать процесс вентиляции пространства под КШТ с целью обеспечения штатной работы датчиковой аппаратуры. Также проводятся расчеты полей температуры при аварийном выключении вентиляторов системы охлаждения для разных типов ГТУ.

**Е.А. Шамов, С.С. Барышникова, Д.Н. Жариков, Д.С. Попов**

Волгоградский государственный технический университет

## ПРОБЛЕМЫ, ВОЗНИКАЮЩИЕ ПРИ МОДЕЛИРОВАНИИ ДИНАМИКИ ПРОИЗВОЛЬНЫХ ОБЪЕКТОВ НА КЛАСТЕРЕ ЦЕНТРАЛЬНЫХ И ГРАФИЧЕСКИХ ПРОЦЕССОРНЫХ УСТРОЙСТВ

Моделирование динамики произвольных объектов является приоритетным направлением развития современной фундаментальной науки. Моделирование динамики объектов и процессов способствует генерации новых знаний, которые уже основаны на знаниях. Моделирование позволяет подтверждать или опровергать различные теории. Особенно важным фактором является то, что моделирование дает новые знания, позволяющие создавать инновационную продукцию. По данной причине наиболее развитые государства вкладывают огромные ресурсы в развитие данного направления и, что немаловажно, получают новые технологии и знания в целом, которые несут еще большую экономическую выгоду.

В мире существует огромное количество систем, позволяющих моделировать динамику различных объектов, однако

нет универсальной системы, которая позволяла бы моделировать динамику произвольных объектов. Все существующие системы загнаны в жесткие рамки, которые не позволяют расширить диапазон моделируемых объектов и соответственно процессов. Поэтому основной проблемой моделирования динамики произвольных объектов является универсальность.

Еще одним камнем преткновения является простота системы моделирования. Подавляющее число систем состоит из очень сложных «переплетений», в которых даже проектировщик, как правило, не может разобраться. Однако даже такие задачи решают узкоспециализированные задачи. Многие исследователи часто говорят, что универсализм не нужен, и можно рассчитывать отдельные модели, однако данное мнение ошибочно. В 60-х – 70-х годах двадцатого века был подобный случай. Тогда «принцип мозаики» использовался в «искусственном интеллекте», что, в свою очередь, и привело к тупиковой ситуации.

Достаточно важным аспектом при моделировании является удобство системы, так как пользователю зачастую оказывается затруднительным использование таких систем по причине сложности и неудобности интерфейса. Порой приходится часами искать, где поставить галочку, даже при наличии справки. Пожалуй, не будем вдаваться в подробности проблем, касающихся простоты и удобства в использовании систем моделирования. Посмотрим на создаваемые системы моделирования с другой стороны.

Система по моделированию динамики произвольных объектов должна за минимальный срок выдавать необходимые результаты. В современном мире создание качественных и мощных систем без использования средств параллельного программирования просто невозможно. Существует несколько аппаратных систем, в которых применяется параллельное программирование. Простейшими являются системы с общей памятью, которые делятся на многоядерные системы с общей памятью, системы с общей памятью, задействующие графические процессорные устройства (ГПУ), а также их гибриды. К сложным системам относятся системы с распределенной памятью, а именно гриды, кластеры и кластеры центральных и графических процессорных систем (кластеры ЦПУ и ГПУ). Грид – группа неоднородных компьютеров объединенных

вычислительной сетью для расчета общей задачи. Кластер – группа однородных вычислительных устройств собранных в единый вычислительный ресурс с помощью коммутационного оборудования. Стоит заметить, что научные термины *кластер* и *кластер ЦПУ* являются синонимами. Кластер ЦПУ и ГПУ-кластер, который использует для вычислений не только ЦПУ, но и ГПУ. Очень важно отметить, что кластер ЦПУ и ГПУ стоит на порядок дешевле, чем обычный кластер, обладающий такой же вычислительной мощностью. Далее будем рассматривать проблемы, возникающие при работе с кластером ЦПУ и ГПУ, так как данная аппаратная система наиболее перспективная при моделировании динамики произвольных объектов.

Основной логической проблемой является разделение исследуемой модели на части (объекты). При моделировании исследователь сам выбирает, из каких объектов необходимо строить модель. Это уже его личное дело, однако можно сказать, что, к примеру, при моделировании Солнечной системы в роли элементарных объектов могут быть использованы планеты или «элементарные» частицы, такие как кварки. Разделив модель на объекты, мы не решим проблему, так как теперь для соблюдения всех правил необходимо учесть взаимодействие между объектами. Поскольку время течет непрерывно, а компьютер – дискретная вычислительная система, при моделировании, как правило, используют численное интегрирование. В результате взаимодействие между объектами перерассчитывается на каждом шаге интегрирования и, соответственно, меняются характеристики объектов.

Перейдем к рассмотрению отдельного класса проблем, связанных с данными. Хранение данных, как ни странно, при моделировании динамики произвольных объектов является весьма актуальной проблемой. Если данные о модели настолько малы, что помещаются в оперативную память каждого ГПУ отдельно взятого вычислительного узла, то на первый взгляд проблема отсутствует, но обращение к ОЗУ ГПУ занимает длительное время, что является критичным при вычислениях. Поэтому необходимо эффективно использовать регистры ГПУ с минимизацией обмена с ОЗУ ГПУ. Однако при такой ситуации, когда все необходимые данные не могут поместиться в ОЗУ ГПУ, приходится использовать ОЗУ вы-

числительного узла, что соответственно замедляет работу. Если же нужные данные не помещаются в ОЗУ узла, то ничего не остается, как работать с жесткими дисками, что, в свою очередь, делает скорость вычислений крайне низкой. Что делать, если данные не помещаются на жесткие диски? Только увеличить объем вмещаемой на дисках информации.

Пересылка данных между узлами зачастую отнимает много времени. Эту проблему можно частично решить, если при пересылке данных маленькие пакеты данных заменить на пакеты оптимального размера. При моделировании, как правило, возможно обмениваться один раз за шаг интегрирования. По крайней мере к этому нужно стремиться. Очень важно для достижения наибольшей эффективности добавить еще одну сеть в кластер ЦПУ и ГПУ. Добавленная сеть может обладать невысокой пропускной способностью, но хорошей латентностью. Эту сеть предполагается использовать в роли управляющей. Самой лучшей для вычислительной сети является на данный момент технология InfinyBand, которая обладает пиковой теоретической скоростью 120 Gb/s. Однако коммутационное оборудование, работающее с InfinyBand, стоит очень дорого и по этой причине используется крайне редко.

