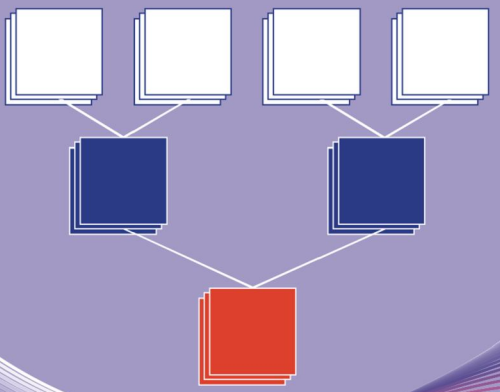


Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Томский государственный университет

ПЯТАЯ СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

1 - 3 декабря 2009 года



МАТЕРИАЛЫ КОНФЕРЕНЦИИ

Томск 2010

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПЯТАЯ СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 1–3 декабря 2009 года

ИЗДАТЕЛЬСТВО ТОМСКОГО УНИВЕРСИТЕТА
2010

УДК 519.6
ББК 22.19
П 99

Пятая Сибирская конференция по параллельным и высокопроизводительным вычислениям / Под ред. проф. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2010. – 188 с.
ISBN 978 –5 –7511 – 1942 – 3

В сборнике содержатся материалы, представленные во время работы Пятой Сибирской конференции по параллельным и высокопроизводительным вычислениям, проходившей 1–3 декабря 2009 года в Томском государственном университете при поддержке Министерства образования и науки РФ и Российского фонда фундаментальных исследований.

Рассмотрены актуальные проблемы организации параллельных вычислений на многопроцессорных системах, современное состояние и перспективы развития методов параллельных вычислений.

Для студентов, аспирантов, преподавателей, научных работников, желающих изучить и практически использовать в научной работе высокопроизводительные вычислительные ресурсы.

УДК 519.6
ББК 22.19

ISBN 978 – 5 – 7511 – 1942 – 3
© Томский государственный университет, 2010

СОДЕРЖАНИЕ

Малышкин В.Э. Особенности исполнения фрагментированных программ 6
Афанасьев К.Е., Окулов Н.Н., Стуколов С.В. Разработка информационного портала параллельных вычислений для проведения научных и инженерных расчетов в режиме on-line 10
Биматов Д.В., Сущенко Д.С., Сущенко С.П. О быстродействии подсистемы памяти узлов вычислительного кластера 15
Городняя Л.В. О проблеме начального обучения параллельному программированию 20
Старченко А.В., Панасенко Е.А. Параллельная реализация численного метода решения обратных задач переноса примеси 25
Данилкин Е.А., Старченко А.В. Распараллеливание итерационных методов численного решения эллиптического уравнения со смешанными граничными условиями и анализ эффективности полученных параллельных реализаций 36
Дьяченко Е.Н. Моделирование очистки воды от механических примесей методом дискретного элемента 43
Бойченко И.В., Корытников С.В., Лежанкин А.В. Сервис-ориентированная архитектура доступа к ресурсам высокопроизводительных вычислительных систем 47
Киреев С.Е. Параллельная реализация на суперЭВМ модели гравитирующей газопылевой системы 51
Бутюгин С.Д., Петухов А.В. Экспериментальный сравнительный анализ МКЭ и МКО для моделирования трехмерных гармонических электромагнитных полей. 56
Петушкеев Б.Л. MPI-IO: Практические аспекты разработки масштабируемых приложений 61
Логинова А.В. Параллельная реализация гексагональной клеточно-автоматной модели волнового процесса 66
Трунов А.А., Старченко А.В., Турчановский И.Ю., Шкляев В.А. Моделирование на вычислительном кластере динамики тонкого трубчатого замагниченного моноэнергетического пучка электронов . 71
Шарифулина А.Е. Параллельная реализация вероятностного асинхронного клеточного автомата, моделирующего каталитическую реакцию окисления монооксида углерода (CO) на поверхности палладия (Pd110) 77
Берцун В.Н., Бородин А.В. Параллельные алгоритмы на графах 82
Дементьева Е.В., Карепова Е.Д. Анализ параллельных реализаций метода конечных элементов для моделей мелкой воды 87

Аркабаев Н.К., Шумилов Б.М., Эшаров Э.А., Кудуев А.Ж. Вариационное моделирование пространственных кривых с помощью нового типа эрмитовых кубических сплайн-вейвлетов…….92
Геренг Ю.В., Сидоренко Ю.Н. Выбор оптимальной структуры композита с использованием генетического алгоритма..... 97
Турсунов Д.А., Шумилов Б.М., Эшаров Э.А., Турсунов Э.А. Новый тип эрмитовых сплайн-вейвлетов пятой степени..... 101
Белов С.В., Жаровцев В.В. Схема «предиктор – корректор» с заданными свойствами для одномерных изотермических уравнений газовой динамики..... 106
Фомин А.А., Фомина Л.Н. Обоснование корректности полинейного рекуррентного метода решения разностных эллиптических уравнений..... 111
Цыденов Б.О. Численный метод расчета явления термического бара в озере Байкал..... 116
Турсунов Д.А., Эшаров Э.А., Бекмуратов А.Т. Приближенное решение дифференциальных уравнений четвертого порядка с помощью эрмитовых сплайн-вейвлетов пятой степени..... 121
Шапошников А.И., Шапошникова Е.В. О кластеризации в анализе движения в изображении с неподвижной камеры..... 126
Шерина Е.С. Исследование неоднородной структуры биологического объекта при пропускании электрического тока небольшой силы..... 131
Ефимов А.В. Генетический алгоритм организации мультипрограммного функционирования распределенных вычислительных систем..... 136
Курносов М.Г., Пазников А.А. Алгоритмы децентрализованной диспетчеризации параллельных программ в пространственно-распределенных вычислительных системах..... 140
Павский В.А., Павский К.В. Расчет показателей эффективности функционирования распределенных вычислительных систем при решении задач потока с отказами..... 145
Перепелкин В.А. Представление алгоритмов в технологии фрагментированного программирования..... 149
Власенко А.Ю. Подсистема автоматического обнаружения логических ошибок в MPI–программах информационно-вычислительного портала Кемеровского государственного университета..... 153
Ларин В.В. Задание управления во фрагментированных программах сетями Петри и проверка корректности управления..... 157
Щукин Г.А. Runtime-система FP RTS управления исполнением фрагментированных программ на мультикомпьютере..... 162

Федорова О.П. Примеры применения сплайнов, сохраняющих интеграл, при обработке изображения 167
Михайлов М.Д., Овчинникова К.Д. Учебной лаборатории кафедры вычислительной математики и компьютерного моделирования 20 лет 171

Особенности исполнения фрагментированных программ

В.Э. Малышкин
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассматриваются проблемы организации эффективного выполнения фрагментированных программ на мультипроцессорах.

В ИВМиМГ СО РАН разрабатывается система фрагментированного программирования для создания библиотеки численных алгоритмов. Основные требования к системе заключаются в следующем:

1. Фрагментированная программа собирается из готовых к исполнению фрагментов вычислений.
2. Фрагментированная структура программы сохраняется в ходе вычислений, т.е. каждый фрагмент вычислений образует независимо исполняющийся процесс.
3. Переменные – это переменные программы, вычисление – именуется каждое срабатывание операции.
4. Все технические детали параллельного программирования должны быть сконцентрированы в системном программном обеспечении .

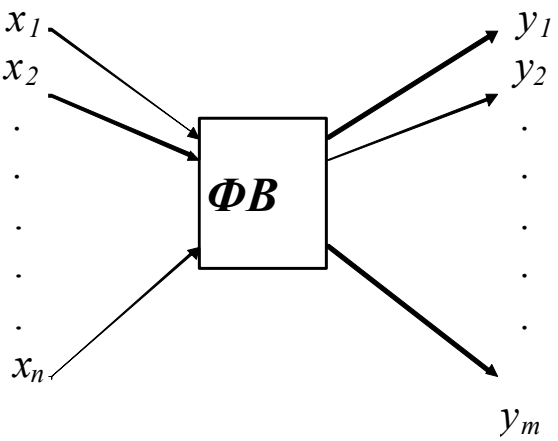


Рис. 1. Обращение к фрагменту вычисления

К фрагменту вычислений можно обратиться, например, как к обычной процедуре в последовательном языке программирования (рис. 1).

Технология, язык и система фрагментированного программирования разрабатываются для того, чтобы автоматически обеспечивать динамические свойства прикладной параллельной программы. Рассматриваются следующие необходимые динамические свойства параллельных программ:

1. При исполнении параллельной программы на мультикомпьютере в каждый его процессорный элемент загружается, по крайней мере, одна программа, а лучше несколько, со своими наборами входных данных для обработки. Таким образом, при исполнении параллельная программа представляется как множество параллельно протекающих и взаимодействующих последовательных процессов.
2. Недетерминизм исполнения. Для последовательного исполнения на множестве всех процессов должен быть установлен линейный порядок. Для параллельного исполнения на множестве процессов прикладной программы устанавливается частичный порядок и процессы исполняются, не нарушая этот порядок. При каждом конкретном исполнении ПП (с одними и теми же входными данными) ее процессы исполняются, вообще говоря, в разном порядке относительно друг друга и в случае ошибки ПП может вырабатывать разные результаты при разных ее исполнениях на одних и тех же данных.
3. Гибкая настройка программы на все доступные ресурсы мультикомпьютера. Серьезная проблема – динамическое распределение работы таким образом, чтобы обеспечить равномерную загрузку ресурсов мультикомпьютера (отображение прикладного алгоритма на ресурсы мультикомпьютера).
4. Динамическая балансировка загрузки. В зависимости от входных данных процессы могут потреблять разное количество ресурсов (например, исполняться за разное время в зависимости от числа итераций внутреннего цикла). В ходе исполнения ПП количество потребляемых процессами ПП ресурсов также может существенно изменяться. Во всех этих случаях нужна динамическая (в ходе исполнения) балансировка загрузки мультикомпьютера.
5. Переносимость в классе мультикомпьютеров. Параллельная программа должна быть хорошо переносима в классе мультикомпьютеров, т.е. хорошо исполняться на каждом мультикомпьютере из некоторого класса. ПП должна хорошо (если можно, оптимально) вы-

полняться на всех тех ресурсах, что ей выделены при текущем исполнении.

6. Динамизм поведения. При реализации численных моделей большого размера ПП должна уметь учитывать поведение модели и подстраиваться под него для достижения высокой производительности.
7. Реализация межпроцессорных коммуникаций на фоне счета.

Фрагментированная программа представляется четырьмя множествами: множество фрагментов данных, множество фрагментов кода, множество фрагментов вычислений и бинарное отношение частичного порядка на множестве фрагментов вычислений. Средства фрагментированного программирования поясним на простых примерах.

Алгоритм умножения матриц

1. Исходный алгоритм:

$$C = A \times B, \quad A = (a_{ij})_{i,j=\overline{1,N}}, \quad B = (b_{ij})_{i,j=\overline{1,N}}, \quad C = (c_{ij})_{i,j=\overline{1,N}},$$

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \times b_{k,j}, \quad i, j = 1,2,...,N.$$

2. Фрагментированный алгоритм

$$C_{I,J} = \sum_{L=1}^{K} A_{I,L} \times B_{L,J} = \sum_{L=1}^{L=K} C^{L}{}_{I,J}.$$

Здесь $C_{I,J}, A_{I,L}, B_{L,J}, C^{L}{}_{I,J} \quad I, J, L = 1,2,...,K.$

2.1.Множество фрагментов данных

$C_{I,J}, A_{I,L}, B_{L,J}, C^{L}{}_{I,J} \quad I, J, L = 1,2,...,K.$

2.2.Множество фрагментов вычислений

1) $\Phi_{IJ}^{L} = C_{I,J}^{L} = A_{I,L} \times B_{L,J}$,

2) $\Phi C_{IJ} = C_{I,J} = \sum_{L=1}^{K} C_{I,J}^{L}$.

2.3.Порядок:

$\forall L$ *и фиксированных I и J,*

$\Phi_{IJ}^{L} < \Phi C_{IJ}$.

Для простоты понимания можно представлять себе каждый фрагмент вычислений как процедуру. Для исполнения фрагментированных программ разработана run-time система, которая работает по общему правилу: если все входные переменные фрагмента вычисления получили значения (готовы к исполнению), то фрагмент может быть (но не обязательно будет) назначен на исполнение. В силу задания управления бинарным отношением частичного порядка многие фрагменты будут готовы к исполнению, у run-time системы выбор очередного

фрагмента на исполнение будет слишком велик и нередко может быть сделан неудачно, т.е. качество исполнения ФП будет неприемлемым.

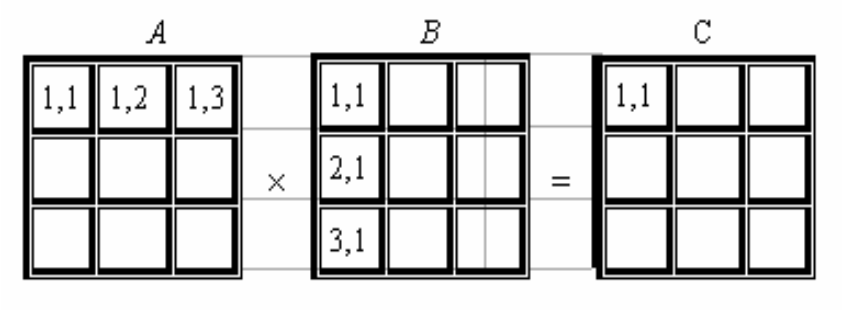


Рис. 2. Фрагментированная программа умножения матриц

Для фрагментированной программы умножения матриц примера такое выполнение фрагментов вычислений, в котором сначала выполняются все C^{L1}_{IJ}, где $L1$=1,2,3, неприемлемо, так как происходит накапливание фрагментов данных C^{L1}_{IJ} для всех I и J без их потребления.

Уже на этом простом примере видны проблемы организации исполнения фрагментированных программ, выбора желаемого фрагмента на исполнение. Аналогичные проблемы возникают и в процессе динамического распределения вычислительных ресурсов мультикомпьютера.

Литература

1. **Valkovskii V.A., Malyshkin V.E.** Parallel Program and System Synthesis on the Basis of Computational Models. Novosibirsk: Nauka, 1988. 128 p.
2. **Kraeva M.A., Malyshkin V.E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers // Int. Journal on Future Generation Computer Systems. 2001. Vol. 17, No. 6, P. 755-765.
3. **Malyshkin V.E.** Assembling of Modeling Parallel Programs // Handbook of Research on Scalable Computing Technologies. 2009. Chapter 13. IGI Global. P. 295-311.
4. **Malyshkin V.E., Sorokin S.B., Chajuk K.G.** Fragmentation of Numerical algorithms for the Parallel Subroutines Library // In Proceedings of the PaCT-2009 Int. conference, LNCS, 2009.

Разработка информационного портала параллельных вычислений для проведения научных и инженерных расчетов в режиме on-line

К.Е. Афанасьев, Н.Н. Окулов, С.В. Стуколов
Кемеровский государственный университет

Рассматривается информационно-вычислительный портал КемГУ, основными возможностями которого являются: проведение численных экспериментов, хранение, поиск и обработка информации, получаемой в результате расчетов; сбор, систематизация, методическая обработка информационных образовательных ресурсов, гибкая система поиска ресурсов; информационная поддержка проведения научных конференций. Также рассмотрены важнейшие подсистемы портала, предоставляющие возможность проведения научных и инженерных расчетов.

Введение

Стремительное развитие глобальных информационных и вычислительных сетей ведет к изменению фундаментальных парадигм обработки данных, которые можно охарактеризовать как переход к распределенным ресурсам и создание инфраструктуры для свободного доступа к ним. Разрабатываемые технологии должны предоставлять возможности для точного и адекватного удовлетворения потребностей пользователей, обращающихся к информации или вычислительному ресурсу. Комбинация возможности доступа к данным через базу метаданных, службы каталогов, зарегистрированного набора методов обработки данных позволяет решать задачи для создания среды обработки данных и управления вычислительными ресурсами.

Создание информационно-вычислительного портала для работы с высокопроизводительными вычислительными ресурсами (ВПВР) позволит расширить сферу использования ВПВР в научных исследованиях и учебном процессе вузов.

Информационно-вычислительный портал

Информационно-вычислительный портал является единым пространством для организации учебной и научной деятельности вуза в сфере решения различных задач с использованием вычислительного эксперимента, проведения научных конференций, хранения и работы с

информационными ресурсами, неформального общения в тематических форумах, информировании о событиях, происходящих в учебном заведении.

Портал является основной технологической единицей общей инфраструктуры единой информационной научно-образовательной среды. По сути, Портал – это точка с определенным регламентом доступа к научно-образовательной жизни вуза (информационным ресурсам, научным конференциям, вычислительным экспериментам) независимо от места расположения потенциального пользователя, учебного заведения или места подключения пользователя к сети Интернет.

Основными возможностями портала являются:

- проведение численных экспериментов, хранение, поиск и обработка информации, получаемой в результате расчетов;
- сбор, систематизация, методическая обработка информационных образовательных ресурсов, гибкая система поиска ресурсов;
- информационная поддержка проведения научных конференций.

Учебный информационно-вычислительный портал представляет собой единую точку доступа к следующим компонентам: система безопасности, система удаленного доступа, каталог ресурсов, сервер конференций, система новостных лент, система форумов, инструменты поиска информации, система управления контентом.

Портал объединяет действующие системы: систему новостных лент, систему форумов, включает в себя каталог ресурсов и сервер конференций. В портал также входят система удаленного доступа и управления распределенными вычислительными ресурсами (УД и УРВР), система поддержки учебного процесса (СПУП) и инженерный пакет GiD.

Система удаленного доступа и управления распределенными вычислительными ресурсами

В современной науке для изучения физических явлений широко используется численный эксперимент. Для получения высокой точности результатов необходимо использовать математические модели, как можно более точно описывающие изучаемые процессы. Чтобы провести численный эксперимент за приемлемое время, необходимо использовать высокопроизводительные вычислительные ресурсы, т.е. ресурсы с параллельной архитектурой.

Поэтому создание системы УД и УРВР, являющейся информационной средой для проведения численных экспериментов и хранения результатов, является актуальной и важнейшей задачей повышения эффективности и удобства использования ВПВР в научных и инженерных исследованиях.

Данная система сочетает в себе возможности удаленного доступа с графическим интерфейсом пользователя, пакетной обработки заданий и мониторинга состояния ресурсов. Для обеспечения заданной функциональности в системе реализованы следующие возможности:

- хранение файлов расчетных программ пользователей ИС в виде исходного и/или бинарного кода, файлов начальных данных, результатов и другой информации, необходимой для проведения научных расчетов;
- создание, удаление и управление пользователем своими объектами (файлами, расчетами);
- хранение информации о доступных вычислительных ресурсах;
- отслеживание состояния доступных вычислительных ресурсов;
- передача команд на удаленные вычислительные ресурсы;
- определение очередности запуска программ в случае существования конкуренции за вычислительные ресурсы;
- предоставление пользователю удобного графического интерфейса для работы с системой в удаленном режиме.

Принимая во внимание распределенный характер обработки информации в данной системе, для нее была выбрана многозвенная клиент-серверная архитектура.

Для работы с системой пользователь использует программу-клиент (браузер), которая выполняет функции отображения данных, т.е. обеспечивает интерфейс.

Виртуальная лаборатория

Компонент «Виртуальная лаборатория» системы УД и УРВР разработан с целью организации виртуального лабораторного практикума с использованием высокопроизводительных вычислительных ресурсов в удаленном режиме в рамках курсов по высокопроизводительным вычислениям.

Создание информационной системы поддержки учебного процесса с возможностью проведения виртуальных практикумов является акту-

альной задачей информационного обеспечения учебно-научного процесса.

«Виртуальная лаборатория» является компонентом системы УД и УРВР, поэтому построена по аналогичной архитектуре и интегрирована в схему БД системы УД и УРВР. «Виртуальная лаборатория» взаимодействует со СПУП, являющейся частью информационно-вычислительного портала КемГУ.

Преподаватель может создавать два типа лабораторных заданий:

1. *Анализ заданной программы.* Данный тип задания предполагает исследование студентом параметров заранее подготовленной преподавателем программы (например, зависимость эффективности программы от количества вычислительных узлов кластера или зависимость точности вычислений от размерности задачи). Для данного типа задания преподаватель обязан посредством web-интерфейса разместить в системе исходный код программы, указать количество параметров, их имя, тип и диапазон принимаемых значений (например, размерность задачи может указываться параметром SIZE типа INT в диапазоне от 100 до 10000), указать архитектуру и операционную систему вычислительного ресурса, а также компилятор, которым можно откомпилировать данный исходный код. Значения входных параметров будут переданы программе в виде аргументов в той последовательности, как они заданы преподавателем для данной работы.

2. *Разработка программы.* Данный тип задания не требует от преподавателя предоставления готового исходного кода. Студент самостоятельно пишет исходный код, размещает его в системе удаленного запуска заданий на вычислительных ресурсах, запускает задание и анализирует результаты.

Препроцессор и постпроцессор GiD

Пакет GiD представляет собой универсальное адаптивное средство для подготовки начальных данных и визуализации результатов численного моделирования. Основное назначение GiD – препроцессор для подготовки начальных данных для программ-решателей. GiD идеально подходит для генерации информации для численных расчетов: структурированной и неструктурированной сетки, начальных и граничных условий. Пакет GiD разработан в центре CIMNE при Каталонском государственном университете (Испания), в России GiD представляет кафедра ЮНЕСКО по НИТ Кемеровского государственного университета. Сотрудниками кафедры была выполнена русификация пакета.

Интеграция пакета GiD в структуру информационно-вычислительного портала КемГУ позволила расширить сферу применения портала, а также проводить сложные научные и инженерные расчеты в режиме on-line.

О быстродействии подсистемы памяти узлов вычислительного кластера

Д.В. Биматов, Д.С. Сущенко, С.П. Сущенко
Томский государственный университет

Введение

В современных вычислительных кластерах в качестве узлов широко используются компьютеры общего назначения, в которых для обеспечения быстрого доступа к наиболее часто используемым данным применяется многоуровневая организация памяти. Основным узким местом современных вычислительных систем является тракт процессор–память [1,2]. Разрыв скорости обработки данных в центральном процессоре и скорости доступа к операндам в оперативной памяти современных компьютеров составляет 1-2 порядка.

Главными требованиями, которым должна удовлетворять подсистема памяти, являются достаточно большая емкость, высокое быстродействие и экономическая эффективность с точки зрения технической реализации [3]. Желательно также, чтобы при минимальных физических размерах память обладала как можно большей информационной емкостью. Удовлетворить все эти требования одновременно в одном устройстве невозможно, поэтому обычно комбинируют несколько запоминающих устройств с различными параметрами, добиваясь создания комплексного решения с требуемыми характеристиками. Обычно оперативная память и механизмы, реализующие виртуальную память, являются средством обеспечения большой информационной емкости, а буферная память (кэш) – средством повышения быстродействия подсистемы памяти.