Зачастую система моделирования вылетает с ошибками или, что крайне трудно обнаружить, просто неправильно работает, хотя логика программы на первый взгляд правильная. Возможно проблема в синхронизации работы оборудования. К примеру, в ГПУ часто возникает ситуация, когда часть мультипроцессоров еще не закончила необходимые расчеты, а другая часть уже начала рассчитывать следующий блок заданий. Чтобы исключить подобные случаи, необходимо использовать специальные команды синхронизации.

При использовании ГПУ в качестве вычислителей бывает вообще абсурдная ситуация. При рассчете малых моделей все работает правильно и получаемые результаты являются верными, однако при исследовании больших моделей происходит зависание программы и сбой в работе драйвера ГПУ, после чего драйвер перезагружается. На разрешение данной проблемы уходит

много времени. Оказывается, у ГПУ есть специальный таймер, который прерывает выполнение задачи при его переполнении. Таймер сбрасывается если задача выполняется быстрее, чем таймер переполнится. Есть два пути решения данной проблемы. Первый – разделить задачу на куски, которые недолго выполняются, и последовательно «скормить» ГПУ. Второй (производитель ГПУ запрещает это делать) – отключить таймер. Запрет сделан по двум причинам, а именно: при зависании ГПУ драйвер не будет перегружен и при длительных вычислениях ГПУ может перегреться, что может повлечь за собой его поломку.

Таким образом, проблем при моделировании динамики произвольных объектов на кластере ЦПУ и ГПУ достаточно. В реальности это далеко не все проблемы, с которыми можно столкнуться, но исследования, проводимые с помощью моделирования, крайне важны и позволяют исследователям получать новые знания, что, без сомнения, способствует развитию не только науки и экономики, а человечества в целом.

## Список литературы

1. Шамов Е.А., Шеин А.Г. Система моделирования динамики электронных потоков в скрещенных полях на вычислительном кластере (MDESonC): свидетельство об офиц. регистрации программы для ЭВМ Российская Федерация № 2010611509; опубл. 19.02.2010.

2. Шамов Е.А., Шеин А.Г., Шаповалов О.В. Система моделирования динамики электронных потоков в скрещенных полях на видеокарте (MDESonG): свидетельство об офиц. регистрации программы для ЭВМ Российская Федерация № 2010611508; опубл. 19.02.2010.

3. Шеин А.Г., Шамов Е.А. Стохастическая модель динамики плоского электронного потока в скрещенных статических электрическом и магнитном полях // Изв. ВолгГТУ. Серия «Электроника, измерительная техника, радиотехника и связь». – Вып. 3. 2009. – С. 48–53.

4. Калинин Ю.А., Кожевников В.Н., Лазерсон А.Г. Сложная динамика и явление динамического хаоса в потоке заряженных частиц, формируемом магнетронно-инжекторной пушкой (численный и физический эксперимент). – СПб.: Питер, 2000. – 101 с.

5. Михайловский А.Б. Теория плазменных неустойчивостей. Неустойчивости неоднородной плазмы. – М.: Атомиздат, 1977. – 360 с.

6. Воеводин В.В., Жуматий С.А. Вычислительное дело и кластерные системы. – М.: Изд-во МГУ, 2007. – 150 с.

7. Развитие параллельного программирования на примере моделирования динамики электронного потока в скрещенных полях / Е.А. Шамов [и др.] // Инновационные технологии в управлении, образовании, промышленности (АСТИНТЕХ–2010): материалы междунар. науч. конф. – М., 2010.

8. Шамов Е.А. Моделирование динамики электронного потока в скрещенных полях на кластере центральных и графических процессорных устройств. // VII Всерос. межвузовская конференция молодых ученых. 2009.

9. Шамов Е.А., Шеин А.Г. Анализ спектрального состава электронного потока в скрещенных полях с использованием кластера центральных и графических процессорных устройств // Современные информационные технологии–2010: материалы междунар. науч.-техн. конф. – М., 2010.

10. Анализ эффективности применения технологии Nvidia CUDA для задач физической электроники / Е.А. Шамов [и др.] // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычислительной техники и информатики в технических системах». – Вып. 8. – 2009.

**Е.А. Шамов, С.С. Барышникова, Д.Н. Жариков, О.В. Шаповалов**

Волгоградский государственный технический университет

# ВИЗУАЛИЗАЦИЯ МОДЕЛИРОВАНИЯ ДИНАМИКИ ПРОИЗВОЛЬНЫХ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ КЛАСТЕРА ЦЕНТРАЛЬНЫХ И ГРАФИЧЕСКИХ ПРОЦЕССОРНЫХ УСТРОЙСТВ ПРИ ПОМОЩИ СИСТЕМЫ PARAVIEW НА ПРИМЕРЕ ЭЛЕКТРОННЫХ ПОТОКОВ

Моделирование динамики произвольных объектов (МДПО) является одним из самых важных направлений развития современной науки Российской Федерации и мирового сообщества в целом. Курс на развитие инновационных проектов, взятый РФ, неотъемлемо связан с получением новых знаний и внедрением их в производство, что, в свою очередь, опирается на современную науку, основными средствами получения знаний в которой являются моделирование и эксперимент.

Создание и использование инновационных программно-аппаратных систем также крайне важно в наше время. Для того чтобы грамотно интерпретировать и представить на всеобщее обозрение полученные результаты, необходимо использовать качественные и эффективные системы визуализации. Визуализация позволяет облечь полученные результаты в форму, которую гораздо легче воспринимать. Иными словами, чтобы донести весь смысл проведенной работы, ее необходимо наглядно показать.

ParaView – мультиплатформенный программный продукт с открытым исходным кодом для визуализации и анализа данных. Может работать как на одном компьютере, так и на кластере центральных процессорных устройств (кластере ЦПУ) или на отдельном графическом процессорном устройстве (ГПУ). С его помощью можно создавать изображения данных, пригодные для презентации без дополнительной обработки. Основными достоинствами являются параллельная обработка данных и возможность обработки огромного объема данных.

Моделирование динамики электронных потоков – частный случай МДПО. Причина популярности исследования направле-

ния моделирования динамики электронных потоков определяется множеством факторов, таких как: влияние солнечной активности на озоновый слой атмосферы и здоровье человека, сушка сыпучих материалов, уничтожение насекомых-вредителей и их личинок, электромагнитное оружие, радиолокация, радиопротиводействие, радионавигация, промышленный нагрев и т.д. Главная особенность моделирования динамики произвольных объектов заключается в том, что это дает возможность производить исследование процессов, недоступных для непосредственного изучения в реальных приборах [1]. Динамика электронного потока описывается следующей формулой:

$$\partial v / \partial t = e / m\{E_0 + E_p + [v_i B_0]\}, \quad (1)$$

где $E_0$ и $B_0$ – векторы напряженности и магнитной индукции статического поля соответственно, $E_p$ – напряженность поля в месте расположения частицы, обусловленное кулоновским взаимодействием частиц, $v_i$ – вектор скорости $i$-й частицы. При моделировании учитывается взаимодействие частиц друг с другом, а также с электрическим и магнитным полями. Для расчета взаимодействия между частицами электронного потока, представляемыми в виде материальных точек, используется метод «частица–частица» [8, 11]. Это, в свою очередь, приводит к квадратичной зависимости времени расчета от количества частиц в потоке. Так же для моделирования также используется метод «крупных частиц». Метод значительно сокращает сложность расчета, но приводит к неточностям при учете сил пространственного заряда. Поэтому чем больше частиц, тем точнее расчет [7].

Моделирование динамики большого числа объектов достаточно сложная задача даже для современных вычислительных устройств. В наше время исследователи понимают, что моделировать сложные процессы, на отдельном вычислительном устройстве за приемлемое время не представляется возможным. Выходом из выше описанной проблемы может стать использование кластера (кластера ЦПУ), то есть однородной группы компьютеров, объединенных в единый вычислительный ресурс. В результате такое объединение дает возможность использовать теоретически не ограниченное число ЦПУ. Для эффективного

программирования кластера очень удобно использовать объединение технологий MPI и OpenMP [4, 5, 9].

Для выявления тенденций и закономерностей при моделировании электронного потока в скрещенных полях на вычислительном кластере был проведен ряд экспериментов. Результаты сравнения кластера, содержащего 68 ядер (все процессоры – Intel Xeon 3 ГГц), и одного ядра процессора Intel Xeon 3 ГГц, представлены в табл. 1 [3].

Таблица 1

**Результаты вычислительных экспериментов**

| Число электронов в потоке | Время расчета на одном ядре Intel Xeon 3 ГГц, с | Время расчета, MPI+OpenMP, с | Ускорение |
|---|---|---|---|
| 20000 | 234 | 11 | 21,3 |
| 50000 | 1232 | 33 | 37,3 |
| 100000 | 4781 | 117 | 40,9 |
| 200000 | 18670 | 385 | 48,5 |
| 500000 | 117124 | 2099 | 55,8 |
| 1000000 | 468496 | 7718 | 60,7 |

В результате исследований выяснилось, что использовать кластер очень эффективно, но дорого. Необходимо было найти альтернативу кластеру за меньшие деньги [10]. Оказалось, теоретически возможно собрать кластер центральных и графических процессорных устройств (кластер ЦПУ и ГПУ), то есть кластер, узлы которого содержат мощные и недорогие ГПУ [2]. В рамках исследования на базе Волгоградского государственного технического университета (ВолгГТУ), кафедры ЭВМ и Систем был собран кластер ЦПУ и ГПУ (рис. 1). Он содержит 8 ГПУ NVIDIA GTX 260 и 2 ГПУ NVIDIA TESLA 1060, что соответствует реальной мощности в 5–6 Tflops плюс мощность процессоров, то есть реальная суммарная мощность около 6 Tflops. Стоимость же такого кластера около 400–450 тыс. рублей без учета стоимости программного обеспечения [6].

С программной точки зрения потребовались приложить гораздо большие усилия, однако с точки зрения программирования данная система стала даже удобнее.

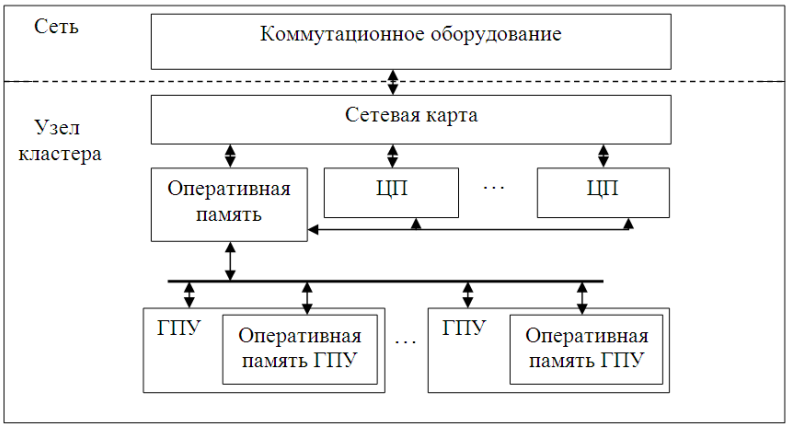



Рис. 1. Аппаратная структура кластера ЦПУ и ГПУ

Результаты экспериментов, показывающие производительность системы кластера ЦПУ и ГПУ, представлены в табл. 2.

Таблица 2

**Производительность системы кластера ЦПУ и ГПУ**

| Число электронов в потоке | Аппроксимированное время расчета на одном ядре Intel Xeon 3 ГГц, (с) | Время расчета на кластере ЦПУ и ГПУ, (с) | Ускорение (разы) |
|---|---|---|---|
| 50000 | 1232 | 48,03 | 25,7 |
| 100000 | 4781 | 80,01 | 59,8 |
| 200000 | 18670 | 96,7 | 193 |
| 500000 | 117124 | 314 | 373 |
| 1000000 | 468496 | 1145 | 409 |
| 2000000 | 1873984 | 4494 | 417 |
| 4000000 | 7495936 | 17847 | 420 |

Из табл. 2 можно сделать заключение о том, что кластер ЦПУ и ГПУ очень эффективен для решения задачи моделирования динамики электронного потока, так как позволяет рассчитывать большие электронные потоки в 400 быстрее, чем при использовании последовательной программы.

Сравнивая кластер ЦПУ и кластер ГПУ как системы с отсутствием явного предела по вычислительной мощности, можно сделать несколько важных выводов:

1) кластеры ЦПУ и ГПУ более чем на порядок дешевле обыкновенного кластера с эквивалентной мощностью;

2) на кластерах ЦПУ и ГПУ можно решать ограниченный круг задач;

3) кластеры ЦПУ и ГПУ потребляет меньше энергии, чем кластер с эквивалентной вычислительной мощностью;

4) кластеры ЦПУ и ГПУ занимает меньшую площадь;

5) программирование кластеров ЦПУ и ГПУ требует больших познаний и практического навыка программирования в области архитектуры компьютера.

Использование ParaView при МДПО позволяет за минимальные сроки произвести визуализацию проведенных исследований, что очень важно, так как наибольший объем информации усваивается человеком визуально. ParaView обладает достаточно удобным и простым инструментарием. Вообще система ParaView является надстройкой над VTK. VTK – это система визуализации данных и обработки изображений с открытым исходным кодом, которая широко используется в научном сообществе. ParaView, как и любая система, обладает некоторыми недостатками. К примеру, верстать файл AVI формата с использованием больше чем $10^7$ точек крайне затруднительно, а $10^8$ вообще система зависает (возможно, не хватает оперативной памяти и система использует swap). Также при верстке AVI-файла могут появляться какие-то лишние объекты, или объекты находятся не там, где должны. Пример 3d электронного потока спроецированного на двухмерную плоскость отображаемого в системе ParaView показан на рис. 2.