Таким образом, многоуровневая организация памяти позволяет сгладить разрыв между скоростью обработки данных центральным процессором и скоростью доступа к адресуемым объектам в оперативной памяти вычислителя. Эффективность доступа к иерархической памяти определяется не только быстродействием и емкостью отдельных уровней подсистемы памяти, но и набором архитектурных параметров, важнейшими среди которых являются коэффициент ассоциативности, длина межуровневого интерфейсного блока и глубина неблокируемости кэш-памяти.

Ассоциативность определяет способ отображения пространства адресов оперативной памяти в кэш-память и, наряду со стратегией вытеснения кэш-строк при конфликте адресов, в значительной мере

влияет на степень локализации прикладных задач в памяти верхнего уровня. Степень локализации в, свою очередь, определяет вероятность «попадания» адресуемого объекта в кэш и время доступа к нему.

Как правило, различные уровни иерархической памяти работают асинхронно, поэтому между ними предусмотрены буферные устройства, которые хранят адреса востребованных вычислителем элементов данных и позволяют параллельно выполнять операции доступа к различным уровням. Емкость межуровневого буферного устройства определяет параметр глубины неблокируемости кэша и задает потенциально достижимую степень параллелизма выполнения совокупности транзакций доступа к различным уровням иерархической памяти.

Эксплуатационные характеристики вычислительной системы в значительной мере определяются не только указанными параметрами иерархической памяти, но и распределением востребованности процессором рассеянных в оперативной памяти адресуемых объектов решаемых пользователями прикладных задач.

Набор важнейших вероятностно-временных характеристик подсистемы иерархической памяти включает вероятность попадания в кэш, вероятность блокировки кэш-памяти, среднее время доступа к адресуемым объектам и пропускную способность подсистемы памяти. Кроме того, при разработке приложений, их настройке на конкретные применения и условия эксплуатации важными являются знание и прогноз пределов изменения операционных показателей пропускной способности и средней задержки доступа к адресуемым объектам в различных нагрузочных условиях. Чрезвычайно полезными являются также оценки времени заполнения кэш-памяти блоками памяти при переключении процессора на выполнение нового приложения.

Влияние ассоциативности на вероятность попадания в кэш

Основным фактором влияния на этот показатель является распределение вероятностей востребованности блоков памяти прикладной задачи центральным процессором вычислительного узла. Кроме того, частоту попадания в кэш в значительной мере определяет стратегия вытеснения блоков из группы кэша при возникновении конфликта адресов [3]. Очевидно, что самой предпочтительной является идеальная стратегия вытеснения самого неиспользуемого блока в группе (имеющего наименьшую вероятность быть востребованным). По оценкам специалистов, наиболее близка к идеальной стратегия LRU [3], имеющая умеренную трудоемкость реализации.

Процессы перемещения адресуемых объектов между уровнями иерархической памяти для идеальной стратегии вытеснения хорошо описываются цепями Маркова. В предположении упорядоченности по убыванию вероятностей востребованности блоков памяти процессором вычислителя с ростом адресов блоков, отображаемых на заданную группу кэша ассоциативности A, динамика процесса замещения блоков данных в кэше моделируется случайным Марковским процессом в A-мерном пространстве. При этом переходные вероятности цепи Маркова, описывающей факты вытеснения наименее вероятных к использованию в группе кэша и подкачки востребованных вычислителем блоков памяти в кэш, имеют вид перемещений вдоль координатной оси с максимальным исходным значением координаты.

В стационарных условиях цепь Маркова, описывающая функционирование группы кэша, распадается на совокупность замкнутых множеств состояний, образующих $A!$ неприводимых подцепей Маркова. Вероятности обнаружения блоков памяти в заданной группе кэша при этом имеют следующий вид [4]:

$$\Pi_a = 1,\ a = \overline{0, A-2};\Pi_m = \frac{f(m)}{1-\sum_{k=0}^{A-2} f(k)},\ m = \overline{A-1, M-1}.$$

Здесь $f(m)$ – условная вероятность востребованности вычислителя в m-м блоке памяти, отображаемом на одну группу кэша; M – количество блоков памяти, отображаемых на одну группу кэша.

Отсюда следует, что в каждой группе кэша все строки кроме одной содержат самые востребованные вычислителем блоки памяти, а одна строка используется для попеременной подкачки недостающих вычислителю адресуемых объектов. Вероятность попадания адресуемого объекта в кэш с идеальным вытеснением определяется зависимостью [4]

$$\Pi = \sum_{g=0}^{G-1}\left[\sum_{m=0}^{A-2} p(g,m) + \frac{\sum_{m=A-1}^{M-1}\left[p(g,m)\right]^2}{\sum_{m=A-1}^{M-1} p(g,m)}\right],$$

где G – число групп кэша; $p(g,m)$ – вероятность потребности вычислителя в m-м блоке памяти, отображаемом на g-ю группу кэша.

Из данного соотношения нетрудно видеть, что вероятность попадания достигает максимального значения при полностью ассоциативном кэше, в котором только одна строка единственной группы используется для замещения блоков памяти, в то время как в других находится

самая востребованная процессором информация. В целом для широкого спектра распределений востребованности адресуемых объектов прикладных задач в оперативной памяти зависимость вероятности попадания от коэффициента ассоциативности имеет характер кривой с насыщением в области $A > 10$. В тех случаях, когда приложение полностью локализовано в кэше или же востребовано процессором в соответствии с равномерным законом, вероятность попадания инвариантна к ассоциативности кэша.

С ростом размера приложений вероятность попадания монотонно снижается и при достижении объема прикладных задач до 5-10 размеров кэш-памяти стабилизируется на определенном уровне. Следует отметить также, что время заполнения кэша самыми востребованными процессором блоками памяти для идеальной стратегии вытеснения распределено по геометрическому закону со снижающимся параметром распределения по мере загрузки наиболее используемых адресуемых объектов. При этом параметр распределения снижается на значение, равное вероятности востребованности загруженного блока памяти, если его номер лежит в пределах от 0 до A–2.

Влияние глубины неблокируемости на среднее время доступа к подсистеме памяти

Для получения аналитических зависимостей среднего времени доступа к адресуемым объектам от параметров вычислителя подсистема памяти описывается стохастическим конвейером с нарастающим временем обработки в последовательных фазах, случайным количеством последовательных фаз обработки операции доступа к адресуемому объекту, определяемым вероятностями промаха на каждом уровне иерархической памяти, и длиной, равной числу уровней иерархии.

Стохастический конвейер описанной структуры моделируется многоэтапной системой массового обслуживания с поступлением транзакций доступа к заданному уровню подсистемы иерархической памяти в темпе промахов в предыдущий уровень иерархии, детерминированным обслуживанием и дискретным временем. Время между поступлениями заявок в систему массового обслуживания распределено по геометрическому закону с параметром, определяемым вероятностью промаха в кэш первого уровня, а число этапов обслуживания – временем обработки заявки на следующем уровне иерархии, вычисленным в длительностях циклов доступа к первому уровню подсистемы памяти.

Среднее время доступа к адресуемым объектам для двухуровневой подсистемы памяти блокирующего типа определяется линейной зави-

симостью от вероятности промаха в кэш и складывается из времени обращения к кэш-памяти и, с вероятностью промаха, времени выборки объекта из оперативной памяти. С ростом глубины неблокируемости зависимость среднего времени доступа принимает нелинейный вид от вероятности попадания. При этом для низких уровней попадания средняя задержка превышает данный показатель для подсистемы памяти блокирующего типа, а для высоких – оказывается ниже.

Область значений вероятности попадания, обеспечивающих выигрыш в среднем времени доступа для каждой следующей величины коэффициента неблокируемости, сужается, из чего следует, что большие значения этого параметра следует применять для сильно локализованных задач, имеющих низкие вероятности промаха. Отметим, что рост глубины неблокируемости при прочих равных условиях всюду приводит к увеличению пропускной способности подсистемы памяти до потенциально возможных значений.

Литература

1. **Корнеев В.В., Киселев А.В.** Современные микропроцессоры. М.: Нолидж, 1998.
2. **Корнеев В.В.** Параллельные вычислительные системы. М.: Нолидж, 1999.
3. **Кохонен Т.** Ассоциативная память. М.: Мир, 1980.
4. **Сущенко М.С.** Анализ производительности множественного ассоциативного кэша // Вестник ТГУ. 2002. № 275. С. 218-223.

О проблеме начального обучения параллельному программированию

Л.В. Городняя
Институт систем информатики им. А.П. Ершова СО РАН,
Новосибирск

Обсуждается проблема обучения параллельному программированию. Дана характеристика языков программирования, изучаемых в курсе «Парадигмы программирования», для студентов ФИТ и ММФ НГУ. На этапе перехода к многопроцессорным архитектурам возрастает актуальность обучения параллельному программированию, что требует развития языково-информационной поддержки введения в программирование. Предлагается проект языка начального обучения параллельному программированию. Работа поддержана грантом РФФИ 08-01-00899-а.

Введение

Мир программирования и его техническая основа претерпели значительные изменения за последние двадцать лет. Возрастает актуальность обучения параллельному программированию, что требует новой языково-информационной поддержки введения в программирование [1]. Для систематичной подготовки специалистов в области параллельного программирования необходима гибкая инструментальная поддержка учебной практики, поддерживающая разные модели параллелизма. Однако для многих языков высокого уровня (ЯВУ) характерно включение в семантику языка одной модели параллелизма. Поэтому становится актуальной проблема создания языка начального обучения параллельному программированию (ЯНОП), предоставляющего основные модели параллелизма для обучения результативному представлению естественного параллелизма при решении важных задач.

При разработке ЯНОП рассмотрен ряд моделей параллелизма, предложенных в языках программирования Apl, Algol-68, Ada, Setl, Sisal, F# и др. Проанализированы требования к языковой и инструментальной поддержке процесса обучения параллельному программированию. Проект учебного языка параллельного программирования нацелен на ознакомление с основными моделями параллелизма, встречающимися в учебно-методических и научных материалах [2], языках высокого уровня, языках высокопроизводительного программирования и при организации сетевых информационных сервисов.

Определены уровни и цели начального (полгода в 10–12 лет) и предпроизводственного (2 года с/к для старшеклассников и студентов) обучения в контексте истории языков программирования. Проанализированы текущие задачи разработки учебных языков программирования, дополняющих основные парадигмы программирования на ЯВУ (императивное, функциональное, логическое, объектно-ориентированное) средствами параллельного программирования (GPU, MPI, Open MP).

Сформулированы требования к языкам начального обучения параллельному программированию:

- Концентричность или разложение понятий по уровням обучения.
- Представление и преобразование моделей управления процессами.
- Определение видов памяти с разной дисциплиной доступа.
- Настраивание на реалии производственных языков и систем.

Выбор решений

Многие затруднения с параллельным программированием имеют образовательный характер. Концепции взаимодействующих процессов следовало бы осваивать при начальном знакомстве с информационными технологиями и программированием.

При проектировании ЯНОП учтены факторы успеха наиболее известных учебных языков программирования, таких как Basic, Pascal, Logo, Grow, Робик и др., с целью проявления их специфики и типовых свойств. Ведущее понятие языка начального обучения программированию Робик – «исполнитель», причем исполнителей может быть много, и они могут обладать разными системами команд. Такой подход в ЯНОП допускает естественное развитие представления программ в направлении моделирования асинхронных процессов.

Программы в ЯНОП строятся из потоков, а потоки – из действий, выполнение которых может быть обусловлено ожиданием времени (пауза) или другого сигнала, предикатом или вероятностью срабатывания. Последнее означает, что при исполнении ведется учет частоты выполнения вероятностных действий, как в системах для разработки компьютерных игр. Кроме того, можно объявить планируемую длительность выполнения действий.

При организации сложных данных и процессов используются общие структуры или средства композиции, такие как списки, множества

и варианты, обеспечивающие представление отношений «после», «одновременно» и «взаимоисключение».

Представление циклов на ЯНОП наследует решения функционального языка параллельного программирования Sisal, а именно, выделяются участки для формирования пространства распараллеливаемых итераций и для свертки полученных параллельно значений в общий результат.

Проект учебного языка параллельного программирования

Реализация ЯНОП образует три уровня – элементарно-учебное ядро, технолого-прагматическое расширение и вводно-ознакомительную оболочку.

Ядро реализации ЯНОП составляют базовые средства уровня операционной системы (ОС) – условия готовности и завершения процесса, очереди и серийная обработка, типичные для ОС UNIX и языка Java. Такие средства рассматриваются как основа для реализации схем управления, встречающихся в ЯВУ (критические участки и семафоры, рандеву, защищенные команды, конвейеры), навыки программирования и применения которых входят в задачу элементарно-учебного уровня.

Остальное формируется как расширения ядра языка, выполняемые с целью достижения понимания реализационной семантики параллельного программирования для современных средств, что и является задачей технолого-прагматического уровня, обеспечивающего решение учебных задач, приведенных в [1,2]. Отдельные приемы программирования на ЯНОП показаны ниже.

Пример 1. Программа масштабируемого решения олимпиадной задачи для младших школьников: «Как поджарить 3 котлеты на 2 сковородах за минимальное время?». Типовое решение: «Поджарив первую сторону двух котлет, одну котлету отложить на тарелку, а на освободившуюся сковороду положить третью котлету и т.д.». Здесь после жарки одной стороны освобождаются обе сковороды.

```
ОЧЕРЕДЬ жарить = (2 2 2),%% Предстоит обжарить 2 стороны 3 котлет
          готово = ( ) ;   %% Дана пустая тарелка для готовых котлет
ВЕКТОР сковороды [1..2] ;  %% Имеется 2 сковороды для жарки

  ПОКА жарить ПОВТОР     %%  цикл работает до опустошения очереди

    жарить → сковороды  ;  %%  в вектор из очереди переданы два элемента
                           %%  переданные элементы исчезают из очереди
```

```
ЖДУ 5;                                  %% Время «жарки»
сковороды = (сковороды - 1) ;           %% Одна сторона обжарена

%% Разделяем котлеты на  готовые и требующие жарки
[ (жарить ; (сковороды ≠ 0) ), %% «Недожаренные» идут в очередь на жарку
(готово ; (сковороды = 0) ) ] ;     %% «Готовые»  -  в тарелку (со 2-го витка)

РЕЗУЛЬТАТ  готово ;                     %% = (0 0 0 )
```

Сравнения *(сковороды ≠ 0)* и *(сковороды = 0)* выполняют роль фильтров: выделяют элементы, удовлетворяющие сравнению. Выражение **(**x **;** y**)** – пополнение очереди X элементами из Y.

Пример 2. Варианты программы «**Простой торговый автомат**» [2].

!x – прием X, **? x –** выдача X.

Последовательно: *прием монетки,* *потом выдача шоколадки*	Асинхронно: *Прием монетки и выдача шоколадки* *в любом порядке*
ПОВТОР **(!** мон **; ?** шок **)**	ПОВТОР **[!** мон **, ?** шок **]**

Пример 3. Расстановка барьеров для взаимодействия процессов

```
ПТА =                                   УЧЕТ =
   ПОВТОР                               шоколадки = N
      ( ! мон ; ? шок ;                    ПОКА шоколадки ПОВТОР
      Контроль:          %% барьер            Контроль:          %% барьер
         ИНОГДА%  ? сувенир                   шоколадки = шоколадки – 1
%% иногда выдается «приманка»          СТОП
)

ПТА_С_УЧЕТОМ = ( Контроль ) [ ПТА , УЧЕТ ]  % общий барьер в потоках
```

Пример 4. Переименование барьера и подстановка действий

```
УЧЕТ_Б = (  N  b )          ПОКА N  ПОВТОР   b:  N = N – 1

Ф_ПТА_Б =  (мон шок p)        ( ! мон ; ? шок ;   p:  Ф_ПТА_Б )

    (Контроль) [ Ф_ПТА_Б ( жетон, игрушка, Контроль ) ,
                                УЧЕТ ( 4, Контроль )]   % в автомате 4
игрушки
```

Литература

1. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
2. **Хоар Ч.** Взаимодействующие последовательные процессы. М.: Мир, 1989. 264 с.
3. **Робик** [Электронный ресурс] / Википедия. Свободная энциклопедия. 2009 Режим доступа: http://en.wikipedia.org/wiki/Cloud_computing, свободный.

Параллельная реализация численного метода решения обратных задач переноса примеси

А.В. Старченко, Е.А. Панасенко
Томский государственный университет

Рассматриваются прямые и обратные задачи переноса примеси. При решении обратной задачи на каждом шаге по времени интегрируется N независимых сопряженных уравнений. Такие условия проведения численного исследования позволяют привлекать высокопроизводительную вычислительную технику. При этом используются следующие способы параллельной реализации: геометрическая декомпозиция, функциональная декомпозиция, комбинация функциональной и геометрической декомпозиций.

В настоящее время наряду с приборным контролем качества воздуха широко используются методы математического моделирования при анализе характера распространения газодисперсных загрязнителей атмосферы. Методы математического моделирования позволяют быстро и с малыми затратами выполнять прогноз детальной картины распределения примесей в приземном слое атмосферы. Но сложность и взаимосвязанность процессов распространения и переноса загрязняющих субстанций делают модели оценки и прогнозирования качества воздуха громоздкими в математической записи и весьма требовательными к вычислительным ресурсам. Перспективным способом решения этих проблем является использование суперкомпьютерной техники, которая обеспечивает существенное ускорение получения результатов расчетов.

Физическая постановка прямой задачи переноса примеси

При известном местоположении источника атмосферных загрязнений, расположенного на высоте h, мощностью $Q(t, x, y, z)$ определить значения концентрации примеси в N точках.

Математическая постановка прямой задачи переноса примеси

С учетом принятой физической постановки задачи адвективно-диффузионное уравнение, моделирующее перенос газообразной примеси в заданном потоке, представляется в следующем виде:

$$\frac{\partial C}{\partial t}+U\frac{\partial C}{\partial x}+V\frac{\partial C}{\partial y}+W\frac{\partial C}{\partial z}=$$
$$=\frac{\partial}{\partial x}\left[D\frac{\partial C}{\partial x}\right]+\frac{\partial}{\partial y}\left[D\frac{\partial C}{\partial y}\right]+\frac{\partial}{\partial z}\left[K_z\frac{\partial C}{\partial z}\right]+Q(t,x,y,z), \quad (1)$$

где C – концентрация примеси; U,V,W – компоненты вектора скорости атмосферного воздуха; D,K_z – коэффициенты турбулентной диффузии; Q – интенсивность поступления примеси от источников.

Начальные и граничные условия имеют следующий вид:

$$t=0:\ C=0;$$
$$x=0:\frac{\partial C}{\partial x}=0;\ x=L_x:\frac{\partial C}{\partial x}=0;\ y=0:\frac{\partial C}{\partial y}=0;\ y=L_y:\frac{\partial C}{\partial y}=0; \quad (2)$$
$$z=0:K_z\frac{\partial C}{\partial z}=\alpha C;\ z=L_z:\frac{\partial C}{\partial z}=0.$$

Физическая постановка обратной задачи переноса примеси

Требуется по известным метеорологическим параметрам атмосферы и результатам измерения концентрации примеси в N точках наблюдения, проводимых в течение некоторого периода времени T, определить координаты и время срабатывания мгновенного точечного источника примеси.

Математическая постановка обратной задачи

В данной работе для решения обратных задач привлекается подход Г.И. Марчука [1]. Математическая постановка обратной задачи переноса примеси включает несколько однотипных сопряженных уравнений и поиск минимума некоторого функционала в процессе решения сопряженных уравнений.

Сопряженная к (1) и (2) задача получается следующим образом: уравнение (1) умножается на некоторые сопряженные функции $C_k^*=C_k^*(t,x,y,z)$ и интегрируется по времени и пространству. Тогда, используя интегрирование по частям, приходим к выражению

$$-\frac{\partial C_k^*}{\partial t}-\frac{\partial UC_k^*}{\partial x}-\frac{\partial VC_k^*}{\partial y}-\frac{\partial WC_k^*}{\partial z}-$$
$$-\frac{\partial}{\partial x}\left(D\frac{\partial C_k^*}{\partial x}\right)-\frac{\partial}{\partial y}\left(D\frac{\partial C_k^*}{\partial y}\right)-\frac{\partial}{\partial z}\left(K_z\frac{\partial C_k^*}{\partial z}\right)=P_k;k=1,...,N \quad (3)$$

с соответствующими начальными и граничными условиями

$$C_k^*(T,x,y,z)=0;$$
$$x=0:UC_k^*+D\frac{\partial C_k^*}{\partial x}=0;\ x=L_x:UC_k^*+D\frac{\partial C_k^*}{\partial x}=0;$$
$$y=0:VC_k^*+D\frac{\partial C_k^*}{\partial y}=0;\ y=L_y:VC_k^*+D\frac{\partial C_k^*}{\partial y}=0; \quad (4)$$
$$z=0:K_z\frac{\partial C_k^*}{\partial z}=\alpha C_k^*;\ \ z=L_z:\frac{\partial C_k^*}{\partial z}=0,$$

где $P_k=\delta(x-x_k)\delta(y-y_k)\delta(z-z_k)\delta(t-t_k)$; N – количество точек наблюдения с координатами (x_k,y_k,z_k); t_k – момент времени измерения концентрации.

В качестве двойственного представления функционала рассматривается следующее выражение [2]:

$$Q_0C_k^*(t_{00},x_{00},y_{00},z_{00})=C_k;\ C_k=C(t_k,x_k,y_k,z_k);\ k=1,...,N. \quad (5)$$

При этом нахождение решения системы уравнений (5) позволяет установить параметры источника загрязнения (координаты (x_{00},y_{00},z_{00}), момент срабатывания источника t_k и интенсивность Q_0).

Численное решение сопряженной задачи

Для численной реализации задачи (3) – (4) использовались метод конечного объёма и явные разностные схемы; для аппроксимации адвективных членов применялась схема MLU Ван Лира [3].

Параллельная реализация сопряженной задачи

При численном решении задач переноса примеси выделяют два основных способа параллельной реализации: распараллеливание по физическим процессам (функциональная декомпозиция) и геометрическая декомпозиция расчетной области [4].

Функциональная декомпозиция

Распараллеливание по физическим процессам метода численного решения обратной задачи переноса примеси производилось с использованием принципа «master-slave». При использовании этого подхода управляющий master-процесс передает каждому slave-процессу значения метеорологических параметров, необходимые для решения сопряженных задач. Подчиненные slave-процессы, получив данные, в свою очередь, ведут расчеты независимо друг от друга, и найденные на каждом шаге по времени приближенные решения своей сопряженной задачи (3) – (4) возвращают управляющему процессу, который ищет глобальный минимум функционала:

$$\sum_{k=1}^{N}\left(Q_0 C_k^*(t,x,y)-C_k\right)^2, Q_0>0. \qquad (6)$$

Рассмотренный выше алгоритм параллельного решения обратной задачи с использованием принципа функциональной декомпозиции можно оценить следующими временными затратами:

$$T_p^{\Phi 1}\approx m\cdot t_{\text{а}}\cdot N_x^2\cdot N_z\cdot M+5\cdot t_{\text{об}}\cdot N_x^2\cdot N_z\cdot M+t_{\text{об}}\cdot M+t_{\text{а}}\cdot(M+3N),\quad N=p-1. \qquad (7)$$

Здесь $t_{\text{а}}$ – время одной арифметической операции; $t_{\text{об}}$ – время на пересылку одного числа по маршруту «master-slave» или обратно; N – количество решаемых сопряженных задач. Первое слагаемое в (7) есть затраты на одновременное решение сопряженных задач, второе – передачу slave-процессам метеорологических полей в каждый момент времени, третье – затраты на передачу рассчитанных значений $Q_0 C_k^*(t,x,y)$ master-процессу, четвертый – расчет master-процессом функционала (6). Заметим, что поскольку размеры вычислительной сетки $N_x^2\cdot N_z$ составляют несколько сотен тысяч узлов, главный вклад в (7) дают первое и второе слагаемые.

Таким образом, для характерных значений определяющих параметров ($m\approx 10\div 100$, $N_x^2\cdot N_z\approx 0.5\div 1\cdot 10^6$, $M\approx 100\div 3000$, $N\approx 5\div 20$):

$$T_p^{\Phi 1}\approx m\cdot t_{\text{а}}\cdot N_x^2\cdot N_z\cdot M+5\cdot t_{\text{об}}\cdot N_x^2\cdot N_z\cdot M/k,\quad (M/k\approx 5\div 50), N=p-1, \qquad (8)$$

где k – частота межпроцессорной передачи метеорологических параметров.

Для случая проведения последовательных расчетов при решении обратной задачи переноса примеси временные затраты с учетом принятой модели оценки представляются следующим образом:

$$T_1 \approx N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M + t_{\text{а}} \cdot (M + 3N). \qquad (9)$$

Поскольку второе слагаемое в (9) существенно меньше первого, то для рассматриваемого алгоритма решения обратной задачи переноса примеси на p-процессорной системе получаются следующие оценки ускорения и эффективности:

$$S_p^{\Phi 1} = \frac{T_1}{T_p^{\Phi 1}} \approx \frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M + 5t_{\text{об}} \cdot N_x^2 \cdot N_z \cdot M / k} =$$

$$= \frac{p-1}{1+\dfrac{5\chi}{m \cdot k}}, \quad E_p^{\Phi 1} = \frac{S_p}{p} \approx \frac{1-1/p}{1+\dfrac{5\chi}{m \cdot k}}, \quad \chi = t_{\text{об}} / t_{\text{а}} > 1. \qquad (10)$$

Из (10) видно, что при таком способе организации параллельных вычислений не удается достичь идеального параллелизма, поскольку нет хорошей балансировки загрузки используемых в расчетах p-процессов [4. С. 57; 5. С. 135]. Это связано с тем, что master-процесс не участвует в проведении массовых вычислений (не решает отдельную сопряженную задачу).

В качестве другого подхода повышения эффективности данного алгоритма может быть рассмотрен подход, в котором master-процесс также решает одну из N сопряженных задач (3) – (4). В этом случае также можно воспользоваться оценкой (8), однако в рассмотренном случае $N = p$. Тогда

$$S_p^{\Phi 2} \approx \frac{p}{1+\dfrac{5\chi}{m \cdot k}}, \; E_p^{\Phi 2} \approx \frac{1}{1+\dfrac{5\chi}{m \cdot k}}. \qquad (11)$$

В табл. 1 и 2 представлены значения времени выполнения параллельных программ, полученных при численном эксперименте, реализующих два варианта функциональной декомпозиции при решении обратной задачи при различном количестве доступных измерений концентрации выброса $(N = 5,10,20)$.

Из таблиц видно, что первый вариант функциональной декомпозиции алгоритма решения обратной задачи имеет небольшое преимущество во времени выполнения параллельной программы, что обусловлено появлением дополнительных возможностей совмещения работы активных процессов, количество которых для одних и тех же условий на единицу больше, чем во втором варианте.

Таблица 1. Время счета параллельной программы (с) для первого варианта функциональной декомпозиции

Число используемых процессов (master не решает сопряженную задачу)	Время, с
$p = 6\,(N = 5)$	312
$p = 11\,(N = 10)$	316
$p = 21\,(N = 20)$	322

Таблица 2. Время счета параллельной программы (с) для второго варианта функциональной декомпозиции

Число используемых процессов (master решает сопряженную задачу)	Время, с
$p = 5\,(N = 5)$	323
$p = 10\,(N = 10)$	337
$p = 20\,(N = 20)$	354

Однако соотношение эффективности рассматриваемых параллельных программ

$$e^{\Phi} = \frac{E_p^{\Phi 1}}{E_p^{\Phi 2}} = \frac{T_1 / \left(T_p \cdot p\right)^{(\Phi 1)}}{T_1 / \left(T_p \cdot p\right)^{(\Phi 2)}} = \frac{\left(T_p \cdot p\right)^{(\Phi 2)}}{\left(T_p \cdot p\right)^{(\Phi 1)}} < 1$$

говорит в пользу лучшей масштабируемости второго варианта декомпозиции, что, кстати, хорошо согласуется с теоретическими оценками (10), (11), согласно которым $\left(e^{\Phi}\right)_{\text{теор}} \approx \frac{p}{p+1}$ (табл. 3).

Таблица 3. Отношение эффективности параллельных программ e^{Φ} для двух вариантов функциональной декомпозиции

Число измерений концентрации N	$\left(e^{\Phi}\right)_{\text{теор}} \approx \frac{p}{p+1}$	$\left(e^{\Phi}\right)_{\text{эксп}}$
5	0,83	0,86
10	0,91	0,97
20	0,95	1,01

Геометрическая декомпозиция

При решении каждой сопряженной задачи (3) – (4) использовалась одномерная (вдоль оси OY, рис. 1) геометрическая декомпозиция сеточной области. В этом случае вычисления осуществлялись в следующем порядке. N сопряженных задач (3) – (4) последовательно одна за другой решаются численно в подобласти вычислительной сетки, распределенной каждому процессу. Вычисления в течение промежутка времени $[0,T]$ проводятся каждым процессом одновременно, но для корректного расчета сеточных значений функции C^* вблизи гра-

ниц подобласти требуются сеточные значения с соседних по расположению подобластей. Это может быть обеспечено путём межпроцессорной передачи данных с использованием, например, функций блокирующего обмена MPI_SENDRECV стандарта MPI [5].

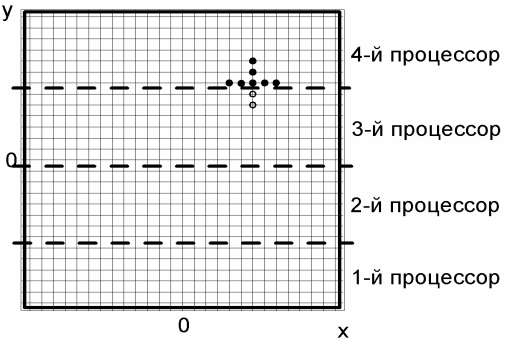


Рис. 1. Расчетная сетка в плоскости xOy с указанием распределения подобластей по четырем процессорам

Аналогично принципу функциональной декомпозиции проведем оценки временных затрат создаваемой параллельной реализации рассматриваемого алгоритма. Для этого запишем:

$$T_p^{\Gamma} \approx \frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z^2 \cdot M}{p} + \frac{5 \cdot t_{\text{об}} \cdot N_x^2 \cdot N_z \cdot M / k}{p} + \\ + 4 \cdot N_x \cdot N_z \cdot N \cdot M \cdot t_{\text{об}} + N \cdot M \cdot t_{\text{об}} + t_{\text{а}} \left(M + 3N \right). \qquad (12)$$

Здесь первое слагаемое есть затраты на решение N сопряженных задач в подобласти сеточной области, второе – передачу всем процессам от master-процесса, который также участвует в решении сопряженных задач, метеорологических полей с частотой k шагов, третье – затраты на передачу данных между соседними по расположению подобластей процессами, четвертое и пятое – получение и оценки минимума функционала (6) master-процессом. С учетом характерных для выбранной задачи значений параметров N_x, N_z, M, k, m, N справедливо

$$T_p^{\Gamma} \approx \frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{p} + \frac{5 \cdot t_{\text{об}} \cdot N_x^2 \cdot N_z \cdot M / k}{p} + \\ + 4 \cdot N_x \cdot N_z \cdot N \cdot M \cdot t_{\text{об}} \qquad (13)$$

и с учетом оценки временных затрат последовательной программы (9) имеем

$$S_p^{\Gamma} = \frac{T_1}{T_p^{\Gamma}} \approx$$

$$\approx \frac{N \cdot m \cdot t_a \cdot N_x^2 \cdot N_z \cdot M}{\frac{N \cdot m \cdot t_a \cdot N_x^2 \cdot N_z \cdot M}{p} + \frac{5 \cdot t_{об} \cdot N_x^2 \cdot N_z \cdot M / k}{p} + 4 \cdot N_x \cdot N_z \cdot N \cdot M \cdot t_{об}} =$$

$$= \frac{p}{1 + \chi\left(\frac{5}{Nmk} + \frac{4p}{N_x m}\right)}, \qquad (14)$$

$$E_p^{\Gamma} = \frac{S_p^{\Gamma}}{p} \approx \frac{1}{1 + \frac{\chi}{m}\left(\frac{5}{Nk} + \frac{4p}{N_x}\right)}.$$

Поскольку в полученной оценке эффективности рассматриваемого способа распараллеливания вклад межпроцессорной передачи данных во временные затраты более значительный, стоит ожидать, что применение геометрической декомпозиции менее перспективно, чем использование функциональной декомпозиции.

Проведенный вычислительный эксперимент показал правомерность полученной теоретической оценки (табл. 4).

Таблица 4. Время счета параллельной программы для геометрической декомпозиции

Число используемых процессов	Время, с
$p = 5\,(N = 5)$	627
$p = 10\,(N = 10)$	643
$p = 20\,(N = 20)$	743

Сопоставляя результаты, приведенные в табл. 1, 2, 4, видно, что применение геометрической декомпозиции более чем в два раза уступает по быстродействию функциональной декомпозиции. Причиной этого являются коммуникационные затраты на межпроцессорную передачу данных, необходимую для корректного решения сопряженных задач (3) – (4) на всей сеточной области, разделенной между активными процессами [6].

Комбинированный способ распараллеливания

Рассмотренные выше подходы создания параллельных версий алгоритмов решения обратной задачи переноса примеси показали неплохие результаты по эффективности, однако количество используемых активных процессов в них ограничено: в случае функциональной декомпозиции – числом проведенных измерений N, а при одномерной геометрической декомпозиции – размером сетки N_x и выбранным сеточным шаблоном.

При использовании комбинированного подхода параллельной реализации предлагается для увеличения в расчетах количества используемых активных процессов совместить применение функциональной и геометрической декомпозиций. Все p активных процессов делятся на N групп, каждая из которых решает численно одну сопряженную задачу (3) – (4), используя одномерную геометрическую декомпозицию. Для получения приближенного решения процессы отдельной группы обмениваются между собой рассчитанными значениями сеточной функции вблизи границ подобластей. Поиск минимума функционала (6) выполняет один из p процессов, которому остальные передают результаты промежуточного расчета.

Проведем теоретические предварительные оценки временных затрат такого способа параллельной реализации:

$$\begin{aligned} T_p^{\text{комб}} &\approx \frac{m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{p/N} + \frac{5 \cdot t_{\text{об}} \cdot N_x^2 \cdot N_z \cdot M/k}{p/N} + \\ &+ 4 \cdot N_x \cdot N_z \cdot M \cdot t_{\text{об}} + M \cdot t_{\text{об}} + t_{\text{а}} \left(M + 3N\right) \approx \\ &\approx \frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{p} + \frac{5 \cdot N_x^2 \cdot N_z \cdot M/k \cdot t_{\text{об}} \cdot N}{p} + \\ &+ 4 N_x \cdot N_z \cdot t_{\text{об}} \cdot M. \end{aligned} \tag{15}$$

Тогда

$$S_p^{\text{комб}} = \frac{T_1}{T_p^{\text{комб}}} \approx$$

$$\approx \frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{\frac{N \cdot m \cdot t_{\text{а}} \cdot N_x^2 \cdot N_z \cdot M}{p} + \frac{5 \cdot N \cdot t_{\text{об}} \cdot N_x^2 \cdot N_z \cdot M/k}{p} + 4 \cdot N_x \cdot N_z \cdot M \cdot t_{\text{об}}} = \tag{16}$$

$$= \frac{p}{1+\frac{\chi}{m}\left(\frac{5}{k}+\frac{4p}{N\cdot N_x}\right)},$$

$$E_p^{\text{комб}} = \frac{S_p^{\text{комб}}}{p} \approx \frac{1}{1+\frac{\chi}{m}\left(\frac{5}{k}+\frac{4p}{N\cdot N_x}\right)}, \quad \frac{p}{N} = 1,2,3,\ldots$$

Сравнивая оценки параллельных реализаций (14) и (16), можно отметить, что эффективность комбинированного подхода не хуже, чем при геометрической декомпозиции, но количество используемых активных процессов можно увеличить на порядок, что при имеющей место масштабируемости параллельной версии алгоритма должно привести к значительному сокращению времени счета (рис. 2).

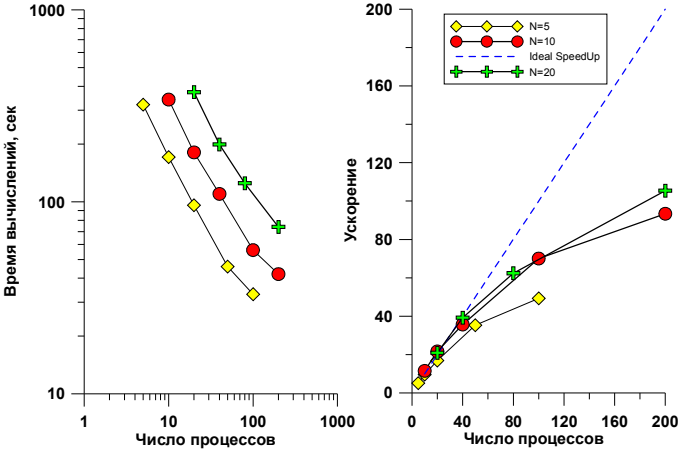


Рис. 2. Время счета (слева) и ускорение (справа) параллельной программы решения обратной задачи переноса примеси (комбинированный способ распараллеливания)

Результаты численных расчетов с использованием комбинированного подхода показывают снижение времени счета с уменьшением вычислительной трудоемкости задачи, замедление ускорения параллельной программы с ростом числа используемых процессов и при сокращении количества сопряженных задач N. Комбинированный подход действительно на порядок позволяет увеличить число используемых процессов, при этом эффективность параллельной программы остается на уровне не менее 50%, что является неплохим для задач такого класса показателем.

Заключение

В работе с единых позиций сформулированы математические постановки для решения обратных задач по определению параметров источника загрязнения атмосферного воздуха. Математические постановки опираются на аппарат сопряженных уравнений и двойственное представление функционала от концентрации примеси. Для ускорения численного решения задачи обоснованы и применены следующие способы параллельной реализации задач переноса примеси: функциональная декомпозиция, геометрическая декомпозиция и комбинированный способ. Проведены оценки ускорения и эффективности всех трех подходов параллельной реализации, которые показали, что в целом функциональная декомпозиция имеет существенное преимущество по сравнению с остальными подходами.

Литература

1. **Марчук Г.И.** Математическое моделирование в проблеме окружающей среды. М.: Наука. Главная редакция физико-математической литературы, 1982. 320 с.
2. **Панасенко Е.А.** Определение городских районов-загрязнителей атмосферного воздуха по данным наблюдений / Е.А. Панасенко, А.В. Старченко // Оптика атмосферы и океана. 2009. Т. 22, № 03. С. 279-283.
3. **Van Leer B.** Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second order scheme // J. Comput. Phys. 1974. Vol.14. P. 361-370.
4. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. 608 с.
5. **Малышкин В.Э., Корнеев В.Д.** Параллельное программирование мультикомпьютеров. Новосибирск: Изд-во НГТУ, 2006. 296 с.
6. **Старченко А.В.** Параллельные вычисления в задачах охраны окружающей среды // Вторая Сибирская школа-семинар по параллельным вычислениям. Томск: Изд-во Том. ун-та, 2004. С. 17-22.

Распараллеливание итерационных методов численного решения эллиптического уравнения со смешанными граничными условиями и анализ эффективности полученных параллельных реализаций

Е.А. Данилкин, А.В. Старченко
Томский государственный университет

Обсуждается параллельная реализация трех итерационных методов решения систем линейных алгебраических уравнений: метода Зейделя, явного метода Булеева и метода бисопряженных градиентов. Наилучшие результаты по количеству итераций, времени вычислений и масштабируемости показывает метод бисопряженных градиентов, предобусловленный с помощью метода Зейделя с красно-черным упорядочиванием.

Введение

За последнее десятилетие произошел большой скачок в развитии вычислительной техники. На смену постоянному увеличению тактовой частоты процессора пришли идеи параллельной обработки данных и использования многоядерных процессоров и многопроцессорных компьютеров. В США – лидере суперкомпьютерного рынка – уже построен петафлопный суперкомпьютер Roadrunner, использующий помимо идеи параллельной обработки данных комбинацию процессоров AMD Opteron и IBM Cell. В последние несколько лет популярным стало использование GPU для проведения расчетов. Естественно возникает вопрос: а современные численные методы переносятся на новые вычислительные машины, заложен ли в них параллелизм?

Постановка задачи

Сравнение различных методов решения системы линейных алгебраических уравнений осуществлялось на примере численного решения эллиптического уравнения в единичном кубе:

$$-\frac{\partial}{\partial x}\left(D\frac{\partial \Phi}{\partial x}\right)-\frac{\partial}{\partial y}\left(D\frac{\partial \Phi}{\partial y}\right)-\frac{\partial}{\partial z}\left(D\frac{\partial \Phi}{\partial z}\right)=1 \qquad (1)$$

с граничными условиями Дирихле на границе $x=1$ и условиями Неймана на остальных границах

$$\left.\frac{\partial\Phi}{\partial x}\right|_{x=0}=0;\Phi\big|_{x=1}=0;\left.\frac{\partial\Phi}{\partial y}\right|_{y=0}=0;\left.\frac{\partial\Phi}{\partial y}\right|_{y=1}=0;\left.\frac{\partial\Phi}{\partial z}\right|_{z=0}=0;\left.\frac{\partial\Phi}{\partial z}\right|_{z=1}=0\,, \quad (2)$$

где $D=1000$ *if* $0{,}1<x,y,z<0{,}9$ *else* $D=1$. Шаг сетки выбирался из условия, чтобы система линейных уравнений имела $120\times120\times120$ неизвестных. Использовалось нулевое начальное приближение.

Аппроксимация задачи осуществлялась на основе метода конечного объема с использованием равномерной сетки. Результатом аппроксимации дифференциального уравнения и граничных условий является их дискретный аналог – система линейных алгебраических уравнений, записанная в следующем виде:

$$\begin{aligned} ap_{i,j,k}\Phi_{i,j,k}+ae_{i,j,k}\Phi_{i+1,j,k}+an_{i,j,k}\Phi_{i,j+1,k}+at_{i,j,k}\Phi_{i,j,k+1}+\\ +aw_{i,j,k}\Phi_{i-1,j,k}+as_{i,j,k}\Phi_{i,j-1,k}+ab_{i,j,k}\Phi_{i,j,k-1}=b_{i,j,k}. \end{aligned} \quad (3)$$

Методы решения сеточных уравнений

Метод Зейделя. Для рассматриваемой задачи формулы метода Зейделя примут следующий вид:

$$\Phi^{n+1}_{i,j,k}=\frac{1}{ap_{i,j,k}}\begin{pmatrix} b_{i,j,k}-ae_{i,j,k}\Phi^{n}_{i+1,j,k} & -aw_{i,j,k}\Phi^{n+1}_{i-1,j,k}\\ -an_{i,j,k}\Phi^{n}_{i,j+1,k} & -as_{i,j,k}\Phi^{n+1}_{i,j-1,k}\\ -at_{i,j,k}\Phi^{n}_{i,j,k+1} & -ab_{i,j,k}\Phi^{n+1}_{i,j,k-1} \end{pmatrix}; \quad (4)$$

$$i=0,\ldots,Nx+1;\quad j=0,\ldots,Ny+1;\quad k=0,\ \ldots,Nz+1,$$

где n – номер итерации метода.

Метод бисопряженных градиентов (BiCGStab). Этот метод широко используется на практике и построен на основе выбора итерационных параметров из условия минимизации функционала, определяющего точность текущих последовательных приближений.

Перед началом итерационного процесса задаются входные данные: $r^0=b-A\Phi^0$; $\hat{r}^0$ – произвольный вектор, для которого $\left(\hat{r}^0,r^0\right)\neq0$, например, $\hat{r}^0=r^0$, $\rho_0=\alpha=\omega_0=1$, $v^0=p^0=0$;

$$\rho_i = \left(\hat{r}^0, r^{i-1}\right);$$
$$\beta_i = \left(\rho_i / \rho_{i-1}\right)\left(\alpha / \omega_{i-1}\right);$$
$$p^i = r^{i-1} + \beta_i \left(p^{i-1} - \omega_{i-1} v^{i-1}\right);$$
$$v^i = Ap^i;$$
$$\alpha = \rho_i / \left(\hat{r}^0, v^i\right);$$
$$s = r^{i-1} - \alpha v^i;$$
$$t = As;$$
$$\omega_i = \left(t, s\right) / \left(t, t\right);$$
$$\Phi^i = \Phi^{i-1} + \alpha p^i + \omega_i s;$$
$$r^i = s - \omega_i t.$$

Затем стартует итерационный процесс, который продолжается до тех пор, пока не будут выполнены условия сходимости.

Метод бисопряженных градиентов рассчитан на решение систем с несимметричной и знаконеопределенной в общем случае матрицей. На практике для ускорения сходимости используется процедура предобуславливания, которая состоит в замене исходной плохообусловленной системы $A\Phi = b$ системой $M^{-1}A\Phi = M^{-1}b$ с хорошим числом обусловленности [1].