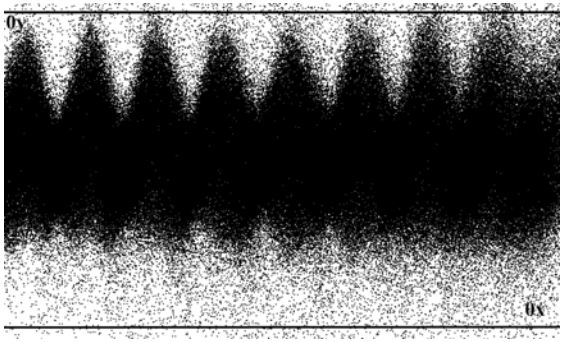



Рис. 2. Структура электронного потока при визуализации в системе ParaView

По результатам проведенных исследований можно сделать несколько ключевых выводов. Во-первых, МДПО крайне важно для развития науки и, соответственно, инновационного производства. Во-вторых, качественное МДПО невозможно без параллельного программирования и использования кластера ЦПУ и ГПУ. В-третьих, очень важна наглядность результатов исследований, а соответственно необходимо грамотное использование таких систем, как ParaView.

## Список литературы

1. Шамов Е.А. Моделирование динамики электронного потока в скрещенных полях на кластере центральных и графических процессорных устройств // Петербург–2010: материалы VII Всерос. межвуз. конф. молодых ученых. – СПб., 2010.

2. Шамов Е.А., Шеин А.Г. Анализ спектрального состава электронного потока в скрещенных полях с использованием кластера центральных и графических процессорных устройств // Современные информационные технологии–2010 (Computer-Based Conference «Contemporary information technologies-2010»): материалы междунар. науч.-техн. конф. – Пенза, 2010.

3. Развитие параллельного программирования на примере моделирования динамики электронного потока в скрещенных полях / Е.А. Шамов [и др.] // Инновационные технологии в

управлении, образовании, промышленности АСТИНТЕХ-2010: материалы междунар. науч. конф. – Астрахань, 2010.

4. Шамов Е.А., Шеин А.Г. Система моделирования динамики электронных потоков в скрещенных полях на вычислительном кластере (MDESonC): свидетельство об офиц. регистрации программы для ЭВМ Российская Федерация № 2010611509 ; опубл. 19.02.2010.

5. Шамов Е.А., Шеин А.Г., Шаповалов О.В. Система моделирования динамики электронных потоков в скрещенных полях на видеокарте (MDESonG): свидетельство об офиц. регистрации программы для ЭВМ Российская Федерация № 2010611508; опубл. 19.02.2010.

6. Шамов Е.А., Шаповалов О.В., Жариков Д.Н., Лукьянов В.С. Моделирование динамики электронного потока в скрещенных полях на кластере центральных и графических процессорных устройств // Фундаментальные исследования: материалы междунар. науч. конф. – Израиль, 2010.

7. Шеин А.Г., Шамов Е.А. Стохастическая модель динамики плоского электронного потока в скрещенных статических электрическом и магнитном полях // Изв. ВолгГТУ. Серия «Электроника, измерительная техника, радиотехника и связь». – Вып. 3. – 2009. – С. 48–53.

8. Калинин Ю.А., Кожевников В.Н., Лазерсон А.Г. Сложная динамика и явление динамического хаоса в потоке заряженных частиц, формируемом магнетронно-инжекторной пушкой (численный и физический эксперимент). – СПб.: Питер, 2000. – 101 с.

9. Воеводин В.В., Жуматий С.А. Вычислительное дело и кластерные системы. – М.: Изд-во МГУ, 2007. – 150 с.

10. Анализ эффективности применения технологии Nvidia CUDA для задач физической электроники / Е.А. Шамов [и др.] // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычислительной техники и информатики в технических системах». Вып. 8. 2009.

11. Михайловский А.Б. Теория плазменных неустойчивостей. Неустойчивости неоднородной плазмы. – М.: Атомиздат, 1977. – 360 с.

**О.В. Шаповалов, Е.А. Шамов, Д.Н. Жариков, А.Е. Андреев**

Волгоградский государственный технический университет

# РЕШЕНИЕ ЗАДАЧИ ТЕПЛОПРОВОДНОСТИ В МНОГОТЕЛЬНОЙ МОДЕЛИ АВТОМОБИЛЯ

В современных транспортных средствах начинают широко применяться регулируемые элементы в системах подрессоривания, в частности, регулируемые гидравлические демпферы, которые позволяют значительно расширить диапазон условий, в которых используется транспортное средство. Принцип действия регулируемых демпферов заключается в прокачке вязкой жидкости поршнем через дроссельные каналы с переменным гидравлическим сопротивлением, в результате чего происходит диссипация механической энергии с последующим её рассеиванием в окружающую среду. В регулируемых демпферах коэффициенты гидравлического сопротивления могут достигать значительных величин и приводить к интенсивному нагреву амортизатора. В связи с этим задача расчета теплового состояния гидравлического амортизатора подвески становится актуальной, поскольку позволит обеспечить допустимый тепловой режим амортизатора за счет оптимизации конструкции механической части амортизатора и коррекции алгоритма управления демпфированием. Оптимизация конструкции механической части может быть достигнута за счет добавления специальных элементов к корпусу амортизатора, например радиаторов, или модификации самой формы корпуса.

Расчет теплового состояния амортизатора может быть осуществлен на базе многотельной модели, которая описывает динамику движения автомобиля по неровностям дорожного микропрофиля с параллельным расчетом процесса тепловыделения и теплопередачи в демпфере подвески. В данной работе рассматриваются особенности одновременного решения уравнений динамики многотельной системы и уравнений теплопроводности в нескольких телах, входящих в многотельную модель. Многотельная модель представлена на рис. 1.

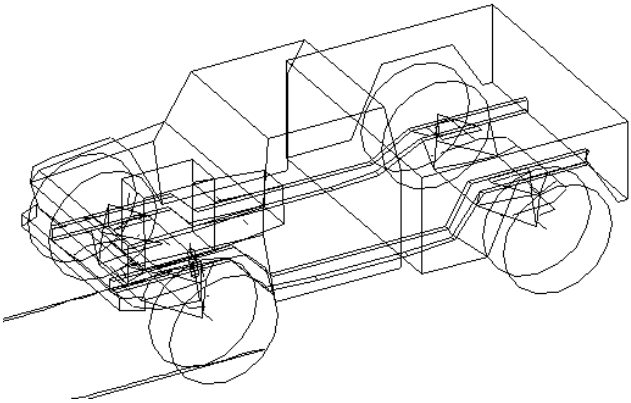


Рис. 1. Расчетная схема автомобиля в системе ФРУНД

Фрагмент модели с амортизатором представлен на рис. 2.

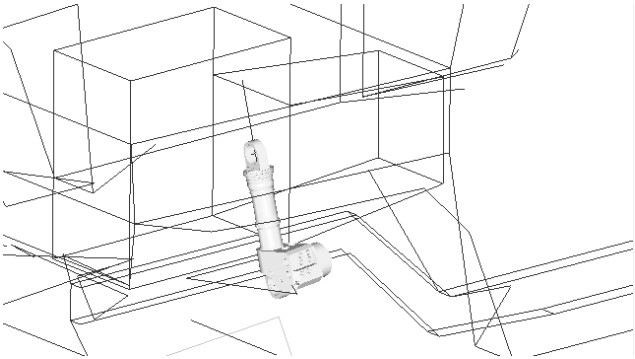


Рис. 2. Фрагмент модели с амортизатором

Для расчета динамики используется система инженерного анализа ФРУНД. В ней используется представление уравнений динамики систем тел в форме уравнений с множителями Лагранжа [4]. Уравнения связей записываются во вторых производных. Уравнения движения произвольной системы тел при таком подходе записываются в виде

$$\begin{cases} M\ddot{x} - D^T\lambda = f(\dot{x},x,t), \\ D\ddot{x} = h(\dot{x},x). \end{cases} \quad (1)$$

Здесь $x$ – вектор переменных системы размерностью $n$, $M$ – постоянная матрица коэффициентов при вторых производных, $f(\dot{x},x,t)$ – вектор – функция правых частей дифференциальных уравнений, $D$ – матрица переменных коэффициентов уравнений связи размерностью $k \times n$ ($k$ – число связей), $h(\dot{x},x)$ – вектор правых частей уравнений связи, $\lambda$ – вектор множителей Лагранжа. Силы в амортизаторах входят в правые части первого уравнения системы (1).

В ходе работы подвески в амортизаторе выделяется энергия, которая переходит в тепло и рассеивается через корпус амортизатора в окружающее пространство. На каждом шаге расчета модели определяется скорость амортизатора, на основе которой по формуле (2) рассчитывается работа, совершенная демпфером.

$$\Delta Q = C\int_{t_1}^{t_2} F_A(t)\cdot V(t)dt \quad (2)$$

где $F_A(t)$ – сила гидравлического сопротивления амортизатора, $V(t)$ – скорость движения поршня.

Затем, считая, что вся энергия идет на равномерный нагрев жидкости амортизатора, по формуле (3) определяется температура границы на каждом шаге расчета, которая будет использоваться при решении задачи нестационарной теплопроводности в качестве граничных условий с внутренней стороны амортизатора.

$$T_{i+1} = T_i + \Delta Q \quad (3)$$

С внешней стороны граничным условием будет служить температура воздуха.

Технология расчета процесса теплопередачи состоит в использовании дискретной модели, аппроксимирующей решение уравнения теплопроводности для произвольной геометрии. Особенностью метода является использование кубической сетки для разбиения геометрии. Технология расчета процесса теплопередачи включает разбиение исходной геометрии на конечные

элементы, расчет полученной конечно-элементной модели и визуализацию поведения модели на основе данных, полученных при проведении расчетов. Сетка представлена на рис. 3.

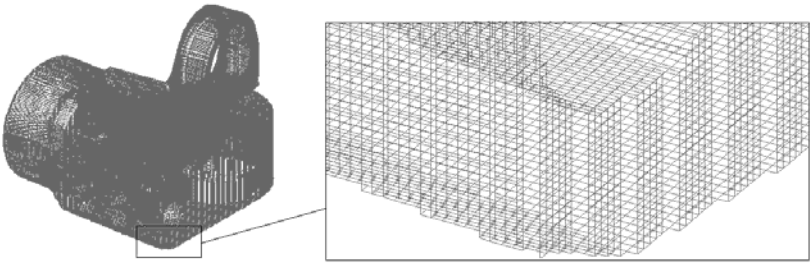


Рис. 3. Сетка разбиения детали амортизатора

На этапе расчета происходит интегрирование системы обыкновенных дифференциальных уравнений теплопроводности, соответствующих рассматриваемому изделию. Используется дискретное представление уравнений теплоемкости в форме, ориентированной на решение методом установления.

Дискретная математическая модель соответствует уравнению теплопроводности в частных производных:

$$\gamma\rho\frac{\partial u}{\partial t} - q\nabla^2 u = 0, \tag{4}$$

где $u$ – температура, $q$ – коэффициент теплопроводности материала, $\gamma$ – теплоемкость, $\rho$ – плотность, с начальными условиями

$$u(x, y, z, 0) = f(x, y, z) \tag{5}$$

и граничными условиями на внешней границе,

$$u(x, y, z)\big|_{T_b} = T_b \tag{6}$$

и граничными условиями на внутренней границе

$$u(x, y, z)\big|_{T_i} = T_{i+1} \tag{7}$$

Уравнение (4) можно с заданной точностью представить в виде

$$\alpha\frac{\partial^2 u}{\partial t^2} + \gamma\rho\frac{\partial u}{\partial t} - q\nabla^2 u = Q(x, y, z, t). \tag{8}$$

Дискретизация уравнения (8) по методу конечных элементов приводит к уравнению вида

$$M\ddot{X} + C\dot{X} + KX = G(x,t), \qquad (9)$$

где $X$ – вектор узловых переменных размерности $n$, $M_{n\times n}$ – матрица при вторых производных, $C_{n\times n}$ – матрица демпфирования, $K_{n\times n}$ – матрица жесткости, $G(x,t)$ – вектор узловых нагрузок, соответствующий источникам тепла.

Матрица жесткости в предлагаемом методе складывается из элементов, которые для связи двух соседних узлов записываются в виде

$$k = q / l / 9, \qquad (10)$$

где $l$ – расстояние между соседними узлами.

Матрица демпфирования складывается из элементов вида

$$C_{ij} = \gamma\rho / l / 9. \qquad (11)$$

В рассматриваемой реализации метода предполагается, что решетка разбиения – кубическая, с кубиками одинакового размера с длиной ребра $l$. Использование равномерной кубической решетки позволяет интегрировать уравнение (9) без явного вычисления матрицы жесткости. Правая часть уравнения (9) для i-го узла будет вычисляться по формуле

$$\alpha\ddot{x}_i = -\sum_{j=1}^{m} k(x_i - x_j) - \sum_{j=1}^{m} c_{ij}(\ddot{x}_i - \dot{x}_j) + g(x,t) \qquad (12)$$

где $m$ – число контактирующих с узлом элементов.