Явный метод Булеева. Для решения системы линейных алгебраических уравнений с ленточной семидиагональной матрицей (3) используются рекуррентные формулы:

$$\Phi_{i,j,k} = +P_{i,j,k}\Phi_{i+1,j,k} + Q_{i,j,k}\Phi_{i,j+1,k} + R_{i,j,k}\Phi_{i,j,k+1} + T_{i,j,k};$$
$$i = Nx+1,0;\ \ j = Ny+1,0;\ \ k = Nz+1,0, \qquad (5)$$

где $P_{i,j,k}$, $Q_{i,j,k}$, $R_{i,j,k}$, $T_{i,j,k}$ – прогоночные коэффициенты, определяемые из:

$$\begin{cases} P_{i,j,k} = ae_{i,j,k} / sum; \\ Q_{i,j,k} = an_{i,j,k} / sum; \\ R_{i,j,k} = at_{i,j,k} / sum; \\ T_{i,j,k} = (aw_{i,j,k}(Q_{i-1,j,k}\Phi_{i-1,j+1,k} + R_{i-1,j,k}\Phi_{i-1,j,k+1} + T_{i-1,j,k}) + \\ \quad + as_{i,j,k}(P_{i,j-1,k}\Phi_{i+1,j-1,k} + R_{i,j-1,k}\Phi_{i,j-1,k+1} + T_{i,j-1,k}) + \\ \quad + ab_{i,j,k}(P_{i,j,k-1}\Phi_{i+1,j,k-1} + Q_{i,j,k-1}\Phi_{i,j+1,k-1} + T_{i,j,k-1}) + b_{i,j,k}) / sum; \\ sum = ap_{i,j,k} - aw_{i,j,k}P_{i-1,j,k} - as_{i,j,k}Q_{i,j-1,k} - ab_{i,j,k}R_{i,j,k-1}; \\ i = 0, Nx+1, \;\; j = 0, Ny+1, \;\; k = 0, Nz+1. \end{cases} \quad (6)$$

Таким образом, каждая итерация состоит из двух этапов: прямого (6) и обратного (5) хода. Во время прямого хода вычисляются значения прогоночных коэффициентов. Обратный ход состоит из расчета величин сеточной функции.

Распараллеливание итерационных методов решения СЛАУ и анализ эффективности полученных реализаций

При распараллеливании используется принцип геометрической декомпозиции сеточной области. В рассматриваемом случае возможны три различных способа разделения значений сеточной функции по вычислительным узлам – одномерная схема, двухмерное или трехмерное разбиение узлов вычислительной сетки. Каждому процессорному элементу вместе с выделенной сеточной подобластью распределяются все значения сеточной функции $\Phi^n_{i,j,k}$, принадлежащие этой подобласти.

Параллельная реализация метода Зейделя. Использование последних рассчитанных значений в методе Зейделя увеличивает скорость сходимости, но ограничивает параллелизм алгоритма и требует синхронизации параллельных вычислений. На практике допустимо использование асинхронного варианта метода Зейделя, в котором на границе подобластей декомпозиции используются данные с предыдущей итерации. Он часто оказывается эффективным, но при его использовании не представляется возможным проведение анализа сходимости численного метода.

Как приемлемая альтернатива используется обход узлов вычислительной сетки в шахматном порядке (красно-черное упорядочивание), такой способ разделения обусловлен выбором используемого конечно-разностного шаблона, в соответствии с которым для расчета в красных

узлах нужны значения в черных узлах и наоборот, что снимает вопрос о синхронизации параллельных вычислений [2].

Параллельная реализация метода BiCGStab. Этот метод не требует никаких других операций, кроме матрично-векторного произведения, вычисления скалярных произведений и сложения векторов. Все эти операции позволяют эффективно применять параллельные вычисления на компьютере с распределенной памятью. Параллельная реализация метода бисопряженных градиентов для систем с ленточными матрицами требует пересылок, а именно обмена приграничными значениями с границы подобластей декомпозиции для обеспечения однородности вычислений и сбора частичных сумм скалярного произведения со всех процессоров и рассылки полученного результата на все процессорные элементы.

Параллельная реализация явного метода Булеева. Параллельное решение сеточных уравнений методом Булеева при 1D, 2D или 3D декомпозиции расчетной области оказывается невозможным, потому как использование этого метода напрямую подразумевает многократный переход от одного узла расчетной сетки к другому через всю расчетную область. В работе [3] предложен способ решения этой проблемы. Идея преодоления этой сложности состоит в следующем: прямой и обратный ход метода Булеева организуется не по всей области, а по подобластям, получившимся в результате декомпозиции расчетной области. При этом в качестве граничных условий используются по аналогии с асинхронным методом Зейделя значения с последней итерации. В [3] показано, что при использовании данного подхода количество итераций возрастает незначительно (10 %).

Результаты тестирования методов решения СЛАУ

Результаты сравнения показали, что для задач большой размерности только метод Зейделя с красно-черным упорядочиванием при распараллеливании целиком повторяет процесс сходимости последовательного алгоритма, что выражается в сохранении количества итераций, требующихся для сходимости метода, вне зависимости от числа использованных процессоров и способа декомпозиции. Важно отметить, что рассматриваемая реализация алгоритма очень хорошо масштабируется на любое разумное количество вычислительных узлов.

Параллельная реализация методов сопряженных и бисопряженных градиентов теоретически также не нарушает процесс сходимости последовательного алгоритма. Но на практике обнаруживается, что количество итераций случайным образом меняется при различных спо-

собах декомпозиции. Численные эксперименты показали, что это является следствием некоммутативности в случае машинной арифметики вычисления суммы большего количества слагаемых. Появляется небольшая погрешность в седьмом или восьмом знаке, что в дальнейшем приводит к изменению хода итерационного процесса. Необходимо отметить, что сам итерационный процесс в конце концов сходится, при этом лишь незначительно меняется получение последовательности приближений к решению.

Параллельный алгоритм метода Булеева для поставленной задачи работает корректно в том случае, если не используется разбиение области по оси Ox и тем самым допускает использование лишь 1D и 2D декомпозиции.

На рис. 1 показано, что метод бисопряженных градиентов с предобуславливанием методом Зейделя с красно-черным упорядочиванием является наиболее эффективным с точки зрения временных затрат и расчет на одном процессоре занимает столько же времени, сколько 64-процессорный вариант любого другого рассматриваемого метода.

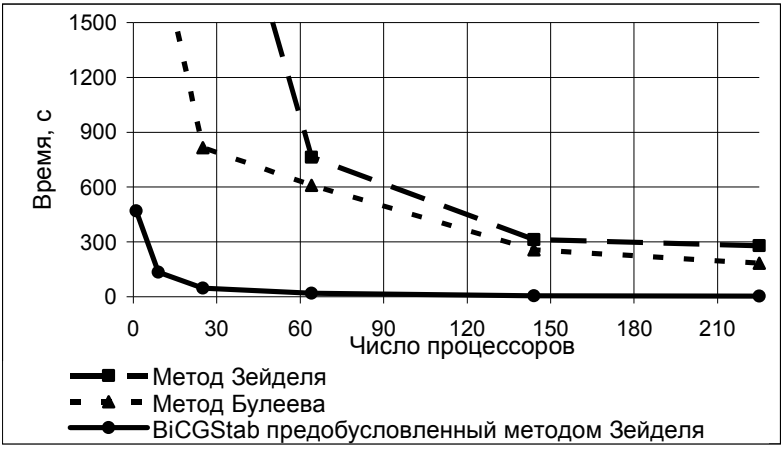


Рис. 1. Время работы параллельной программы в зависимости от используемого количества вычислительных узлов для рассматриваемых методов

Метод Зейделя с красно-черным упорядочиванием и метод Булеева сходятся к решению более равномерно, но требуют значительного количества итераций: 68373 – метод Булеева и 200756 – метод Зейделя, в то время как метод бисопряженных градиентов с предобуславливанием методом Зейделя сходится за 198 итераций.

Также необходимо отметить, что количество итераций для сходимости методов Булеева и бисопряженных градиентов, предобусловленного методом Зейделя, увеличивается на 35 % при увеличении количества вычислительных узлов до 225.

Максимальную эффективность показывают методы Зейделя с красно-черным упорядочиванием и метод бисопряженных градиентов, предобусловленный методом Зейделя, чуть хуже показатели у метода Булеева. Метод бисопряженных градиентов с предобуславливанием по Булееву показывает низкую эффективность вследствие значительного увеличения количества итераций, требующихся для сходимости метода, при росте количества используемых процессоров.

Выводы

На основе полученных результатов можно сделать вывод, что наилучшим вариантом для решения выбранной задачи на многопроцессорной вычислительной машине с распределенной памятью является использование метода бисопряженных градиентов с предобуславливанием методом Зейделя.

Литература

1. **Van der Vorst H.A.** Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetrical linear systems: Preprint 633. Univ.Utrecht., 1990.
2. **Беликов Д.А., Говязов И.В., Данилкин Е.А. и др.** Высокопроизводительные вычисления на кластерах. Томск: Изд-во Том. ун-та, 2008. 198 с.
3. **Богословский Н.Н., Есаулов А.О., Старченко А.В.** Параллельная реализация алгоритма вычислительной гидродинамики SIMPLE // Труды Сибирской школы-семенара по параллельным вычислениям. Томск: Изд-во Том. ун-та, 2002. С. 118-124.

Моделирование очистки воды от механических примесей методом дискретного элемента

Е.Н. Дьяченко
Томский государственный университет

Рассматривается моделирование седиментации и фильтрации механических примесей в жидкости методом дискретного элемента и при помощи супервычислительной техники. Результаты такого моделирования могут быть использованы для оптимизации параметров очистных сооружений и повышения степени очистки воды.

При моделировании процессов очистки воды, седиментации и фильтрации необходимо учитывать движение множества частиц примесей различной формы и размеров в жидкой среде. Суперкомпьютерное моделирование при этом целесообразно проводить методом дискретного элемента, изначально предложенным Кундаллом и Стреком [1] для задач механики горных пород и впоследствии модифицированным в работах [2 – 3] для приложения к задаче осаждения частиц.

Суть метода дискретного элемента заключается в двойном численном интегрировании уравнения движения (второго закона Ньютона) для каждой частицы.

Поскольку необходимо рассчитывать взаимодействие всех частиц со всеми, то вычислительная сложность метода пропорциональна квадрату количества рассматриваемых элементов, что значительно ограничивает либо продолжительность моделируемого физического процесса по времени, либо количество частиц.

Обозначенная задача в точной постановке сложно поддается математической формализации (в частности, возникает множество проблем при описании частиц сложной и разнообразной формы) и является крайне ресурсоемкой, поэтому в модели используются следующие упрощенные представления: не учитывается сложный характер движения жидкости между частицами; полагается, что все частицы имеют сферическую форму; частицы взаимодействуют при контакте друг с другом.

На рис. 1 показана схема пресс-фильтра – аппарата для разделения под давлением жидких неоднородных систем (суспензий) на жидкую фазу – фильтрат и твердую фазу – осадок. Пресс-фильтры применяются для фильтрации широкого класса суспензий, а также они пригодны для разделения суспензий с небольшой концентрацией твердых частиц

и суспензий с повышенной температурой, охлаждение которых недопустимо вследствие выпадения кристаллов из жидкости.

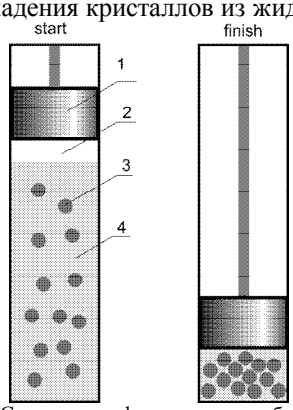


Рис. 1. Схема пресс-фильтра и его работы:
1 – поршень; *2* – воздушный слой;
3 – частицы твердой фазы; *4* – жидкость

Максимальное количество частиц, которое удается просчитать в течение суток на кластерном суперкомпьютере СКИФ Cyberia с пиковой производительностью 12 ТФлопс, – около одного миллиона при продолжительности физического процесса порядка 100 с. Распараллеливание производилось посредством трехмерной динамической пространственной декомпозиции.

На рис. 2 представлена динамика роста слоя – зависимость высоты слоя от времени. Видно, что до момента времени 0,5 с слой растет, потом достигает максимума высоты и далее (для больших значений коэффициента адгезии Kad) происходит медленный процесс усадки слоя. Похожая картина процесса обнаруживается в натурном эксперименте: для сравнения на рис. 3 приводятся данные из работы [4]. Сравнивая рис. 2 и 3, можно обнаружить качественное сходство данных, но при этом наблюдается огромное количественное различие. Это связано с тем, что в численном моделировании не удается просчитать такое большое количество частиц, как в натурном эксперименте. В дальнейшем планируется преодолеть эту проблему посредством введения в рассмотрение безразмерных критериев.

На рис. 4 представлены данные о распределении плотности слоя частиц по высоте. Суспензия вначале седиментировала, а затем слой подвергался воздействию поршня. Момент времени 3 с соответствует окончанию процесса седиментации (поршень еще не коснулся частиц), 4 с – началу процесса сжатия поршнем слоя, 6,8 с – остановке поршня.

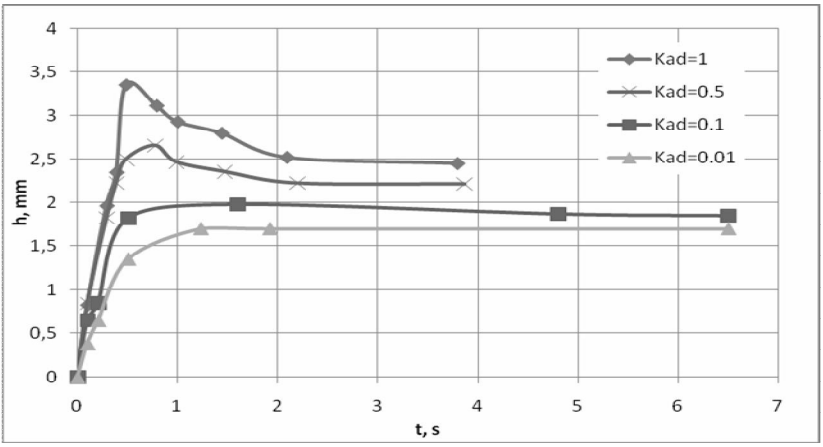


Рис. 2. Динамика роста слоя для различных уровней адгезии частиц (компьютерное моделирование): число частиц – 1910, радиус частиц – 0,05 мм, начальная высота слоя – 10 мм, начальная концентрация – 0,1

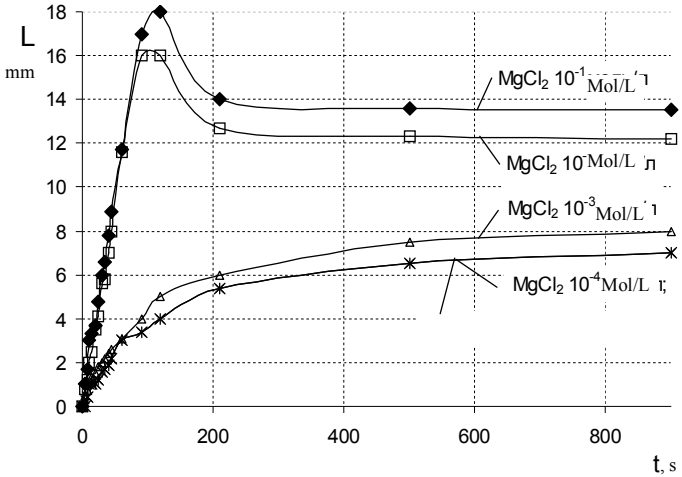


Рис. 3. Динамика роста слоя (натурный эксперимент, взято из [4])

Из рис. 4 видно, что в момент времени 3 с плотность слоя понижается с увеличением высоты. Это связано с тем, что вышележащие частицы давят на частицы внизу и уплотняют их за счет пространственного перераспределения. В момент времени 6,8 с распределение становится практически равномерным, что связано с действием поршня. Стохастические колебания кривых вызваны случайной природой про-

цесса и тем, что в рассмотрении было не так много частиц, как в эксперименте.

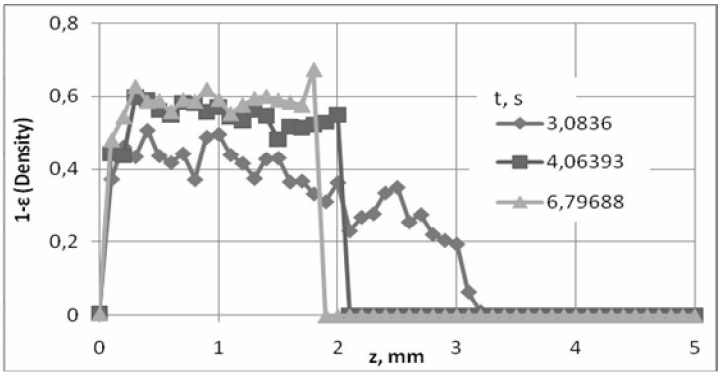


Рис. 4. Распределение плотности слоя по высоте для различных моментов времени

Суперкомпьютерное моделирование находит применение при изучении процессов седиментации и фильтрации частиц механической примеси, позволяет понять физику процессов, дает возможность улучшить степень и скорость очистки воды, увеличивать время работы фильтров без дорогостоящих процедур регенерации и удешевлять строительство очистных сооружений за счет выбора оптимальных размеров и фильтрационных материалов. Но вычислительных ресурсов недостаточно для проведения прямого моделирования, поскольку количество частиц, участвующих в процессах в природе, на много порядков больше, чем позволяют просчитать суперкомпьютеры. Для преодоления в некоторой степени этой проблемы планируется использовать безразмерные критерии.

Литература

1. **Cundall P.A., Strack O.D.L.** A distinct element model for granular assemblies // Geotechnique. 1979. Vol. 29.P. 47-65.
2. **Дик И.Г., Дьяченко Е.Н., Миньков Л.Л.** Моделирование случайной упаковки шаров // Физическая мезомеханика. 2006. Т. 9. №4. С. 63-69.
3. **Neesse T.H., Dueck J., Djatchenko E.** Simulation of filter cake porosity in solid/liquid separation // Powder Technology. 2009. №193. P. 332-336.
4. **Dueck J., Purevjav D., Kilimnik D. Yu.** A contribution to the theory of porosity of fine-grained sediments // Journal of Engineering Physics and Thermophysics. 2004. Vol. 77. № 1. P. 93-102.

Сервис-ориентированная архитектура доступа к ресурсам высокопроизводительных вычислительных систем

И.В. Бойченко, С.В. Корытников, А.В. Лежанкин
Томский университет систем управления и радиоэлектроники

В силу архитектурных особенностей применение ВПВС требует особых моделей программирования, что увеличивает время и стоимость разработки ПО. Организация доступа к ресурсам ВПВС в виде набора информационных сервисов повышает эффективность разработки прикладного ПО и информационных систем за счет повторного использования уже существующих решений и ослабления низкоуровневых связей с программно-аппаратными платформами вычислительных систем.

Высокопроизводительные вычислительные системы (ВПВС) в силу высокой стоимости являются разделяемым ресурсом на крупном предприятии или в научном центре. При этом в силу архитектурных особенностей применение ВПВС требует особой модели программирования. Общепризнано, что программное обеспечение (ПО) для суперкомпьютеров и вычислительных кластеров является гораздо более трудоемким в разработке, чем обычное ПО, рассчитанное на применение одного процессора.

Следует отметить, что не всегда вычислительные задачи требуют применения ВПВС. Необходимость возникает, когда размер (размерность) входных данных превышает некоторый порог, за которым время выполнения задачи на одном процессоре (вычислительном блоке) становится неприемлемо большим.

Мы задались следующими вопросами:

1. Следует ли при необходимости применения ВПВС переносить все созданное ранее программное обеспечение (со всей своей функциональностью) с других платформ, например персональных ЭВМ, на высокопроизводительные?
2. Можно ли отдельные функции (работа с большими массивами данных, решение вычислительно трудоемких задач и др.) реализовать в виде информационных сервисов, доступных по требованию?

Полный перенос всех возможностей с одной платформы на другую – очень трудоемкое решение, и уж тем более не по силам одному разработчику. Но в случае положительного ответа на второй вопрос

можно получить гибкую инфраструктуру, в которой ВПВС предоставляют набор сервисов, доступных по требованию.

Набор сервисов может включать в себя:

1) решение численными методами задач линейной алгебры;
2) обработку цифровых сигналов, в том числе мультимедийных;
3) решение задач дискретной математики и теории графов;
4) индексацию и поиск по сверхбольшим массивам информации;
5) распознавание образов;
6) виртуализацию приложений и др.

Такая организация доступа, получившая название сервис-ориентированной, позволяет снизить стоимость разработки ПО для решения задач с применением ВПВС и, следовательно, повысить коэффициент полезного действия таких дорогостоящих систем, а также обеспечить возможность включения ВПВС в состав регулярно действующих информационных систем, таких, например, как поисковые, предметно-ориентированные системы и т.п.

В настоящее время имеется два отчасти пересекающихся направления развития технологий распределенных вычислений, имеющих некоторые необходимые, но недостаточные, на данный момент, свойства для организации сервис-ориентированной архитектуры доступа к ресурсам ВПВС: «Облачные вычисления» (Cloud computing) [1, 2] и GRID [3, 4]. Нами начаты исследования и эксперименты в этих направлениях. В качестве ВПВС, ресурсы которой предполагается предоставить в виде набора информационных сервисов, используется вычислительный кластер нашего университета (http://cluster.tusur.ru).

Один из экспериментов, предпринятый нами, заключается в том, чтобы взять определенную предметно-ориентированную систему, обладающую возможностями расширения, и создать программную инфраструктуру, позволяющую осуществлять часть расчетов на вычислительном кластере (рис. 1). В качестве прикладной системы был выбран пакет MATLAB. Для этого пакета нами разработано специальное расширение на языке Java, позволяющее обращаться к веб-сервису для решения простой задачи — перемножение двух матриц. Основная задача расширения, работающего на уровне клиента, — передача запроса на решение задачи и ее параметров на сервер приложений (использовался Apache Tomcat), а затем после выполнения расчетов получение результата и возврат в программу. При этом предусмотрено хранение массивов в виде удаленных объектов, так как локальная система может не умещать все параметры в локальной ОЗУ.

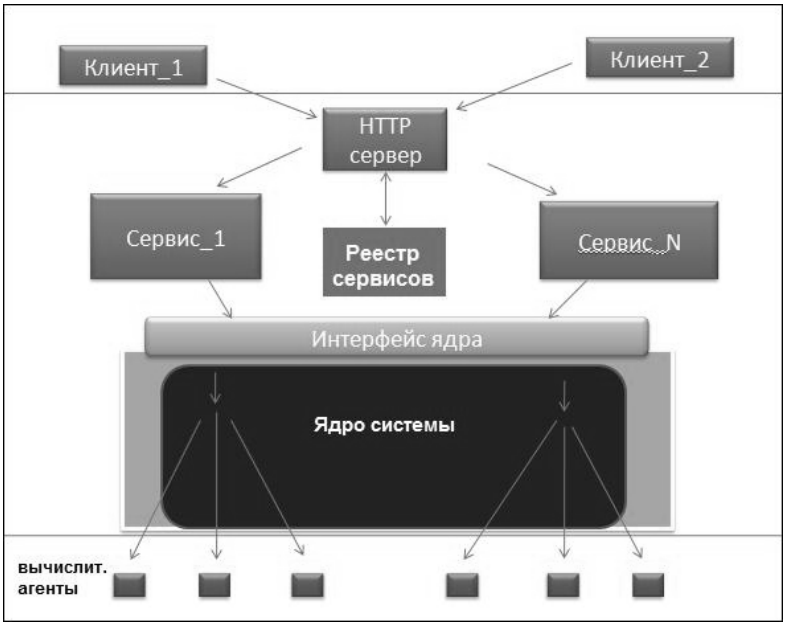


Рис. 1. Сервис-ориентированная архитектура доступа к ресурсам ВПВС

Реестр сервисов обеспечивает хранение и выдачу технических параметров для решения конкретного типа задачи. Решение задачи происходит в ядре системы, которое определяет метод решения задачи, разделяет данные в зависимости от выбранного метода, запускает процесс счета на вычислительных агентах, контролирует выполнение и выдает результат на уровень сервиса, чтобы затем результат был возвращен клиенту.

Предполагается, что задача должна быть типовая и метод решения должен быть полностью описан и зарегистрирован в системе. Мы проводили эксперимент с использованием двух методов решения: с использованием прикладной библиотеки Intel Math Kernel Library (MKL), установленной на вычислительном кластере, и с использованием собственной разработки, основанной на технологии MPI. Это позволяет смоделировать две ситуации: первая, когда в систему интегрируется уже готовое программное обеспечение, и вторая, когда система может быть расширена за счет включения пользовательского прикладного ПО. На данный момент разработанное программное обеспечение находится в стадии испытаний.

Кроме чисто вычислительных задач, существуют и другие прикладные задачи, для которых уже разработаны соответствующие сервисы. Например:

- сервис обработки распределенных баз данных с использованием технологии MapReduce. Реализация этого сервиса осуществлена в проекте Hadoop (http://hadoop.apache.org/);
- сервис доступа к виртуальным средам. Здесь пользователь может собрать как из конструктора свою прикладную систему, используя блоки: операционная система, библиотеки выполнения программ, прикладные библиотеки, серверное ПО и т.д. Собранная таким образом система загружается на один из имеющихся серверов и предоставляется пользователю как реально действующая. Примером реализации является проект Tashi (http://incubator.apache.org/tashi/);
- сервис доступа к физическим средам. В отличие от предыдущего сервиса здесь предполагается, что пользователь может также конфигурировать аппаратное обеспечение. Примером реализации является проект Tycoon.

В будущем мы предполагаем встраивать в нашу сервис-ориентированную архитектуру и такие виды сервисов.

Литература

1. Cloud computing [Электронный ресурс] / Википедия. Свободная энциклопедия. 2009 Режим доступа: http://en.wikipedia.org/wiki/Cloud_computing, свободный
2. Grid computing [Электронный ресурс] / Википедия. Свободная энциклопедия. 2009 Режим доступа: http://en.wikipedia.org/wiki/Grid_computing, свободный
3. Cloud Computing: высокая облачность [Электронный ресурс] / Авт. Андрей Крупин. Электрон. журнал. М. : Компьютерра, 2009. Режим доступа: http://www.computerra.ru/interactive/461761/, свободный
4. Распределённые вычисления, GRID-технологии или кластеры? [Электронный ресурс] / Авт. Леонид Черняк. Электрон. журнал М. : Открытые системы, 2004. Режим доступа: http://www.osp.ru/cw/2004/04/72923, свободный

Параллельная реализация на суперЭВМ модели гравитирующей газопылевой системы

С.Е. Киреев
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Разработан параллельный трехмерный код для моделирования астрофизических объектов. Модель включает пылевую и газовую компоненты, гравитацию и трение между пылью и газом. Для решения уравнения для пыли используется метод частиц-в-ячейках. Уравнения газовой динамики решаются методом крупных частиц. При распараллеливании используется декомпозиция по пространству и по частицам. Представлены оценки эффективности распараллеливания.

Введение

В работе рассматривается параллельный трехмерный код для моделирования астрофизических объектов, таких как галактики и протопланетные диски. Используемая модель включает самогравитирующие газовую и пылевую компоненты, а также трение между ними. Необходимость распараллеливания обусловлена большими объемами обрабатываемых данных и долгим временем вычислений. Сложность параллельной реализации заключается в необходимости совмещать несколько различных физических процессов и использовать различные численные методы для их моделирования.

Описание задачи

Пылевая компонента галактики или протопланетного диска описывается бесстолкновительным уравнением Власова, которое решается методом частиц-в-ячейках [1, 2]. Газовая компонента описывается системой уравнений газовой динамики. Для ее моделирования используется метод крупных частиц (FLIC) [3, 4]. Распределение гравитационного потенциала удовлетворяет уравнению Пуассона, которое решается методом быстрого преобразования Фурье по трем направлениям. Для дискретизации пространства используется равномерная декартова сетка, что дает одинаковые свойства пространства во всей области моделирования.

Параллельная реализация

Основная проблема при параллельной реализации метода частиц-в-ячейках состоит в необходимости хранить модельные частицы и участок сетки из одной области пространства в памяти одного процесса. Есть опасность, что все частицы могут собраться в одной подобласти, что может сильно ограничить масштабирование задачи. Данная проблема была решена с помощью специального способа декомпозиции, совмещающего декомпозицию пространства с распределением частиц каждой подобласти по группе процессоров [5]. Число процессоров в группе для обработки каждой подобласти зависит от числа частиц в этой подобласти. Для начального задания и поддержки в процессе счета оптимального распределения процессоров по группам был реализован алгоритм балансировки.

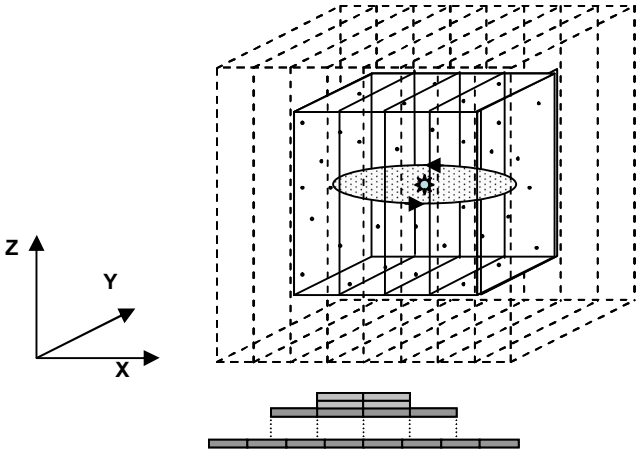


Рис. 1. Декомпозиция большой и малой сеток

Распараллеливание метода крупных частиц для решения уравнений газовой динамики выполнено путем декомпозиции пространства моделирования. Параллельное быстрое преобразование Фурье при решении уравнения Пуассона выполняется с помощью библиотеки FFTW, которая сама задает способ декомпозиции. Таким образом, имеются две вложенные сетки, которые по-разному распределены между процессорами: малая сетка для движения вещества и большая сетка для решения уравнения Пуассона (рис. 1). Поэтому на каждом шаге по времени необходимо дважды делать перераспределение сеточных зна-

чений между процессорами: перераспределение значений плотности вещества перед решением уравнения Пуассона и значений гравитационного потенциала после него.

Оценка эффективности параллельной реализации

Для оценки эффективности созданного параллельного кода была проведена серия тестов на кластере МВС-100К (МСЦ РАН). Параметры тестов представлены в следующей таблице:

N	Компоненты	Размер малой сетки	Размер большой сетки	Число частиц
(1)	Пыль	200×200×200	400×400×400	10^7
(2)	Пыль	300×300×300	600×600×600	10^7
(3)	Пыль	200×200×200	400×400×400	10^8
(4)	Газ	200×200×200	400×400×400	–
(5)	Газ	250×250×250	500×500×500	–
(6)	Пыль + газ	500×500×500	1000×1000×1000	10^9

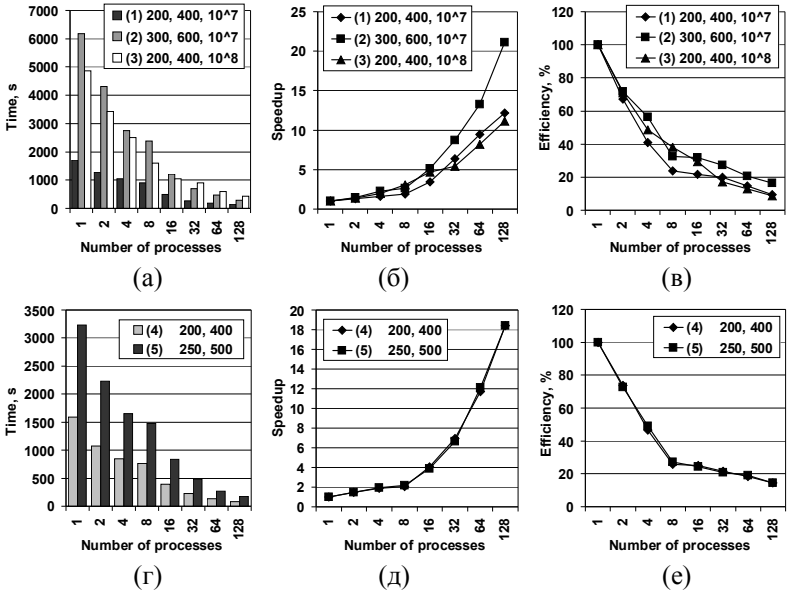


Рис. 2. Время работы (*а,г*), ускорение (*б,д*) и эффективность (*в,е*) расчета пылевой (*а,б,в*) и газовой (*г,д,е*) компонент

На рис. 2 показаны результаты расчетов отдельно пылевой и газовой компонент. Наибольшее падение эффективности наблюдается при использовании до 8 ядер. Дополнительное тестирование показало, что основная причина такого падения эффективности – ограничение на пропускную способность общей памяти 8-ядерного узла кластера.

В представленных результатах число подобластей при декомпозиции области бралось равным числу процессоров. Расчет пылевой компоненты при большом числе частиц можно ускорить, если взять меньшее число подобластей.

На рис. 3 показаны результаты расчета всей комплексной задачи с большим объемом данных на большом числе процессоров. Видно, что уменьшение времени расчета при увеличении числа процессоров имеет место при оптимальном выборе числа подобластей (групп процессоров).

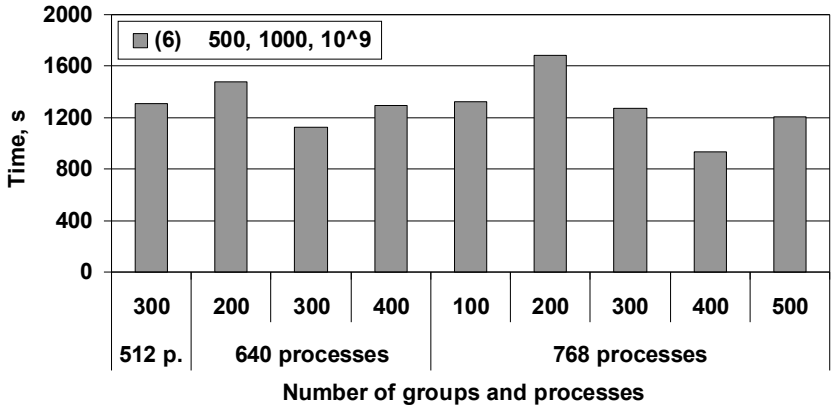


Рис. 3. Время расчета при моделировании газопылевого самогравитирующего облака

Заключение

Реализованный трехмерный код позволяет выполнять расчеты на сетках порядка 500×500×500 ячеек и использовать 1 млрд модельных частиц. При увеличении числа используемых процессоров имеет место ускорение счета.

Литература

1. **Хокни Р., Иствуд Дж.** Численное моделирование методом частиц. М.:Мир, 1987.
2. **Григорьев Ю.Н., Вшивков В.А., Федорук М.П.** Численное моделирование методами частиц-в-ячейках. Новосибирск: Изд-во СО РАН, 2004.
3. **Белоцерковский О.М., Давыдов Ю.М.** Метод крупных частиц в газовой динамике. М.: Наука, 1982.
4. **Вшивков В.А., Лазарева Г.Г., Киреев С.Е., Куликов И.М.** Параллельная реализация модели газовой компоненты самогравитирующего протопланетного диска на суперЭВМ // Вычислительные технологии. 2007. Т. 12, №3. С. 38-52.
5. **Киреев С.Е.** Параллельная реализация метода частиц в ячейках для моделирования задач гравитационной космодинамики // Автометрия. 2006. №3. С. 32-39.

Экспериментальный сравнительный анализ МКЭ и МКО для моделирования трехмерных гармонических электромагнитных полей*

С.Д. Бутюгин, А.В. Петухов
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Экспериментально исследуется и сравнивается эффективность различных численных аппроксимаций (методами конечных объемов (МКО) и конечных элементов (МКЭ)) для решения трехмерной краевой задачи электромагнетизма в частотной области. Полученные вещественные несимметричные и комплексные симметричные знаконеопределенные СЛАУ решались различными предобусловленными итерационными методами в подпространствах Крылова. Представлены результаты экспериментов для представительного набора модельных задач, демонстрирующие работу предложенных алгоритмов.

Постановка задачи

Система уравнений Максвелла для электромагнитного поля $\vec{\mathbf{E}} = \vec{\mathbf{E}}(\vec{\mathbf{r}})e^{i\omega t}$, $\vec{\mathbf{H}} = \vec{\mathbf{H}}(\vec{\mathbf{r}})e^{i\omega t}$ с гармонической зависимостью от времени может быть преобразована к уравнению Гельмгольца [1, 2]

$$\nabla \times \mu_r^{-1} \nabla \times \vec{\mathbf{E}} - k_0^2 \dot{\varepsilon}_r \vec{\mathbf{E}} = 0, \qquad (1)$$

при $\vec{J} = 0$ и $\rho = 0$, где $\dot{\varepsilon} = \varepsilon_r \varepsilon_0 - i\sigma^e/\omega$, $k_0 = \omega\sqrt{\varepsilon_0 \mu_0}$. Решение ищется в трехмерной области Ω с границей $\Gamma = S_0 \bigcup S_1$. Задаются граничные условия на $S_0 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = \vec{\mathbf{n}} \times \vec{\mathbf{E}}_0$ – условие волнового входа и на $S_1 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = 0$ – условие металлической стенки. Здесь и далее под $\vec{\mathbf{n}}$ понимается вектор внешней нормали относительно грани. Условия согласования на внутренних границах описываются следующими уравнениями:

$$\vec{\mathbf{n}} \cdot \left(\dot{\varepsilon}_1 \vec{\mathbf{E}}_1 - \dot{\varepsilon}_2 \vec{\mathbf{E}}_2\right) = 0, \quad \vec{\mathbf{n}} \times \left(\vec{\mathbf{E}}_1 - \vec{\mathbf{E}}_2\right) = 0,$$
$$\vec{\mathbf{n}} \cdot \left(\dot{\mu}_1 \vec{\mathbf{H}}_1 - \dot{\mu}_2 \vec{\mathbf{H}}_2\right) = 0, \quad \vec{\mathbf{n}} \times \left(\vec{\mathbf{H}}_1 - \vec{\mathbf{H}}_2\right) = 0.$$

Дополнительно рассматривается задача для потенциалов

* Работа выполнена при поддержке РФФИ (грант 08-01-00526) и ОМН РАН (проект 1.3.4).

$\vec{\mathbf{E}} = \vec{\mathbf{A}} + \nabla V$, $\vec{\mathbf{B}} = (i/\omega)\nabla \times \vec{\mathbf{A}}$ с краевым условием для V: $\Gamma : \nabla V = 0$, для $\vec{\mathbf{A}}$ берутся те же условия, что и для $\vec{\mathbf{E}}$.

Аппроксимация

Для аппроксимации МКО дополним граничные условия равенством $div\vec{\mathbf{E}} = 0$, уравнение (1) в подобластях с постоянными материальными параметрами приведем к виду

$$-\Delta\vec{\mathbf{E}} + \kappa\vec{\mathbf{E}} = 0, \quad \kappa = -\mu_r\dot{\varepsilon}_r k_0^2,$$

которое выводится для гладкого $\vec{\mathbf{E}}$ на основе операторного равенства $\nabla \times \nabla \times = -\Delta + graddiv$.

Более подробно о построении аппроксимации МКО описано в [3, 4], в которой более подробно описан барицентрический МКО в применении к уравнению Гельмгольца.

В случае аппроксимации МКЭ комплексное уравнение Гельмгольца (1) может быть переписано в вариационной постановке [5]: найти $\vec{E} \in H_\Gamma^{\mathrm{rot}}$, для которого выполнено

$$\int_\Omega \left[\mu_r^{-1}\left(\nabla \times \vec{\mathbf{E}}\right)\cdot\left(\nabla \times \vec{\psi}\right) - k_0^2\dot{\varepsilon}_r\left(\vec{\mathbf{E}}\cdot\vec{\psi}\right)\right]d\Omega = 0, \quad \forall\vec{\psi} \in H_0^{rot},$$

дискретный аналог которой в случае задачи для потенциалов выглядит следующим образом [5]:

$$\sum_i u_i^A \int_\Omega \left[\mu_r^{-1}\left(\nabla \times \vec{\psi}_i\right)\cdot\left(\nabla \times \vec{\psi}_j\right) - k_0^2\dot{\varepsilon}_r\left(\vec{\psi}_i\cdot\vec{\psi}_j\right)\right]d\Omega - \sum_i u_i^V k_0^2 \int_\Omega \dot{\varepsilon}_r\left(\nabla\varphi_i\cdot\vec{\psi}_j\right)d\Omega = 0,$$

$$-\sum_i u_i^A k_0^2 \int_\Omega \dot{\varepsilon}_r\left(\vec{\psi}_i\cdot\nabla\varphi_j\right)d\Omega - \sum_i u_i^V k_0^2 \int_\Omega \dot{\varepsilon}_r\left(\nabla\varphi_i\cdot\nabla\varphi_j\right)d\Omega = 0,$$

где φ_j – узловые базисные функции; ψ_j – реберные базисные функции тетраэдра первого порядка. Данные уравнения записываются для функций ψ_j и φ_j, равных 0 на Г, для остальных берутся уравнения

$$u_j^A = \left(\vec{\mathbf{E}}_0(\vec{x}_j)\cdot\vec{\tau}_j\right)\Big/\left(\vec{\psi}_j(\vec{x}_j)\cdot\vec{\tau}_j\right), \quad u_j^V = 0.$$

В обоих методах матрица системы уравнений может быть собрана на основе поэлементной технологии [3, 6].

Численные методы

Полученные системы уравнений решались при помощи широко известных методов сопряженных и бисопряженных направлений подпространств Крылова (CG, BiCG, BiCGStab), описанных в [7, 8] и предобусловленных методом неполной факторизации Айзенштата.

Тестовые задачи

В качестве тестовой задачи рассматривался волновод с линейными размерами: a = 72, b = 34, c =200. На грани z = 200 задавалось условие $S_0 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = \vec{\mathbf{n}} \times \vec{\mathbf{E}}_0$, на остальной части границы расчетной области задавалось условие $S_1 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = 0$.

Во всех тестах полагается $\mu_r = 1$, $\vec{\mathbf{E}}_0 = \vec{\mathbf{e}}_y \sin(\pi x/a), \omega = 2\pi \cdot 3 \cdot 10^9$.

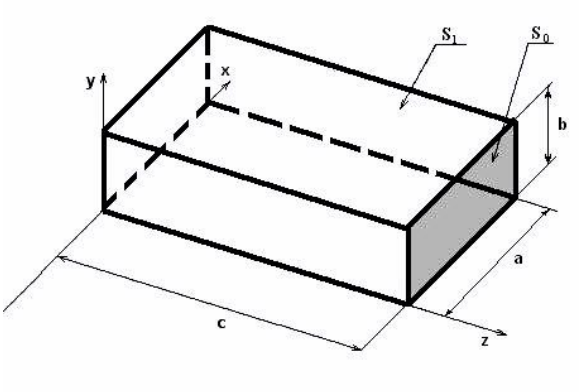


Рис. 1. Расчетная область для тестовых задач

Тест 1: $\varepsilon_r = 1$.

Тест 2: $\varepsilon_r = 2 - 0{,}1i$.

Тест 3: $\begin{cases} \varepsilon_r = 5, & 0 \le z \le c/2, \\ \varepsilon_r = 1, & c/2 \le z \le c. \end{cases}$

Расчетная область делилась на параллелепипеды, каждый из которых, в свою очередь, разбивался на 6 тетраэдров. Использовались следующие разбиения: 8×4×20, 14×6×40, 28×14×80, 58×28×160 по оси *x*, *y* и *z* соответственно.

Таблица 1. Тест №1

№	МКО			МКЭ		
	δE	n	t''	δE	n	t''
1	$1{,}5 \cdot 10^{0}$	19	<<1	$1{,}3 \cdot 10^{-1}$	57	0,5
2	$3{,}0 \cdot 10^{-1}$	28	0,1	$8{,}1 \cdot 10^{-2}$	110	11
3	$7{,}0 \cdot 10^{-2}$	45	2	$3{,}2 \cdot 10^{-2}$	217	220
4	$1{,}7 \cdot 10^{-2}$	84	24	$1{,}6 \cdot 10^{-2}$	558	7800

Таблица 2. Тест №2

№	МКО			МКЭ		
	δE		t''	δE	n	t''
1	$1{,}7\cdot10^{-1}$	32	<<1	$1{,}4\cdot10^{-1}$	110	1
2	$4{,}7\cdot10^{-2}$	47	0,2	$7{,}8\cdot10^{-2}$	221	21
3	$1{,}2\cdot10^{-2}$	98	3,3	$3{,}5\cdot10^{-2}$	896	910
4	$2{,}8\cdot10^{-3}$	172	48	$1{,}7\cdot10^{-2}$	2203	13700

Таблица 3. Тест №3

№	МКО			МКЭ		
	δE	N	t''	δE	n	t''
1	$1{,}5\cdot10^{-0}$	95	<<1	$3{,}6\cdot10^{-1}$	245	0,8
2	$2{,}7\cdot10^{-1}$	125	0,3	$1{,}7\cdot10^{-1}$	509	19
3	$5{,}1\cdot10^{-2}$	207	7,1	$5{,}2\cdot10^{-2}$	1342	560
4	$1{,}2\cdot10^{-2}$	278	78	$2{,}4\cdot10^{-2}$	3234	12000

Критерием останова итерационного процесса служило условие $\left\|\vec{\mathbf{f}} - \mathbf{A}\vec{\mathbf{u}}_n\right\| < \varepsilon\left\|\vec{\mathbf{f}}\right\|$, $\varepsilon = 10^{-7}$. В табл. 1–3 приведена относительная ошибка δE полученных численных решений с аналитическими решениями, количество итераций n и время работы t в секундах для метода BiCGStab.

Выводы

Результаты тестов, представленные в таблицах, показывают сходимость методов на последовательности сгущающихся сеток с первым порядком для МКЭ и вторым порядком для МКО. Из табл. 1–3 видно, что МКО требует меньшее число итераций и меньше времени, что связано с лучшей обусловленностью матрицы системы.

Литература

1. **Григорьев А.Д., Янкевич В.Б.** Резонаторы и резонаторные замедляющие системы СВЧ. М.: Радио и связь, 1984.
2. **Bossavit A.** Computational Elactromagnetism. N.Y.: Academ Press, 1987.
3. **Ильин В.П.** Метод конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск: Изд-во ИВМиМГ СО РАН, 2001.