Выбранные методы разбиения модели и проведения расчетов допускают эффективное распараллеливание на многопроцессорных вычислительных системах. В качестве возможных путей распараллеливания были рассмотрены различные технологии. При адаптации алгоритма расчета для распределенных систем с использованием стандарта MPI было выяснено, что коммуникационные затраты превышают выигрыш от параллельного решения. При анализе алгоритма была доказана принципиальная невозможность эффективного распараллеливания для подобного класса вычислительных систем с использование MPI. Доказательство основывалось на сетевом законе Амдала.

При адаптации программы для систем с общей памятью выяснилось, что затраты на порождение параллельных потоков слиш-

ком велики для данной задачи и параллельная программа не выигрывает в скорости у последовательной версии. Идеальным решением для данного алгоритма стало применение технологий GPGPU, поскольку коммуникационные затраты в данном случае незначительны и сами потоки более легковесны, так как аппаратная структура графических процессоров рассчитана на мелкозернистый параллелизм. Для эффективной реализации алгоритма на данной платформе необходимо использование большого числа потоков, что не является проблемой для выбранного алгоритма решения.

Реализация расчета теплопроводности в параллельном режиме выполнена на видеокартах Nvidia GeForce GTX260. Для рассматриваемых моделей распараллеливание производилось на сотни тысяч потоков. При этом удалось достичь ускорения расчета в 2–3 раза по сравнению с оптимизированной однопроцессорной реализацией.

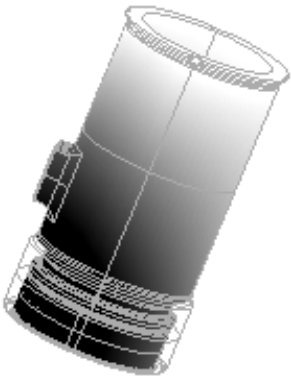


Рис. 4. Распределение температур в амортизаторе

Описанная выше технология расчетов позволяет использовать CAD-геометрию любой сложности, имеет достаточную точность расчета, предоставляет высокую скорость разбиения геометрии на сетку, генерации и интегрирования дифференциальных уравнений. В работе представлены результаты оптимизации конструкции корпуса упруго-демпфирующего элемента управляемой автомобильной подвески для обеспечения заданного температурного режима. Исследовалось движение автомобиля по случайному профилю различных типов. Модель автомобиля состояла из 50 тел. Дискретная модель теплопроводности корпуса упруго-демпфирующего элемента подвески строилась по точной CAD геометрии и насчитывала более 200 тыс. узлов. На рис. 4 представлено распределение температур в амортизаторе.

Совместное решение задачи динамики и теплопроводности реализовано в программном комплексе ФРУНД [2].

### Список литературы

1. Internet resource. Open CASCADE Technology, 3D modeling & numerical simulation. – URL: http://www.opencascade.org, 2010.

2. Internet resource. Software for simulation the multibody dynamics of the rigid and flexible bodies. – URL: http://www.frund.vstu.ru, 2010.

3. Maki K. Habib, Bioinspiration and Robotics Walking and Climbing Robots, 2007.

4. Principles of CAD/CAM/CAE Systems, Kunwoo Lee, 2004.

5. Vittendurg I.S. The dynamics of solid body system, 1980.

**[1]А.Н. Шарифулин, [2]С.А. Суслов**

[1]Пермский государственный технический университет
[2]Swinburne University of technology, Hawthorn, Australia

## КОНВЕКТИВНЫЕ БИФУРКАЦИИ НЕСЖИМАЕМОЙ ЖИДКОСТИ В НАКЛОНЯЕМОЙ ПОЛОСТИ КВАДРАТНОГО СЕЧЕНИЯ

Рассматривается стационарная двумерная конвекция воздуха в цилиндрической горизонтальной полости квадратного сечения, две противоположные стенки которой теплоизолированы, а в двух других поддерживаются постоянные различные температуры. Полость может медленно наклоняться на произвольный угол так, что цилиндр все время остается горизонтальным, а направление подогрева жидкости из-за этого может плавно изменяться от направления подогрева снизу к направлению подогрева сбоку и так до подогрева сверху и обратно. Основное внимание уделено определению бифуркационной кривой на плоскости параметров: число Релея-угол наклона полости, при пересечении которой стационарный режим конвекции испытывает бифуркации.

Интерес к стационарным режимам конвекции воздуха в замкнутой наклоняемой полости связан с надеждой, что через по-

нимание закономерностей их бифуркаций в лабораторных моделях будет понят механизм зарождения сложных атмосферных явлений. Наклон полости используется для лабораторного моделирования, например, внезапного зарождения тропических циклонов [1] и парадоксального переноса загрязнений воздуха из города в горные районы с дорогой недвижимостью [2]. Стационарные режимы тепловой конвекции воздуха в замкнутой полости часто имеют форму валов, т.е. вихрей с горизонтальной осью. В них частицы жидкости движутся вдоль плоских линий тока вокруг точки пересечения этой плоскости с осью вихря. Поэтому такие режимы можно назвать квазидвумерными. Это обстоятельство позволяет авторам данной работы надеяться, что исследование плоских двумерных течений воздуха, т.е. бесконечно вытянутых горизонтальных вихрей, в абстрактных бесконечных цилиндрах поможет в понимании закономерностей бифуркаций стационарных режимов конвекции в полостях лабораторных экспериментов. Бифуркации конвективных течений воздуха, вызванные наклоном экспериментально начали изучаться для кубической полости, подогреваемой снизу в [3], где были рассмотрены лишь малые девиации угла наклона, соответствующего подогреву строго снизу. Поясним, что наклон при малых значениях числа Рэлея приводит к формированию горизонтального вихря с таким направлением циркуляции, что воздух, прилегающий к наклоненной нагретой нижней стенке полости, движется вверх. Это вихрь с нормальным вращением, если полость привести в горизонтальное положение, он прекратит свое вращение. Однако при значениях числа Рэлея, превышающих критическое, наряду с таким нормальным вихрем возможен и вихрь с обратным

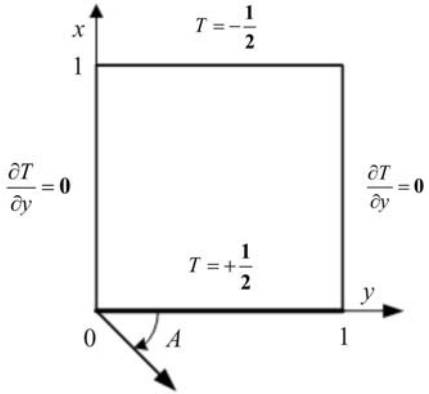



Рис. 1. Условия подогрева на твердых стенках полости. Направление вниз показано стрелкой и составляет с осью $Y$ угол $A$

направлением циркуляции, т.е. аномальный вихрь. В таком вихре воздух вдоль нагретой нижней грани куба движется вниз.