4. **Петухов А.В.** Решение трехмерного комплексного уравнения Гельмгольца методом барицентрических конечных объемов // Автометрия. 2007. Т. 43, №2. С. 112-123.
5. **Dyczij-Edlinger R., Biro O.** A joint and scalar potential formulation for driven high frequency problems using hybrid edge and nodal finite elements // IEEE Trans. on Microwave Theory and Techniques. 1996. Vol. 44, № 1 P. 15-23.
6. **Соловейчик Ю. Г., Рояк М.Э., Персова М.Г.** Метод конечных элементов для решения скалярных и векторных задач. Новосибирск: Изд-во НГТУ, 2007.
7. **Ильин В.П.** Методы и технологии конечных элементов. Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.
8. **Saad Y.** Iterative Method for Sparse Linear Systems. N.Y.:PWS Pubblish., 1996.

MPI-IO: Практические аспекты разработки масштабируемых приложений*

Б.Л. Петушкеев
Томский государственный университет

Производительность и доступность вычислительных кластеров в России растет быстрыми темпами и позволяет решать задачи, ранее не доступные. Однако этот качественный переход требует изменения подхода к программированию по причине возрастающих объемов данных. MPI-2 предоставляет средства высокоэффективного параллельного ввода/вывода. В работе рассматривается реализация модели вентилирования в выработанных пространствах шахт с учетом такой возможности.

Пример стандартного подхода (назовем его «обычным») при написании программ, использующих MPI как средство передачи сообщений, предполагает ассоциирование некоторого участка данных в глобальной области решения с определенным процессом. Последний выполняет расчет локального массива, а затем соседние процессы обмениваются «затененными» данными. Особенностью и узким местом «обычного» подхода является выделение памяти для всей глобальной области каждым процессом. С одной стороны, это позволяет упростить написание всей программы в рамках SPMD модели, поскольку каждый процесс использует глобальную индексацию и сшивка всей области не представляет затруднений, с другой – существенно ограничивает возможности исследователя по наращиванию количества задействованных в расчете процессоров (ядер) числом не более нескольких сотен. Типичен пример программы, написанной «обычным» способом и запущенной на, например, кластере СКИФ Cyberia МВЦ ТГУ, но достигающей насыщения уже при 250–300 процессах и показывающей практически останов в росте производительности из-за чрезмерно малой гранулярности расчетов. Следует отметить, что на данной машине на 1 ядро приходится всего 1GB оперативной памяти, что при обычной практике запуска MPI-процесса на каждом ядре ведет в итоге к существенному ограничению размерности решаемой задачи.

* Работа выполнена при финансовой поддержке РФФИ № 08-08-12029-офи.

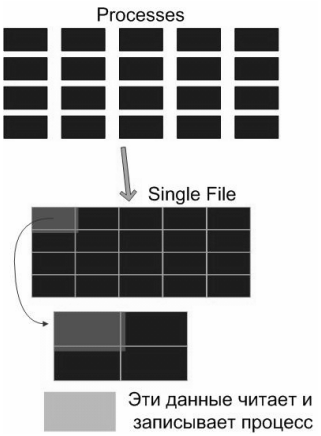


Рис. 1. Обмен данными

Для обхода подобных узких мест в работе предлагается избавиться от глобального массива, необходимого в «обычном» подходе для сборки решения в целях изучения каких-либо интегральных характеристик и, наконец, вывода результатов расчета в файл. Вместо этого предполагается, что на всем протяжении расчета никто из процессов не «видит» полностью всю картину решения, а вывод производится напрямую каждым процессом (рис. 1). При этом если производить запись каждым процессом в свой уникальный файл, то очень скоро встанет вопрос управления и обслуживания огромного количества, что само по себе представляет большую отдельную задачу.

Удачным примером может служить программа FLASH, генерирующая при запуске на 32000 узлах суперкомпьютера BG/L более 72 млн файлов. Вместе с тем возникнут неизбежные проблемы при попытке рестарта программы с контрольной точки на отличном от изначального количестве процессоров.

К параллельной стратегии оперирования с файлами приводят следующие основные доводы:

- последовательный I/O в больших задачах становится узким местом;
- для работы с глобальным файлом (слияние, извлечение и т.д.) требуется трудоемкий пост- и препроцессинг.

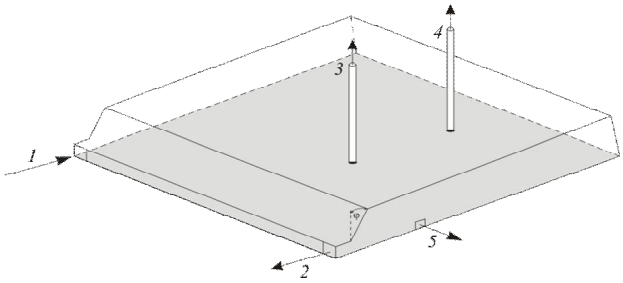


Рис. 2. Область исследования

Предложенный подход к реализации параллельных программ был применен к задаче моделирования процесса фильтрации метано-воздушной смеси через пористую среду шахтных выработок.

Пример физической области расчетной задачи показан на рис. 2, где введены обозначения: *1* – сечение конвейерного штрека; *2* – сечение вентиляционного штрека; *3* и *4* – газоотсасывающие скважины; *5* – открытая сбойка на газодренажный штрек.

Особенностью рассматриваемой проблемы являются чрезвычайно низкие скорости движения газа, составляющие порядка нескольких см/с . С учетом размеров выработанных пространств действующих шахт задача не представляется разрешимой в разумные сроки на обычном ПК. К тому же вкупе с применением суперкомпьютера применяется процедура ускорения, предложенная в [1], что позволяет получать итоговые результаты в очень короткое время.

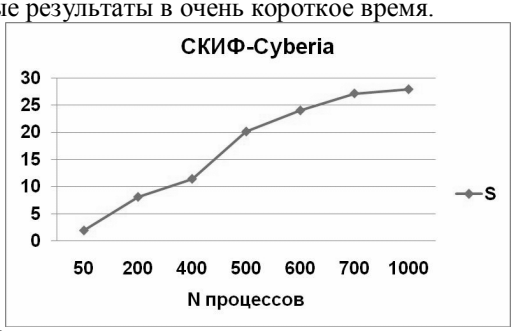


Рис. 3. Зависимость ускорения от числа используемых процессоров

На рис. 3 представлены результаты экспериментов на ускорение итоговой программы при числе ячеек около 24 млн. Объем требуемой оперативной памяти превышает 18 Гб, поэтому величины ускорения отнесены ко времени расчета на 25 процессорах (ядрах). Отметим

практически линейный характер кривой ускорения с наметившейся просадкой в районе 1000 задействованных ядер. Это закономерный результат, поскольку такое количество точек уже недостаточно для загрузки процессоров и на показатели сильно влияют накладные расходы в виде задержек при передаче сообщений.

Также проведены эксперименты на операциях чтения/записи трехмерного массива вещественных чисел double в/из файла общим размером 4Гб при различных режимах работы MPI-IO. Массив на сетку процессов распределялся блочно в форме (block, block, block), часто используемой в научных приложениях. Целевая платформа кластера СКИФ-Cyberia использует параллельное хранилище ReadyStorage ActiveScale Cluster, основанное на файловой системе PanFS, с заявленной производительностью на уровне 700 Мб/с.

На рис. 4 представлены результаты производительности файловой системы на операциях чтения/записи в зависимости как от уровня вызовов MPI-IO, так и от количества задействованных процессов. Пропускная способность рассчитывалась как объем переданных данных, отнесенный к максимальному времени чтения/записи любого из процессов. На обеих диаграммах не приведены результаты, соответствующие level-0 и level-1 вызовам MPI-IO, показавшие одинаково плохие ре-

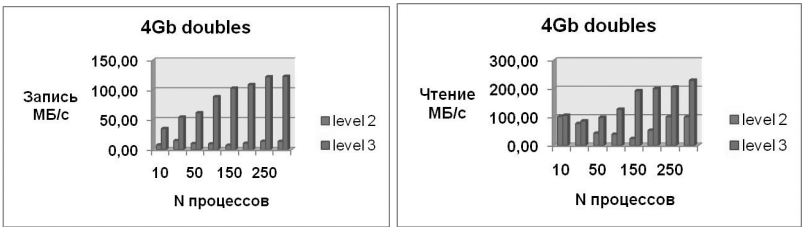


Рис. 4. Производительность файловой системы

зультаты при выбранной политике доступа к файлу. Level-3 демонстрирует стабильный рост показателей при увеличении количества процессов, что указывает на хорошую масштабируемость работы файлового хранилища. Хорошо заметна зона прогиба на операции чтения в районе 10–100 процессов, причем такая же картина наблюдается и при меньшем объеме файла (1 Гб). Объяснение этому, полагаем, следует искать в работе механизма data-sieving, точнее в условиях, при которых он себя эффективно проявляет. Level-2 показывает низкие результаты при записи, связанные с блокировкой файла, и, как следствие, постоянные конфликты за ресурс. Интересно ведут себя level-2 запросы при чтении файла: падение до очень низких показателей, а затем быстрый рост. Level-2 генерирует дополнительные обращения к дис-

кам из-за отсутствия координации, а также наличия больших расстояний между нужными данными (о поиске компромисса в работе data sieving пишут сами авторы ROMIO). При увеличении же количества процессов сверх 200 блоки данных в файле становятся небольшими и эффективность механизма data sieving улучшается, возможно, из-за попадания в буфер ROMIO (по умолчанию 4 Мб) сразу нескольких нужных блоков.

Предлагаемый подход показал свою применимость и актуальность в свете появления в России кластера, достигшего полупетафлопной производительности. Показатели производительности параллельной ФС ожидались лучше, но все же они гораздо выше последовательного I/O (следует иметь в виду, что файл в итоге единый!).

Литература

1. **Васенин И.М., Петушкеев Б.Л.** Задача об ускорении решения совместных уравнений газовой динамики и переноса // Известия вузов. Физика. 2007. №9/2. С. 274-281.

Параллельная реализация гексагональной клеточно-автоматной модели волнового процесса

А.В. Логинова
Новосибирский государственный технический университет

Предложена и проанализирована параллельная реализация КА-алгоритма моделирования волнового процесса с использованием движущейся окрестности.

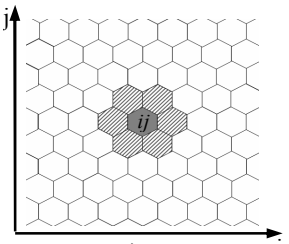
Введение

В [1] доказано, что FHPrp-модель (FHP аббревиатура фамилий авторов, rp – rest particles) [2] описывает волновое уравнение. В модели волновой процесс представляется множеством гипотетических частиц, которые под воздействием внешнего источника движутся и взаимодействуют друг с другом в дискретном пространстве. Для того чтобы наблюдать волновой процесс в привычном представлении, вычисляются осредненные значения плотности частиц в каждой клетке клеточного автомата (КА) по некоторой области осреднения (окрестности). Известно, что осреднение является самой трудоемкой операцией в КА-моделировании. Однако уменьшение радиуса окрестности приводит к увеличению автоматного шума.

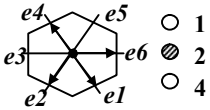
FHPrp-модель

FHPrp-модель построена на двумерной гексагональной решетке (рис.1, *а)*. Каждая клетка имеет 6 соседей и содержит движущиеся частицы и частицы покоя (рис. 1. *б*). Движущиеся частицы имеют единичную массу и единичную скорость. В заданный момент времени в каждом направлении может двигаться только одна частица. Частицы покоя имеют нулевую скорость и массу, отличную от нуля. Частицы, находящиеся в клетке с именем (i,*j*) в заданный момент времени *t*, определяют состояние клетки $\vec{a}_{i,j}^{t}$. На рис. 1. *б* показана клетка с четырьмя движущимися частицами и одной частицей покоя с массой 2.

Каждой клетке ставится в соответствие конечный автомат. Один такт работы автомата состоит на двух этапов: сдвиг и столкновение.



а)



$\vec{a}_{i,j}^{t}$ = (**0**, **1**, **0**, 1, 0, 1, 0, 1, 1),

б)

Рис. 1. FHPгр-модель: *а* – гексагональная решётка; *б* – FHP3гр клетка

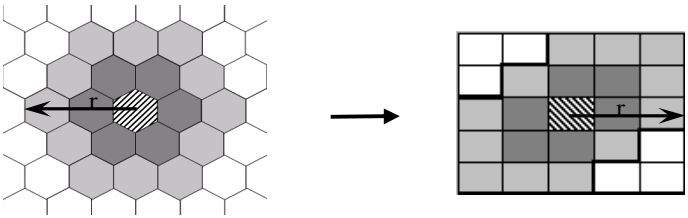
Во время сдвига все частицы сдвигаются к своим соседям. Набор правил столкновения выбирается таким образом, чтобы при каждом столкновении суммарные масса и импульс всех частиц клеточного массива оставались неизменными. Все клетки КА-модели вычисляют новое состояние синхронно и параллельно, в результате чего происходит изменение глобального состояния автомата. Итеративная смена глобальных состояний описывает динамику волнового процесса.

Моделирование КА-волны

Для визуализации волнового процесса через T тактов для каждой клетки массива вычисляются осредненные значения плотности по некоторой области осреднения с радиусом r (рис. 2, *а*) по формуле

$$\langle \rho_{ij} \rangle^{t} = \frac{1}{|\, Av(ij)\,|} \sum_{i,j \in Av(ij)} \rho_{ij}^{t},$$

где $\rho_{i,j}$ – плотность клетки *(i,j)* в момент времени t. Область осреднения в оперативной памяти компьютера представляется в виде фигуры на рис. 2, *б*.



а) b)

Рис. 2. Представление гексагональной решётки: *а* – в модели; *б* – в оперативной памяти компьютера

На рис. 3 показаны времена, которые тратятся на собственно моделирование (100 итераций) и на одно осреднение для КА размера 10000×10000 клеток при разных радиусах осреднения r.

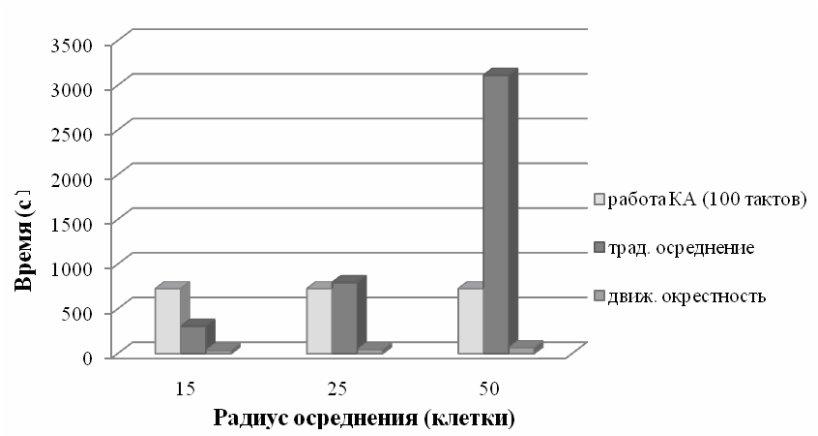
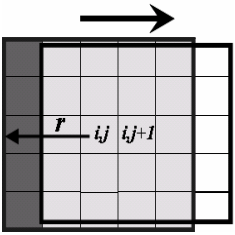


Рис. 3. Сравнение времени моделирования (100 итераций) со временем одного осреднения

Для уменьшения времени осреднения будем использовать движущуюся окрестность. Для этого переходим к квадратной области осреднения размера $(2r+1)\times(2r+1)$ клеток. Идея вычисления осреднённого значения плотности клетки с именем (i,j) с помощью движущейся окрестности проиллюстрирована на рис. 4. Осреднённое значение клетки с именем $(i,j+1)$ равно значению плотности клетки с именем (i,j) минус сумма плотностей первого столбца из окрестности клетки (i,j) и плюс сумма плотностей последнего столбца из окрестности клетки $(i,j+1)$. Аналогично для клетки с именем $(i+1,j)$ только вычитаем и прибавляем соответствующие строки.



$$\langle \rho_{i,j+1} \rangle^t = \frac{1}{|Av(i,j+1)|}\left(\sum_{i,j \in Av(ij)} \rho_{ij}^t + \sum_{k=i-r}^{i+r} \left(\rho_{k,(j+1)+r}^t - \rho_{k,j-r}^t \right) \right)$$

Рис. 4. Иллюстрация идеи движущейся окрестности

Уменьшение времени вычисления осреднённых значений плотности клеток массива с помощью движущейся окрестности (см. рис. 3)

связано с тем, что большая часть вычислений при определении осреднённого значения для близлежащей клетки не выполняется. Более того, моделирование показало, что от увеличения радиуса осреднения время осреднения практически не зависит.

Параллельная реализация

Для получения минимального времени моделирования волнового процесса клеточный массив между процессорами кластера должен быть разделён таким образом, чтобы время осреднения было меньше времени работы КА в течение T тактов на одном процессоре. Для выполнения этого условия на кластере с SMP-архитектурой* при моделировании волнового процесса КА размера 20000×20000 клеток и радиусом $r = 50$ под осреднение было выделено одно ядро, $T = 100$. На рис. 5 показана зависимость времени моделирования от количества ядер.

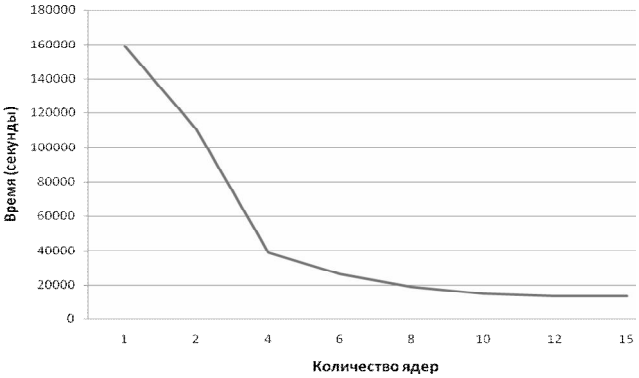


Рис. 5. Зависимость времени работы КА от количества ядер

Заключение

При моделировании волнового процесса линейное ускорение было достигнуто не только за счёт параллельного выполнения, но и за счёт использования нетрадиционного способа осреднения.

* Четыре процессора по четыре ядра (ИВМ и МГ СО РАН).

Литература

1. **Zhang M.** Computing Electromagnetic Fields in Inhomogeneous Media Using Lattice Gas Automata / M. Zhang, D. Cule, L. Shafai // Proceedings of 1998 Symposium on Antenna Technology and Applied Electromagnetics. Ottawa, Canada, 1998. №11. P. 14-16.
2. **Frish U.** Lattice-Gas hydrodynamics in two and three dimensions / U. Frish, D. d'Humieres, B. Hasslacher // Complex Systems. 1987. №1. P. 649-707.

Моделирование на вычислительном кластере динамики тонкого трубчатого замагниченного моноэнергетического пучка электронов

А.А. Трунов, А.В. Старченко, И.Ю. Турчановский, В.А. Шкляев
Томский научный центр

Рассматривается задача о транспортировке пучка заряженных частиц в двумерной осесимметричной постановке. Решение проводится методом «частицы-в-ячейках» для самосогласованного электромагнитного поля. Проведено распараллеливание лагранжевого этапа в методе. Предложена динамическая балансировка загрузки, существенно использующая свойства задачи и обеспечивающая малый дисбаланс при низких затратах процессорного времени.

Введение

Исследование релятивистских электронных пучков является актуальной задачей в связи с постоянно возникающими новыми областями их прикладного использования, а также по причине существования целого ряда нерешённых задач теории пучков заряженных частиц. Аналитическое исследование их динамики возможно только в редких случаях из-за сложности описывающих их уравнений. Как следствие этого, основную роль играет численное моделирование.

Постановка задачи

Рассматривается осесимметричная задача о транспортировке тонкого трубчатого замагниченного моноэнергетического пучка электронов в эквипотенциальном цилиндрическом канале дрейфа (рис. 1). Все заряженные частицы обладают одинаковым удельным зарядом. Плотность тока инжекции распределена равномерно по области инжекции. Область инжекции расположена на одном из торцевых электродов, её площадь много меньше площади электрода, с которого осуществляется инжекция.

Рассматриваемый процесс описывается с помощью уравнений движения частиц под действием собственных электрических и магнитных полей, а также сильного продольного внешнего магнитного поля. Ввиду осевой симметрии исследуемой системы используется цилиндрическая система координат.

Собственные поля, создаваемые частицами, находятся из решения уравнений Максвелла (1), (2) по известным значениям координат и скорости частиц.

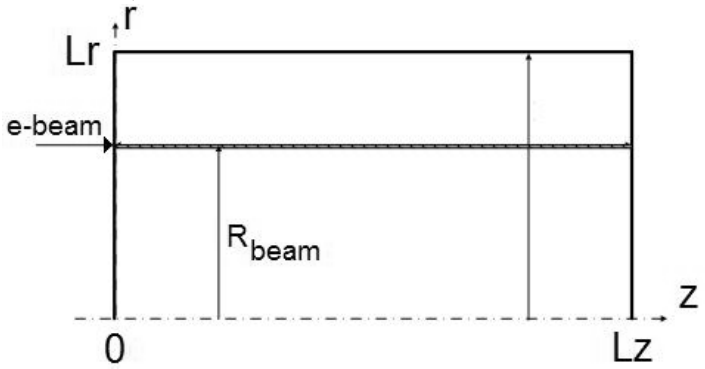


Рис. 1. Геометрия задачи

Lr = 1 см; Lz = 10 см; R_{beam} = 0,67 см; ширина пучка 0,1 мм,

$$\frac{1}{c}\frac{\partial \bar{B}^i}{\partial t} = -rot\ \bar{E}^i,\ \ \frac{1}{c}\frac{\partial \bar{E}^i}{\partial t} = rot\ \bar{B}^i - \frac{4\pi}{c}\bar{J}; \qquad (1)$$

$$div\ \bar{B}^i = 0;\ div\ \bar{E}^i = 4\pi\rho, \qquad (2)$$

где $\bar{B}^i = \left(B_r^i, B_\varphi^i, B_z^i\right)$ – вектор собственного магнитного поля;

$\bar{E}^i = \left(E_r^i, E_\varphi^i, E_z^i\right)$ – вектор собственного электрического поля;

$\bar{J} = (J_r, J_\varphi, J_z)$ – вектор плотности тока; c – скорость света;

ρ – плотность заряда.

Начальные условия для векторов собственного электромагнитного поля: $E_z^i = E_r^i = E_\varphi^i = B_z^i = B_r^i = B_\varphi^i = 0.$ Внешнее продольное магнитное поле: $B_z^e = 100T;\ B_r^e = B_\varphi^e = 0.$ На границах области исследования (кроме оси симметрии) ставятся граничные условия для векторов электромагнитного поля на идеальном металле [2]. На оси симметрии используются условия симметрии [2].

Движение частиц описывается уравнениями Ньютона–Лоренца:

$$\frac{d\bar{u}_p}{dt} = \frac{q}{m}\left(\bar{E} + \frac{1}{c}\left[\frac{\bar{u}_p}{\gamma} \times \bar{B}\right]\right),\ \ \frac{d\bar{x}_p}{dt} = \frac{\bar{u}_p}{\gamma}, \qquad (3)$$

где $\bar{u}_p$ – релятивистская скорость частицы; $\bar{u}_p = \bar{v}_p \cdot \gamma;\ \bar{v}_p = \left(v_{zp}, v_{rp}, v_{\varphi p}\right)$ – скорость частицы; q – заряд частицы; m – масса покоя частицы; $\bar{E} = \bar{E}^i$ – электрическое поле, действующее на частицу; γ – релятивист-

ский коэффициент; $\overline{B} = \overline{B}^i + \overline{B}^e$ – магнитное поле, действующее на частицу; $x_p=(r_p, z_p)$ – радиус-вектор частицы (φ_p не учитывается ввиду осевой симметрии задачи).