Целью настоящей работы было теоретически, путем численного решения полных уравнений тепловой конвекции в приближении Буссинеска, определить величину критического угла наклона, до достижения, которого возможно существование такого аномального вихря.

Рассматривается свободная тепловая конвекция несжимаемой жидкости в бесконечной замкнутой полости квадратного сечения с твердыми границами. Две противоположные стенки полости теплоизолированы, а две других поддерживаются при постоянных различных температурах, как показано на рис. 1. Введем угол наклона полости $A$ так, что значение $A = -90^0$ соответствует подогреву снизу, а $A = 90^0 -$ сверху. При подогреве снизу возможно условие механического равновесия и конвективное движение возникает в результате fork type bifurcation при превышении числом Рэлея критического значения $R_c = 2586$ [4]. При превышении числом Рэлея критического значения $R_c$ устанавливается стационарный одноячеистый режим конвекции.

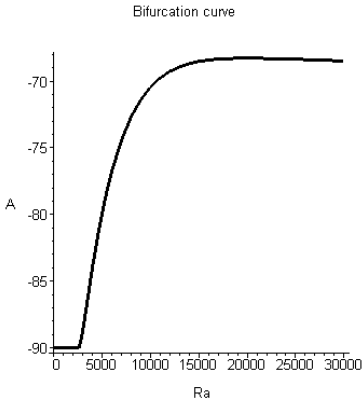



Рис. 2. Бифуркационная кривая. Область под ней характеризуется наличием трех вихревых решений – двух устойчивых с аномальным и нормальным направлением циркуляции (см. определения в тексте) и неустойчивого по седловому типу с нормальным направлением циркуляции

Причем возможны два надкритических режима, отличающихся друг от друга лишь направлением циркуляции. Для произвольных значений $A$ до настоящей работы задача не исследовалась. Имеется лишь работа [5], где в аналогичной постановке изучен случай идеально теплопроводных границ. Основное внимание в ней уделено построению бифуркационных диаграмм

и бифуркациям между одно- и двухвихревыми течениями. Получены устойчивые режимы стационарного аномального вихря для значений девиации от угла наклона, соответствующего подогреву снизу в 1 и $5^{o}$. Бифуркационная диаграмма, определяющая границы существования аномального вихря, была построена для значения числа Прандтля $\mathrm{Pr} = 1$ конечно-разностным методом для цилиндра кругового сечения [6]. Бифуркационная кривая имела экстремум, максимальное значение критического угла составляло $12^{o}$ при $\mathrm{Ra} = 2\mathrm{Ra}_{c}$.

В настоящей работе использовался метод Петрова-Галеркина [7], который позволил получить как устойчивые, так и неустойчивые стационарные решения задачи. В качестве базисных функций использовались полиномы Чебышева.

Расчеты проводились на одном процессоре компьютера с производительностью около 1000 GFlops. Время вычислений для стационарных режимов было порядка нескольких минут, интегрирование по времени занимало несколько часов.

Бифуркационная кривая, определяющая область существования аномального вихря, представлена на рис. 2 Видно, что критический угол достигает значения в $22^{o}$. Отметим, что в области под бифуркационной кривой кроме аномального вихря и основного нормального вихря имеется еще один вихрь с нормальным направлением циркуляции, но неустойчивый. Его неустойчивость имеет седловой характер, т.е. существует направление в фазовом пространстве, для которого он является притягивающим. Это означает, что он может наблюдаться некоторое (не обязательно короткое) время в численном или лабораторном эксперименте.

В заключение отметим, что бифуркационная кривая, как и кривая, полученная в [6], имеет экстремум. Однако этот экстремум значительно менее выражен, что говорит о том, что область существования устойчивого аномального вихря для цилиндра с квадратным сечением значительно шире, чем для кругового цилиндра. Это позволяет предположить, что на степень уменьшения критического угла с ростом числа Рэлея отрицательное влияние оказывают факторы наличия углов отношение площади границ к объему жидкости.

## Список литературы

1. Sharifulin A., Poludnitsin A., Kravchuk A. Bulletin of the American Physical Society, 2007. 52, 17, p. 168–169.

2. Princevac M., Fernando H. J.S. Physics of fluids, 2007. 19, 105102.

3. Zimin V., Ketov A. Izvest. AN SSSR.Mekh.zhigkocti I gaza, 1974. 5. P. 110–114.

4. Gershuni G.Z., Zhukhovitskii Convective Stability of Incompressible Fluids. – Moscow: Nauka, 1972.

5. Mizushima J., Hara Y. Journal of the Physical Society of Japan, 2000. 69, 2371–2374.

6. Nikitin A., Sharifulin A. HEAT TRANSFER-Soviet Research, 1989. 21, 213–221.

7. Suslov S.A., Paolucci S. A.:Proc. Of the ASME Heat Transfer Division, 2001. 4, 39–46.



**Л.П. Шингель**

Пермский государственный технический университет


## О РАСЧЕТЕ ТОРСИОНА НЕСУЩЕГО ВИНТА ЭКСПЕРИМЕНТАЛЬНОГО ВЕРТОЛЕТА ИЗ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ С ИСПОЛЬЗОВАНИЕМ КЛАСТЕРА

При разработке несущей системы вертолетов и некоторых самолетов вертикального взлета и посадки, особенно имеющих небольшой взлетный вес, прогрессивным является использование торсиона для установки лопастей несущего винта. Еще в 1977 г. в литературе отмечалась эффективность несущих винтов со втулкой полужесткого типа, использующих упругую торсионную подвеску. Одновременно отмечался большой объем расчетно-конструкторских работ и исследований, необходимых для получения удовлетворительных характеристик такой подвески.

В работе рассмотрен торсион втулки двухлопастного несущего винта легкого экспериментального самолета вертикального взлета и посадки, аналогичный торсиону вертолета Rotorlfy. Торсион выполнен из набора чередующихся слоев стеклопласти-

ка и резины. Особенностью является малая толщина слоев стеклопластика (один слой на основе ткани Т-25 толщиной 0,35 мм), толщина слоя резины составляет 0,6 мм. Другой особенностью данной конструкции торсиона является его пространственная, форма, не имеющая плоскостей симметрии, симметрия имеется только относительно оси, перпендикулярной горизонтальной плоскости. Длина торсиона составляет 700 мм при толщине 20 мм. Форма торсиона показана на рис. 1, в плоскости, перпендикулярной оси *Y*, торсион имеет конструктивный изгиб.