Для решения уравнений (3) имеются начальные условия:

$$t = 0 : u_{zp} \approx 5{,}2 \cdot 10^{10}\ см/с,\ u_{rp} = 0,\ u_{\varphi p} = 0;\ 0 \le r_p \le r_1,\ 0 \le z_p \le z_1.$$

Частицы распределяются по псевдослучайному равномерному закону на небольшом расстоянии от инжектирующего электрода. В случае если частица достигает любой границы расчётной области, кроме оси симметрии, исследование её движения прекращается. При переходе через ось симметрии знаки радиальной координаты r_p и радиальной компоненты скорости u_{rp} частицы изменяются.

Вычислительные технологии

Для решения данной задачи применяется метод «частицы-в-ячейках», относящийся к классу эйлерово-лагранжевых методов. Они характерны тем, что эволюция системы частиц на каждом временном шаге разбивается на два этапа. На одном из них (эйлеровом этапе) при фиксированном положении частиц предварительно вычисляется результат их взаимодействия и (или) их коллективного воздействия на среду. Расчёт ведётся на неподвижной (эйлеровой) сетке. На другом, лагранжевом, этапе выполняется интегрирование на очередном временном шаге динамической системы, правая часть которой (поля) вычислена на эйлеровом этапе [1]. Для решения уравнений Максвелла используется явная разностная схема, имеющая второй порядок аппроксимации по времени и пространству и обеспечивающая точное выполнение граничных условий для векторов электромагнитного поля на идеальном проводнике [2]. Ввиду осевой симметрии используется двумерная пространственная сетка (r–z-координаты). Для интегрирования уравнений движения частиц применяется алгоритм Бориса с перешагиванием для осесимметричной задачи, имеющий второй порядок аппроксимации и сохраняющий импульсы частиц [1]. Вычислительный цикл последовательной программы состоит из попеременного решения двух задач: нахождения векторов собственного электромагнитного поля $\overline{E}^i$ и $\overline{B}^i$ (эйлеров этап) и интегрирования уравнений движения частиц под действием рассчитанных собственных полей (лагранжев этап). Поскольку электромагнитное поле определяется на эйлеровой сетке, а координаты и скорости частиц – на лагранжевой, то необходимы операция раздачи плотности заряда и плотности тока в узлы эйлеровой сетки для решения уравнений Максвелла и обратная

процедура интерполяции для интегрирования уравнений движения. В качестве такой процедуры применена PIC-интерполяция [1]. При этом вычисление сеточной плотности тока согласовано с изменением плотности заряда в ячейке эйлеровой сетки, что позволяет удовлетворить соотношениям (2) [1].

Выполненный тайминг последовательной программы показал, что доля временных затрат центрального процессора, приходящихся на вычисление собственного электромагнитного поля по уравнениям (2), составляет приблизительно $3 \cdot 10^{-4}$ при использовании пространственных сеток 100×100 и количества частиц приблизительно $4 \cdot 10^{6}$. Поскольку время выполнения эйлерова этапа алгоритма занимает много меньше времени выполнения лагранжева, проведено распараллеливание только лагранжева этапа. При параллельной реализации алгоритма использовался язык программирования Fortran 90 с библиотекой MPI. Частицы инжектируются в расчётную область, полностью рассматривающуюся каждым из процессов, при этом производится динамическая балансировка загрузки процессов вычислительной системы при расчёте движения частиц (см. ниже). На начальном шаге по времени частицы распределяются равномерно по процессам. Далее происходит раздача плотности тока в узлы сетки на каждом из процессов, на основе имеющихся координат частиц данного процесса. В результате получаются «неполные» плотности тока для каждого процесса. Для интегрирования уравнений Максвелла (1) необходимо собрать на каждом из процессов источники (сеточные плотности тока). Сборка (суммированием) рассчитанных «неполных» сеточных плотностей тока J_r, J_φ, J_z на всех процессах осуществляется процедурой MPI_ALLREDUCE. На эйлеровом этапе все процессы интегрируют уравнения Максвелла во всей расчётной области. Далее параллельно интегрируются уравнения движения распределённых по процессам частиц (3).

Балансировка загрузки процессов осуществляется при помощи алгоритма регулирования инжекции частиц: максимальное число частиц, инжектируемых на данном шаге по времени, получает процесс, имеющий в расчётной области минимальное число частиц. На очередном шаге по времени мастер-процесс получает число частиц, рассчитываемых остальными процессами, и в соответствии с дисбалансом загрузки распределяет инжектируемые частицы по процессам. Результаты расчётов показывают, что дисбаланс загрузки составляет малые доли процента, при этом временные затраты процессорного времени на балансировку минимальны. К недостаткам алгоритма можно отне-

сти его ориентированность на задачи, имеющие существенно инжекционный характер.

Результаты

На рис. 2 приведены графики зависимости ускорения и эффективности работы параллельной программы от числа процессов при количестве обрабатываемых частиц $40 \cdot 10^6$ и пространственных сетках 100×100. Пунктиром показано максимальное теоретическое ускорение, вычисленное в соответствии с законом Амдаля при доле вычислительных операций параллельной программы, которые выполняются сугубо последовательно, равной $3 \cdot 10^{-4}$. Эффективность работы параллельной программы рассчитывалась как отношение реального ускорения к теоретическому. Полученные результаты говорят о хорошей эффективности распараллеливания алгоритма.

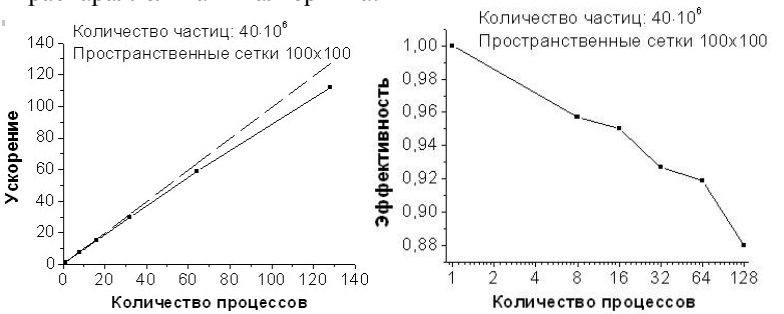
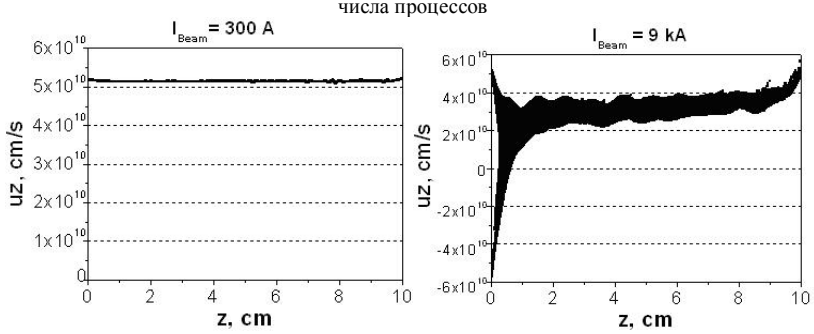


Рис. 2. Зависимости ускорения и эффективности работы параллельной программы от числа процессов



а) б)

Рис. 3. Зависимости z-компоненты скорости частиц от координаты z для тока пучка 300 А (*а*) и 9 кА (*б*)

Полученное на кластере ТГУ СКИФ Cyberia решение (рис. 3) хорошо согласуется с решением подобной задачи PIC-кодом KARAT [2] и PIC-кодом OOPIC. При этом видно образование виртуального катода на рис. 3, *б* при токе пучка выше критического (около 9 кА).

Литература

1. **Григорьев Ю.Н., Вшивков В.А., Федорук М.П.** Численное моделирование методами частиц-в-ячейках. Новосибирск: Изд-во СО РАН, 2004. 360 с.
2. **Tarakanov V.P.** Users's manual for code KARAT (ver. 7.03, May 97), Berkley Research Associates, Inc., 1997. 109 p.
3. **Вшивков В.А., Краева М.А., Малышкин В.Э.** Параллельная реализация метода частиц //Программирование, 1997. №2. С. 39-51.

Параллельная реализация вероятностного асинхронного клеточного автомата, моделирующего каталитическую реакцию окисления монооксида углерода (CO) на поверхности палладия (Pd110)

А.Е. Шарифулина
Новосибирский государственный технический университет

Предложена параллельная реализация вероятностной асинхронной клеточно-автоматной модели, имитирующей колебательную и автоволновую динамику реакции окисления монооксида углерода на поверхности палладия.

Введение

Каталитическое окисление монооксида углерода на поверхности палладия в условиях, далеких от равновесных, может сопровождаться появлением автоколебаний и химических волн. Каталитические реакции моделируются вероятностными асинхронными клеточными автоматами (КА), распараллеливание которых сопряжено с определёнными трудностями. В работе используется аппроксимация асинхронного КА блочно-синхронным КА [1]. В результате распараллеливания блочно-синхронного КА получено линейное ускорение.

Механизм реакции окисления CO на Pd_{110}

Реакция каталитического окисления CO на поверхности Pd_{110} описывается следующими уравнениями [2]:

1) $O_{2(gas)} + 2* \xrightarrow{k_1} 2O_{ads}$;
2) $CO_{gas} + * \underset{k_{-2}}{\overset{k_2}{\longleftrightarrow}} CO_{ads}$;
3) $CO_{ads} + O_{ads} \rightarrow CO_{2(gas)} + 2*$;
4) $O_{ads} + *v \xrightarrow{k_4} O_{sub}$;
5) $CO_{ads} + O_{sub} \xrightarrow{k_5} CO_{2(gas)} + 2* + *v$;
6) $CO_{gas} + O_{sub} \underset{k_{-6}}{\overset{k_6}{\longleftrightarrow}} [CO_{ads}O_{sub}]$;
7) $[CO_{ads}O_{sub}] \xrightarrow{k_7} CO_{2(gas)} + * + *v$;
8) $CO_{ads} + * \xrightarrow{k_{diff}} * + CO_{ads}$;
9) $CO_{ads} + O_{sub} \xrightarrow{k_{diff}} * + [CO_{ads}O_{sub}]$;
10) $[CO_{ads}O_{sub}] + O_{sub} \xrightarrow{k_{diff}} O_{sub} + [CO_{ads}O_{sub}]$,

где * и *v – активные центры поверхностного и приповерхностного слоя, k_i и k_{-i} – константы скорости прямых и обратных элементарных стадий реакции соответственно. Первая стадия описывает необратимую адсорбцию кислорода. Адсорбция и десорбция CO представлены стадией 2. Реакции между молекулами CO_{ads} и атомами кислорода O_{ads} соответствует третья стадия. Образование слоя приповерхностного кислорода O_{sub} отображено стадией 4. Пятая стадия описывает реакцию между молекулами CO_{ads} и O_{sub}. Образование комплекса молекул CO_{ads} на растворенном кислороде в виде $[CO_{ads}\ O_{sub}]$ происходит через прямую адсорбцию CO из газовой фазы (стадия 6). Стадия 7 описывает разрушение комплекса $[CO_{ads}\ O_{sub}]$, сопровождающееся образованием молекул CO_2 с освобождением адсорбционных мест * и *v. Стадии 8–10 описывают диффузию молекул CO_{ads} по поверхности палладия. Диффузия CO_{ads} на свободные участки поверхности представлена стадией 8. Стадия 9 описывает образование комплекса $[CO_{ads}\ O_{sub}]$ вследствие диффузии CO_{ads} по поверхности. Диффузия комплекса $[CO_{ads}\ O_{sub}]$ отображена стадией 10.

Каждая из стадий 1,2,4–10 реализуется с определённой вероятностью $w_i = k_i / \sum k_i$ где k_i – константа скорости данной стадии, $i = \{1, 2, -2, 4, 5, 6, -6, 7\}$. Значения констант скоростей стадий выбраны в соответствии с парциальным давлением реагирующих веществ, концентрацией активных центров поверхности и коэффициентами прилипания реагентов к поверхности палладия [3]. Стадия 3 выполняется каждый раз при наличии соответствующих реагентов после попытки адсорбции CO или O_2, а также после диффузии молекул CO по поверхности.

Константа скорости стадий 8–10, описывающих диффузию, вычисляется в зависимости от параметра интенсивности диффузии M_{diff}: $k_{diff} = M_{diff} \cdot \sum k_i$. Все процессы на поверхности катализатора происходят асинхронно.

Реакция окисления сопровождается колебаниями скорости реакции и концентрации веществ, адсорбирующихся на поверхности Pd_{110}. При этом на моделируемой поверхности возникают различные волновые процессы. Причина колебаний на поверхности палладия – фазовые переходы в адсорбированных слоях на поверхности катализатора. Периодическое образование и расходование приповерхностного кислорода в ходе реакции изменяет адсорбционные и каталитические свойства поверхности. Смена покрытий поверхности различными реагентами сопровождается образованием подвижной волны, фронт которой характе-

ризуется высокой концентрацией активных центров, обеспечивающих максимальную скорость образования молекул CO_2 [3].

КА-модель реакции окисления

Для моделирования реакции окисления CO на Pd_{110} используется вероятностный асинхронный клеточный автомат (КА). В терминах КА-моделирования поверхность катализатора представлена клеточным массивом $\Omega(A,M\times N)$ (рис. 1), где $A=\{*, CO_{ads}, O_{ads}, O_{sub}, [CO_{ads}\ O_{sub}]\}$ – алфавит состояний, установленный согласно уравнениям реакции (1–7), M×N – размер двумерного массива. Каждой клетке массива ставится в соответствие пара (x,m), где $x \in A$ обозначает состояние клетки, а $m=(i,j):\{i=0..M, j=0..N\}$ – это имя клетки, обозначающее её координаты в массиве.

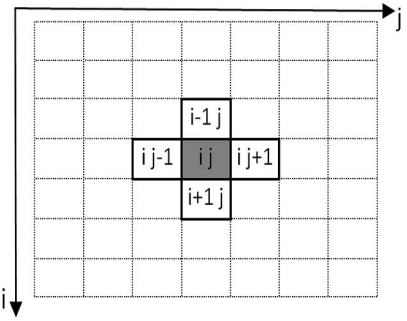


Рис. 1. Клеточный массив

Правила перехода КА соответствуют уравнениям реакции (1–10). Вероятность выбора правила для каждой клетки определяется вероятностью w_i. Новое состояние клетки (i,j) вычисляется в зависимости от состояний соседних клеток $\{(i–1, j), (i, j+1), (i+1, j), (i, j–1)\}$ (см. рис. 1). Режим работы данного клеточного автомата – асинхронный.

При срабатывании правил переходов, соответствующих адсорбции O_2 и диффузии CO, для клетки (i,j) выбирается одна из четырёх соседних клеток, к которой применяется правило, соответствующее стадии 3. Новое состояние выбранной клетки также вычисляется в зависимости от состояний четырёх соседних клеток. Поэтому при моделировании используется шаблон, отличный от исходного (рис. 2). На каждом такте осуществляется выбор одной из стадий (1–2, 4–10). Итерация состоит из M×N тактов. На границах поверхности заданы периодические граничные условия.

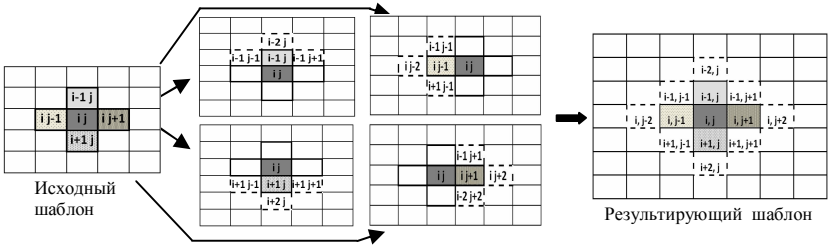


Рис. 2. Шаблон моделирования

Параллельная реализация реакции окисления

Известно, что при распараллеливании асинхронных КА-режимов межпроцессорный обмен должен выполняться всякий раз, когда изменяется состояние хотя бы одной граничной клетки. Поэтому в данной работе используется аппроксимация асинхронного КА блочно-синхронным (БСКА) [1]. Суть блочно-синхронного режима работы заключается в следующем.



Рис. 3. Блочно-синхронный режим работы КА

1. КА-массив разбивается на непересекающиеся блоки, состоящие из тринадцати клеток. Блок выбран в соответствии с шаблоном моделирования.
2. На каждом такте во всех блоках выбираются клетки с одинаковым номером (например, на рис. 3 обрабатываются клетки с номером 6). Порядок выбора клеток не важен, т.к.

функции переходов вычисляются только от клеток, принадлежащих одному блоку.

3. Применение правил переходов ко всем клеткам КА-массива выполняется за 13 тактов.

При распараллеливании алгоритма реакции окисления для клеточного автомата размером 3000×3000 на 30 узлах суперкомпьютера НКС-160 (ИВМиМГ СОРАН) получено линейное ускорение, поскольку при блочно-синхронном режиме работы КА межпроцессорный обмен осуществляется в конце каждого такта один раз, в отличие от асинхронного КА.

Заключение

В работе реализован параллельный алгоритм моделирования поверхностной реакции окисления CO на палладии с помощью аппроксимации асинхронного КА блочно-синхронным КА. В результате распараллеливания БСКА получено линейное ускорение.

Литература

1. **Bandman O.L.** Parallel simulation of asynchronous cellular automata evolution // Lecture Notes in Computer Science. Berlin: Springer. 2006. Vol. 4173. P. 41-48.
2. **Elokhin V.I., Latkin E.L, Matveev A.V., Gorodetskii V.V.** The role of subsurface oxygen in oscillatory behaviour of CO + O_2 reaction over Pd metal catalysts: Monte Carlo model // Journal of Molecular Catalysis A: Chemical. 2000. Vol. 158. P. 161-166.
3. **Elokhin V.I., Latkin E.I., Matveev A.V., Gorodetskii V.V.** Application of statistical lattice models to the analysis of oscillatory and autowave processes in the reaction of carbon monoxide oxidation over platinum and palladium surfaces // Kinetics and Catalysis. 2003. Vol. 44, №. 5. P. 692-700.

Параллельные алгоритмы на графах

В.Н. Берцун, А.В. Бородин
Томский государственный университет

Рассматриваются последовательный и параллельный алгоритмы Флойда поиска кратчайших путей и количества путей в графе. Приводятся оценки их эффективности на основе результатов вычислительного эксперимента на кластере СКИФ Cyberia.

Пусть дан ориентированный граф $G(V, E)$ на *n* вершинах и его матрица примыканий [1]

$$A^{(1)}{}_p = \begin{cases} w_{ij}, \text{ при } x_i x_j \in E, \\ 0, \text{при } i = j, \text{где } w_{ij} \text{ - вес ребра } x_i \text{из в } x_j \\ \infty \text{ при } x_i x_j \notin E. \end{cases}$$

Матрицу $A_p^{(2)}$ по матрице примыканий $A_p^{(1)}$ можно построить, находя элементы $A_p^{(2)}$ по формуле $a_{i,j}^{(2)} = \min(a_{ij}^{(1)}, a_{ik}^{(1)} + a_{kj}^{(1)}), k = \overline{1,n}$. Последовательно вычисляя $A_p^{(i)}, 2 \le i \le N$, где N – наименьшее 2^S (S – натуральное), $2^s < n–1$, получим матрицу кратчайших путей $A_p^{(N)}$.

По матрице $A_p^{(N)}$ можно найти также и некоторые другие характеристики графа: эксцентриситет его вершин, радиус и диаметр графа.

Вершины, через которые проходят кратчайшие пути, определяются с помощью записи информации о самих путях (наряду с информацией о длинах путей) в матрице $B = [b_{ij}]$. Элемент b_{ij} указывает вершину, непосредственно предшествующую вершине x_i в кратчайшем пути от x_i к x_j. Элементам матрицы B присваиваются начальные значения $b_{ij} = i$ для всех i и j.

Обновление матрицы B происходит следующим образом:

$$b_{ij} = \begin{cases} b_{ij}, \textit{если } (a_{ik} + a_{kj}) < a_{ij}, \\ \textit{неизменяется, если } (a_{ik} + a_{kj}) \ge a_{ij}. \end{cases}$$

Вершины кратчайших путей получаются непосредственно из заключительной матрицы B после вычисления $A^{(N)}{}_p$. Тогда кратчайший

путь между двумя вершинами x_i и x_j дается следующей последовательностью вершин [2,3]:

$$x_i, x_n, ..., x_2, x_1, x_j \ ,$$

где $x_1 = b_{ij}, x_2 = b_{ix_1}, x_3 = b_{ix_2}$ и т. д. до $x_i = b_{ix_n}$.

Как правило, число доступных процессоров p существенно меньше, чем число базовых задач m ($p<<m$). Возможный способ укрупнения вычислений состоит в использовании ленточной схемы разбиения матрицы примыканий A и вспомогательной матрицы B. Такой подход соответствует объединению в рамках одной базовой подзадачи вычислений, связанных с определением элементов одной или нескольких строк (горизонтальное разбиение) или столбцов (вертикальное разбиение) матриц A и B. Так как для алгоритмического языка Fortran массивы располагаются по столбцам, использовалось разбиение матриц A и B на вертикальные полосы. При таком способе разбиения данных на каждой итерации алгоритма Флойда потребуется передавать между процессорами только элементы одного из столбцов матриц A и B.

Ускорение алгоритма, полученное при выполнении на идеальной параллельной машине из p процессоров (без учета обменов) :

$$S_p = \frac{1}{S + (1 - S)/p},$$

где S – доля операций последовательной части алгоритма [6 – 8].

Ускорение можно определить и как отношение процессорного времени на выполнение последовательной программы ко времени выполнения вычислений параллельной программой на p-процессорной машине. При этом предполагается, что обе программы реализуют один и тот же алгоритм.

Формулировка закона Амдаля не учитывает временных затрат на обеспечение обмена информацией между вычислительными узлами. Поскольку эти затраты, обусловленные необходимостью выполнения коммуникационных обменов, предотвращения конфликтов памяти и синхронизации, лишь увеличивают общее время работы параллельной программы, то реальное ускорение будет всегда меньше идеального.