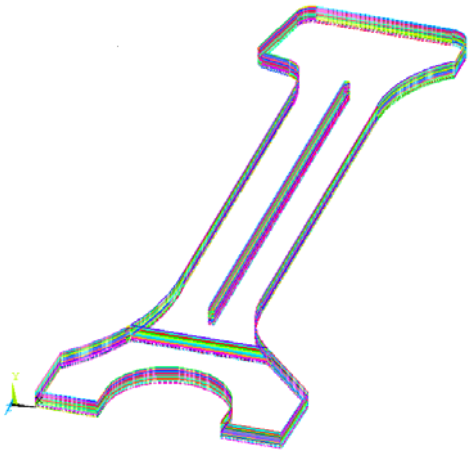


Рис. Слоистая форма торсиона

Поскольку для расчета напряженно-деформированного состояния и частотных характеристик применялся ANSYS, особенности формы не позволяли рассматривать четвертую часть торсиона, как это обычно принято. Сложность представляет так же большое отношение длинны к толщине торсиона при малой толщине слоев, из которых он набран. Эти особенности обуславливают необходимость при построении конечно-элементной модели разбивать рассматриваемую конструкцию на большое число элементов (десятки миллионов). Это приводит при решении задачи для тел, содержащих анизотропные элементы к решению большого числа уравнений. Такая задача реализуется только при использовании кластера и программного продукта ANSYS 11,0.

Исходная геометрическая модель создана с использованием программного пакета SolidWork, сохранялась в формате Parasolid и затем часть элементов импортировалась в ANSIS. Разбиение на конечно-элементную сеть и дальнейшие операции осуществлялись в ANSYS. Решалась задача нахождения НДС при приложении управляющих усилий, находилась зависимость угла поворота торсиона (лопасти) от приложенного усилия. При решении механические свойства стеклопластика задавались как для ортотропного материала. Для резинового слоя рассматриваемые механические свойства соответствовали изотропному материалу.

Полученные результаты позволяют оценить возможность и эффективность применения такого типа крепления лопасти, а также определить конструктивные параметры торсиона.

**А.В. Шулепов, И.Ю. Зубко**

Пермский государственный технический университет

## ИСПОЛЬЗОВАНИЕ КЛАСТЕРНЫХ СИСТЕМ ДЛЯ РАСЧЕТА КОНСТАНТ В НЕСИММЕТРИЧНОМ ЗАКОНЕ УПРУГОСТИ

Стремительное наращивание мощности современных вычислительных устройств позволяет решать прикладные задачи на качественно новом уровне. Еще десятилетие назад прямые методы моделирования использовались крайне редко и в основном для получения качественных оценок ввиду большой ресурсоемкости данных методов. На сегодняшний день прямое моделирование используется для получения как качественных, так и количественных оценок.

В представляемой работе используются методы прямого моделирования внутренней структуры материала для получения независимых компонент тензора упругих свойств **G**. Определение компонент тензора напряжений осуществляется с использованием метода молекулярной статики [1]. Как и многие прямые методы моделирования, данный метод является ресурсоемким в силу не-

обходимости расчета парного взаимодействия между десятками тысяч частиц. В работе рассматривается представительный на атомарном уровне объем меди, имеющей ГЦК-решетку. В качестве парного потенциала взаимодействия используется потенциал Морзе, параметры которого были взяты из [2]:

$$U(r) = 4\varepsilon\left(e^{-2\alpha(r-\sigma)} - 2e^{-\alpha(r-\sigma)}\right).$$

Метод молекулярной статики позволяет рассчитывать силы парного взаимодействия между частицами системы (атомами кристаллической решетки). В данном случае необходимо определить компоненты меры напряженного состояния, которая используется в несимметричном законе упругости. Для этого рассматривается некоторая атомарная плоскость (конечная площадка) и компонента тензора напряжений определяется как сумма всех сил, действующих на атомы из выделенной плоскости, отнесенная к ее площади. Небольшой обзор по существующим методам определения напряжений на атомарном уровне можно найти в [3]. В силу быстрого затухания силы взаимодействия на расстоянии уже около $3\sigma$ сила составляет около 1 % от максимальной. Все частицы, находящиеся на большем расстоянии, можно просто не учитывать (в крайнем случае вводить некоторую поправку для потенциала). Таким образом, данный метод позволяет получить практически *N*-кратное ускорение ввиду того, что задачу можно разделить на независимые подзадачи.

Второй частью решаемой задачи является определение параметров потенциала, так как при введенных выше параметрах значения компонент $G_{1111}$ и $G_{1122}$ не совпадают с экспериментальными значениями. Для этого решается двумерная задача оптимизации, где варьируемыми переменными являются параметры $\alpha$ и $\varepsilon$, а критерием является соответствие получаемых компонент $G_{1111}$ и $G_{1122}$ экспериментальным значениям.

Основной вычислительный блок (решатель) был разработан с использованием языка Fortran. Для эффективного распараллеливания и возможности работы на кластерных системах также была использована библиотека MPI (возможно, в дальнейшем решатель будет использован в гибридной системе, для

чего потребуется также использование других библиотек, например OpenMP). Алгоритм решения задачи можно условно разделить на несколько частей:

первоначальное определение координат частиц;

разделение расчетной области между процессорами (сюда же входит обмен данными для последующего расчета);

параллельное вычисление компонент вектора результирующей силы;

сборка данных и вычисление компонент тензора упругих свойств.

При таком алгоритме одним из узких мест является первый шаг, состоящий в определении координат частиц. Тем не менее, при тестировании решателя удалось подтвердить возможность эффективного распараллеливания, а также определить максимальное количество процессов, необходимое для эффективного решения задачи в зависимости от количества частиц. В дальнейшем планируется осуществить оптимизацию алгоритма, а также добавить возможность эффективного использования решателя в гибридных системах.

## Список литературы

1. Кривцов А.М. Деформирование и разрушение твердых тел с микроструктурой. – М.: Наука, 2007. – 301 с.

2. Girifalco L. A., Weizer V. G. Phys. Rev. 1959.

3. Trovalusci P., Capecchi D., Ruta G. Genesis of the multiscale approach for materials with microstructure, Springer-Verlag, 2008.

## СОДЕРЖАНИЕ

Научное издание

# ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ (HPC–2010)

Материалы X Международной конференции

В двух томах

Том 2

Редактор и корректор *И.А. Мангасарова*

---

Подписано в печать 26.10.2010. Формат 60×90/16.
Усл. печ. л. 20,75.
Тираж 100. Заказ № 222/2010.

---

Издательство
Пермского государственного технического университета.
Адрес: 614990, г. Пермь, Комсомольский пр., 29, к. 113.
Тел. (342) 219-80-33.