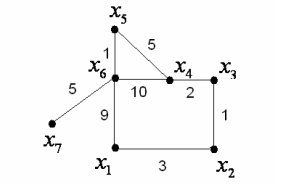


Рис. 1. Неориентированный граф

Для неориентированного графа из рис. 1 в результате расчетов по алгоритму Флойда получены матрица кратчайших путей и вспомогательная матрица:

$$A_p^{(8)} = \begin{pmatrix} 0 & 3 & 4 & 6 & 10 & 9 & 14 \\ 3 & 0 & 1 & 3 & 8 & 9 & 14 \\ 4 & 1 & 0 & 2 & 7 & 8 & 13 \\ 6 & 3 & 2 & 0 & 5 & 6 & 11 \\ 10 & 8 & 7 & 5 & 0 & 1 & 6 \\ 9 & 9 & 8 & 6 & 1 & 0 & 5 \\ 14 & 14 & 13 & 11 & 6 & 5 & 0 \end{pmatrix} \qquad B^{(8)} = \begin{pmatrix} 1 & 1 & 2 & 3 & 6 & 1 & 6 \\ 1 & 2 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 5 & 6 \\ 1 & 3 & 3 & 4 & 5 & 6 & 6 \\ 1 & 1 & 4 & 4 & 5 & 6 & 7 \end{pmatrix}$$

Используя эти матрицы, получим, например, последовательность вершин кратчайшего пути длины 6 из вершины x_1 в вершину x_4. Из матрицы $B^{(8)}$ восстановим вершины этого кратчайшего пути по алгоритму, описанному выше:

$x_1^p = b_{14} = 3,\ x_2^p = b_{1x_1^p} = b_{13} = 2,\ x_3^p = b_{1x_2^p} = b_{12} = 1$.

Таким образом, кратчайший путь проходит через вершины x_2^p, x_1^p, т.е. через вершины x_2 и x_3.

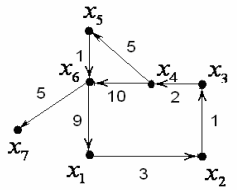


Рис. 2. Ориентированный граф

В случае ориентированного графа из рис. 2 искомые матрицы имеют вид

$$A_p^{(8)} = \begin{pmatrix} 0 & 3 & 4 & 6 & 11 & 12 & 17 \\ 18 & 0 & 1 & 3 & 8 & 9 & 14 \\ 17 & 20 & 0 & 2 & 7 & 8 & 13 \\ 15 & 18 & 19 & 0 & 5 & 6 & 11 \\ 10 & 13 & 14 & 16 & 0 & 1 & 6 \\ 9 & 12 & 13 & 15 & 20 & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix} \qquad B^{(8)} = \begin{pmatrix} 1 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 3 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 4 & 4 & 5 & 6 \\ 6 & 1 & 2 & 3 & 5 & 5 & 6 \\ 6 & 1 & 2 & 3 & 4 & 6 & 6 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

Например, кратчайший путь из вершины x_1 в вершину x_7 по этим матрицам определится так:

$$x_1^p = b_{17} = 6,\ x_2^p = b_{1x_1^p} = b_{16} = 5,\ x_3^p = 4,\ x_4^p = 3,\ x_5^p = 2,\ x_6^p = 1.$$

Таким образом, кратчайший путь проходит через вершины $x_5^p, x_4^p, x_3^p, x_2^p, x_1^p$, т.е. через вершины x_2, x_3, x_4, x_5, x_6.

В таблице приведены результаты оценки эффективности распараллеливания вычислений на вычислительном кластере ТГУ СКИФ Cyberia.

Количество вершин графа	Последовательный алгоритм, с	Параллельный алгоритм					
		10 процессоров		50 процессоров		100 процессоров	
		Время,с	Ускорение	Время,с	Ускорение	Время, с	Ускорение
500	3,75	0,44	8,52	0,26	14,42	0,30	12,5
1000	54,64	6,34	8,61	2,78	19,65	1,89	28,90
3000	2588,6	268,6	9,63	59,06	43,83	32,2	80,25

Таким образом, с увеличением количества вершин графа растет ускорение параллельного алгоритма, так как количество обменов между процессорными элементами растет медленнее, чем количество арифметических операций в алгоритме Флойда, сложность которого $O(n^3)$.

Известно [2, 3], что если $A = (\alpha_{ij})_{n\times n}$ – матрица смежности графа G без петель и $A^l = (\gamma_{ij})_{n\times n}$, где $l \in N$, тогда γ_{ij} равно числу (v_i, v_j) маршрутов длины l . Это верно и для графов с петлями, если считать, что каждая петля имеет два обхода (этим число маршрутов, проходящих через данную петлю, увеличивается в два раза).

Поскольку матрица смежности A графа G является вещественной симметрической матрицей и так как она ортогонально подобна некоторой вещественной диагональной матрице D то

$$A = T^{-1}DT \ , \ A^l = T^{-1}D^lT ,$$

где матрица D имеет отличные от нуля элементы только на главной диагонали, и они совпадают со спектром графа G.

Это соотношение позволяет вычислять степени матрицы A^l по известному спектру графа, используя параллельные алгоритмы умножения матриц.

Литература

1. **Макконелл Дж.** Анализ алгоритмов. Вводный курс. М.:Техносфера, 2002. 304 с.
2. **Харари Ф.** Теория графов. М.: Мир, 2003. 300 с.
3. **Кристофидес Н.** Теория графов: алгоритмический подход. М.,1978. 432 с.
4. **Берцун В.Н.** Математическое моделирование на графах. Ч. 1. Томск: Изд-во НТЛ, 2006. 88 с.
5. **Оре О.** Графы и их применение. М.:Мир, 2002. 71 с.
6. **Воеводин В.В.** Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296 с.
7. **Старченко А.В., Есаулов А.О.** Параллельные вычисления на многопроцессорных вычислительных системах. Томск: Изд-во ТГУ, 2002. 55 с.
8. **Дацюк В.Н., Букатов А.А., Жегуло А.И.** Многопроцессорные системы и параллельное программирование. Ч. 2. Среда параллельного программирования MPI. Ростов: Изд-во РГУ, 2000. 65 с.

Анализ параллельных реализаций метода конечных элементов для моделей мелкой воды*

Е.В. Дементьева, Е.Д. Карепова
Институт математики Сибирского федерального университета,
Красноярск

Представлено исследование эффективности нескольких параллельных реализаций алгоритма численного решения краевой задачи для уравнений мелкой воды, выполненных с помощью библиотеки MPI для языка Си.

Моделирование поверхностных волн в водоемах методом конечных элементов

Численное моделирование поверхностных волн в больших акваториях проводилось с учетом сферичности Земли и ускорения Кориолиса на основе уравнений мелкой воды [1]. В [2] для этой задачи построен метод конечных элементов (МКЭ) и дифференциальная задача сведена к векторно-матричной форме. Полученная система линейных алгебраических уравнений решается итерационным методом Якоби, который обладает хорошим параллелизмом, а диагональное преобладание для его сходимости обеспечивается выбором шага по времени [3].

Параллельный алгоритм

Используя явный параллелизм по данным, исходную расчетную область можно разбить на несколько частично перекрывающихся подобластей, в каждой из которых расчеты выполняются независимо в рамках итерации Якоби. После каждой итерации необходимо проводить согласование данных в перекрытиях. Имеет место, по крайней мере, два варианта разбиения – с теневыми гранями и без них [3]. Декомпозиция без теневых граней более экономична по памяти, проста в программировании, однако предполагает дополнительные арифметические операции на каждой итерации Якоби. Очевидно ее достоинство для неструктурированных сеток.

При использовании библиотеки MPI возможно два принципиально разных способа организации обменов — с использованием блокирую-

* Работа поддержана грантом РФФИ № 08-01-00621.

щих или неблокирующих функций передачи данных [3]. В работе [3] оценено время, необходимое для обменов в обоих случаях, а также выполнены теоретические оценки возможного ускорения каждого из параллельных алгоритмов.

Вычислительный эксперимент

Для численного исследования ускорения параллельного алгоритма рассмотрена модельная задача для квадратной области на сфере с твердыми границами [2]. В расчетной области построена равномерная квадратная сетка 801×801 точек с соответствующей согласованной триангуляцией. В вычислительных экспериментах было сделано 1000 шагов по времени.

Расчеты были проведены на трех высокопроизводительных ВС. Во-первых, вычисления выполнялись на 99-ядерном кластере ИВМ СО РАН, отличительной чертой которого является его гетерогенная архитектура (кластер содержит однопроцессорные одноядерные, однопроцессорные двуядерные и двухпроцессорные двуядерные вычислительные узлы, сеть передачи данных — GigabitEthernet). Вторым кластером является кластер МВЦ ТГУ СКИФ Cyberia, который содержит 283 двухпроцессорных двуядерных вычислительных узла, объединенных высокопроизводительной сетью передачи данных InfiniBand. Наконец, эксперименты были частично повторены на высокопроизводительном вычислительном комплексе ИВЦ НГУ (HP-НГУ) на базе кластера из 64 восьмиядерных вычислительных узлов, основанных на блэйд-серверах Hewlett-Packard BL460c, объединенных сетью InfiniBand. Кластеры SKIF Cyberia и HP-кластер ИВЦ НГУ отличаются количеством ядер, приходящихся на узел.

Исследования зависимости ускорения проводились вплоть до 32 процессов, где на рассматриваемой сетке алгоритм хорошо масштабируем. Численные эксперименты показали несомненное преимущество однородной архитектуры ВС над гетерогенной, неблокирующих двухточечных обменов над блокирующими и несущественное различие между способами декомпозиции расчетной области.

На рис. 1 представлена зависимость ускорения вычислений от количества используемых процессов, полученных на кластерах ИВМ СО РАН и SKIF Cyberia. Для сравнения представлен график потенциального ускорения согласно теоретической оценке (см. [3]). В экспериментах на кластере ИВМ СО РАН общий тренд ускорения совпадает с теоретическими оценками, однако его рост носит сильно неустойчивый характер. Расчеты, проведенные на кластере СКИФ Cyberia, пока-

зывают классическую картину ускорения, подтверждающую линейный характер его роста с увеличением количества процессоров с эффективностью около единицы. Эксперименты на НР-НГУ подтверждают эти результаты. Таким образом, неустойчивый характер ускорения в расчетах, проведенных на кластере ИВМ СО РАН, объясняется, скорее всего, его неоднородной архитектурой.

Поскольку время выполнения программы складывается из времени вычислений, времени коммуникаций и дополнительного времени для операций, связанных с распределенностью данных, то были проведены серии расчетов, в которых названные составляющие были измерены отдельно для случая декомпозиции без перекрытий и неблокирующего режима двухточечных обменов (рис. 2). Время выполнения вычислений (см. рис. 2, вверху) для гомогенных архитектур совпадает с теоретическими оценками. Ввиду неоднородности узлов кластера ИВМ СО РАН время вычислений в этом случае уменьшается немонотонно.

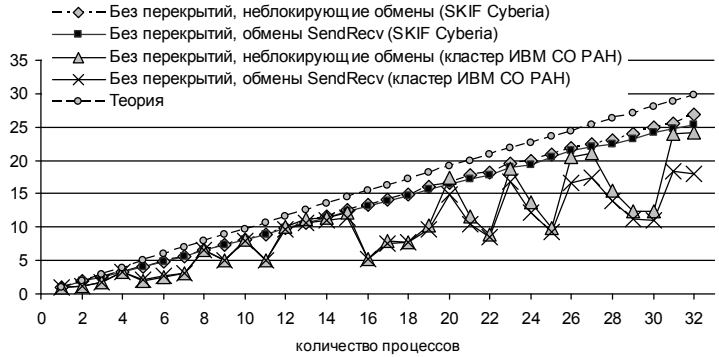


Рис. 1. Зависимость ускорения вычислений от количества доступных процессов

Время выполнения обменов в неблокирующем режиме не зависит от количества процессов, участвующих в расчетах. Для кластера с неоднородной архитектурой наблюдается ожидаемое немонотонное поведение исследуемой величины. Однако немонотонное поведение наблюдается и в случае расчетов для гомогенных архитектур. Исследование времени выполнения обменов показало следующее: 1) время обменов минимально и не зависит от количества процессов, участвующих в обменах, если загружены все ядра на узле по одному процессу на ядро; 2) при существовании в расчетах узлов, не полностью загруженных, время обменов тем больше, чем больше количество простаивающих ядер; 3) время, затраченное на обмены, уменьшается с

ростом количества задействованных процессов (т.е. с уменьшением времени вычислений).

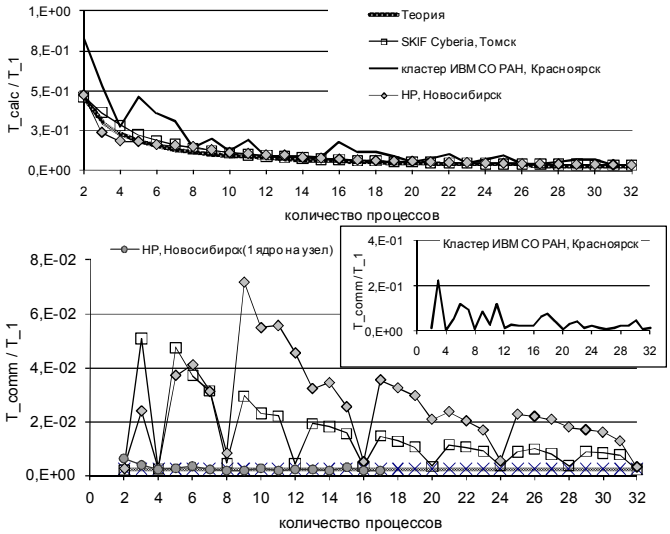


Рис. 2. Исследование времени счета и коммуникаций

На рис. 2 (внизу) представлены зависимости времени выполнения неблокирующих обменов (НР-НГУ) от количества используемых процессов для двух способов загрузки узлов: 1) используется лишь одно ядро на каждом расчетном узле; 2) загружено максимально возможное число ядер на используемых узлах. Данное исследование показало преимущество первого способа загрузки вычислительных узлов, когда время, потраченное на обмены, соответствует теоретическим оценкам. В этом случае ускорение вычислений возрастает линейно и значительно превышает ускорение, полученное при загрузке узлов вторым способом. Недостатком первого способа, однако, является неэффективное использование ресурсов кластера, поскольку незадействованные ядра в узлах простаивают в ожидании окончания вычислений.

Авторы благодарят коллектив МВЦ ТГУ и ИВЦ НГУ, профессора А.В. Старченко и Д.Л. Чубарова за предоставленную возможность проведения серии вычислительных экспериментов на кластере СКИФ Cyberia и кластере ИВЦ НГУ.

Литература

1. **Agoshkov V.I.** Inverse problems of the mathematical theory of tides: boundary-function problem // Russ. J. Numer. Anal. Math. Modelling. 2005. Vol. 20, № 1. P. 1-18.
2. **Kamenshchikov L.P.** Simulation of surface waves in basins by the finite element method / Kamenshchikov L.P., Karepova E.D., Shaidurov V.V. // Russian J. Numer. Anal. Math. Modelling. 2006. Vol. 21(4). P. 305-320.
3. **Карепова Е.Д., Шайдуров В.В.** Параллельная реализация МКЭ для начально-краевой задачи мелкой воды // Вычислительные технологии. 2009. Т. 14, № 6. С. 45-57.

Вариационное моделирование пространственных кривых с помощью нового типа эрмитовых кубических сплайн-вейвлетов*

Н.К. Аркабаев, Б.М. Шумилов, Э.А. Эшаров, А.Ж. Кудуев
Ошский государственный университет, Киргизия
Томский государственный архитектурно-строительный университет

Описано построение соотношений разложения и восстановления для нового типа эрмитовых кубических сплайн-вейвлетов, полуортогональных в метрике <u',v'>. Предложено применение полученной системы базисных сплайн-мультивейвлетов для вариационного моделирования трасс автомобильных дорог и других линейно-протяженных пространственных объектов.

Вейвлет-преобразование является математическим инструментом, с помощью которого можно иерархически разложить заданную функцию (сигнал, изображение, кривую или поверхность) на грубое приближенное представление с некоторыми деталями. И. Добеши [1] впервые построила ортонормальные вейвлеты неполиномиального типа с компактным носителем. Чуи и др. [2] построили полуортогональные сплайн-вейвлеты. Недостатками построенных ранее вейвлетов [1–3] является то, что они либо не имеют аналитического представления, либо расположены на достаточно широком носителе. И то и другое бывает чрезвычайно важно при их использовании для приближенного решения задач численного анализа.

Один из конструктивных подходов к построению вейвлетов состоит в использовании базисных *B*-сплайнов. Пусть степень восполняющего многочлена равна 3. Здесь *восполняющий многочлен* – это многочлен, который совпадает со значениями сплайн-реконструкции соответствующего вейвлета между узлами (точками склейки соседних кусков сплайна). В [4] был предложен новый тип эрмитовых мультивейвлетов, полуортогональных в метрике $\langle u', v' \rangle$. Доказано, что предложенное условие полуортогональности позволяет уменьшить длину носителя базисных вейвлетов до величины [0,2]. В данной работе для этого типа мультивейвлетов с помощью предложенного в [5] метода

* Работа выполнена при поддержке РФФИ, проекты №09-07-90900, №09-01-90716.

неопределенных коэффициентов получены соотношения разложения и восстановления.

Двухмасштабные соотношения для базисных эрмитовых сплайнов 3-й степени можно получить в виде следующих формул [7]:

$$\varphi_3(t)=\frac{1}{2}\varphi_3(2t)+\varphi_3(2t-1)+\frac{1}{2}\varphi_3(2t-2)+\frac{3}{4}\big(\psi(2t)-\psi(2t-2)\big),$$
$$\psi(t)=\frac{1}{2}\psi(2t-1)-\frac{1}{8}\psi(2t)-\frac{1}{8}\psi(2t-2)-\frac{1}{8}\Big(\varphi_3(2t)-\varphi_3(2t-2)\Big). \qquad (1)$$

Здесь [6]: $\varphi_3(t)=\begin{cases}(1-t)^2(1+2t), & 0\le t\le 1,\\ t^2(3-2t), & 1\le t\le 2,\\ 0, & t\notin[0,2];\end{cases}$ $\psi(t)=\begin{cases}t(1-t)^2, & 0\le t\le 1,\\ -t^2(1-t), & 1\le t\le 2,\\ 0, & t\notin[0,2].\end{cases}$

Базисные вейвлеты представляют дополнение пространства сплайн-разложений на прореженной сетке до пространства сплайнов на густой сетке:

$$\psi\varphi_3(x)=-2\varphi_3(2x)+4\varphi_3(2x-1)-2\varphi_3(2x-2)-21\psi(2x)+21\psi(2x-k), \qquad (2)$$
$$\psi\psi(x)=\varphi_3(2x)-\varphi_3(2x-2)+9\psi(2x)+12\psi(2x-1)+9\psi(2x-k). \qquad (3)$$

Определение. Пусть имеется двумерный дискретный сигнал, содержащий значения в узлах эрмитового сплайна и его производные $\{S=\{s_j\}, S'=\{s'_j\}\}$, $j=0,1,\ldots,m$, $m=2^k$, $k\ge 1$. Согласно (1)–(3) вычисляются значения:

$$v_j=\frac{1}{4}(S_{2j}+2S_{2j+1}+S_{2j+2})+\frac{3}{8}(S'_{2j}-S'_{2j+2}),\ v'_j=-\frac{1}{2}(S'_{2j}-4S'_{2j+1}+S'_{2j+2})-\frac{1}{2}(S_{2j}-S_{2j+2}), \qquad (4)$$
$$w_j=-2S_{2j}+4S_{2j+1}-2S_{2j+2}-21S'_{2j}+21S'_{2j+2},\ w'_j=S_{2j}-S_{2j+2}+9S'_{2j}+12S'_{2j+1}+9S'_{2j+2}.$$

Эти значения формируют два новых двумерных сигнала $\{V=\{v_j\}$, $V'=\{v'_j\}\}$ и $\{W=\{w_j\}$, $W'=\{w'_j\}\}$, $j=0,1,\ldots,2^{m-1}$, первый из которых является огрубленной версией исходного сигнала (каждой тройке элементов S соответствует их экспоненциальное среднее, так что для случая линейных зависимостей значения v_j и v'_j в точности равны соответствующим промежуточным значениям S_{2j+1} и S'_{2j+1}). Коэффициенты w_j и w'_j далее называем *вейвлет-коэффициентами*, и они содержат детализирующую информацию, необходимую для восстановления исходного сигнала в соответствии с формулами

$$S_{2j}=\frac{1}{8}\left(\frac{1}{4}w_j+7v_j-3S_{2j-1}-3S_{2j+1}\right),\ S'_{2j}=\frac{1}{24}\left(\frac{w'_j+144w_j}{7}+5S'_{2j-1}+5S'_{2j+1}\right) \qquad (5)$$

для четных узлов. Недостающие значения для нечетных узлов определяются из соотношений

$$35S_{2j-1} - 74S_{2j+1} + 35S_{2j+3} = \frac{1}{4}(w_{j+1} + w_j) - v_{j+1} - v_j + 24(v'_j - v'_{j+1} + \frac{1}{g_1}(w'_j - w'_{j+1})),$$
$$-\frac{1}{2}S'_{2j-1} + 5S'_{2j+1} - \frac{1}{2}S'_{2j+3} = 4(v_{j+1} - v_j) + \frac{4}{q_1}(w_{j+1} - w_j) - 16(v'_j + v'_{j+1} + \frac{1}{g_1}(w'_j + w'_{j+1})). \quad (6)$$

Для замыкания полученных трехдиагональных систем уравнений добавляются по два оборванных слева и справа уравнения. Формулы (4) и (5), (6) определяют, соответственно, *прямое и обратное преобразование* двумерного дискретного сигнала на любом уровне разрешения.

В пакете программ автоматизированного проектирования автомобильных дорог (САПР «IndorCAD/RoAD», ООО «Индорсофт», г. Томск) [8]) трасса автомобильной дороги описывается как набор отрезков прямой и дуг окружностей, к которым добавлены сопряжения кривых плана дугами клотоид и кривыми Безье, а также интерполяция кривых профиля кубическими сплайнами. В процессе редактирования плана и профиля пользователь производит операции с геометрическими элементами изображения явным образом – либо передвигая контрольные точки и решая задачу сплайн-интерполяции [6] (метод «прямого манипулирования»), либо прибегая к сглаживанию кривой в заданном коридоре ограничений с помощью многократного применения формул сплайн-фильтрации [9, 10]. Известны, по крайней мере, еще два подхода к редактированию плоских сплайн-кривых. Первый состоит в решении полной задачи построения сплайна в выпуклом множестве [9]. Достоинством этого подхода является то, что в процессе решения экстремальной задачи удаляются «лишние» узлы, в результате чего достигается сжатие информации. Недостатки проявляются в неестественности минимизируемого функционала (интеграл от квадрата второй производной вместо полной кривизны), возможности утраты существенной, с точки зрения инженера-дорожника, информации и отсутствии надежных алгоритмов. Второй подход основан на методике вставки/удаления узлов *B*-сплайн разложения [11], когда поведение сплайн-кривой локально уточняется по мере введения дополнительных узлов либо кривая сглаживается путем идентификации и затем удаления «плохих» узлов. Некоторые предложения по правилам отбора удаляемых узлов в соответствии с аппроксимационным критерием содержатся, например, в [12]. Общее правило гласит, что если число параметров сплайна меньше, чем число предписанных пользователем значений положения, касательных и/или значений старших производных в некоторых узлах, то отыскивается решение в смысле метода наименьших квадратов; если же число параметров больше, чем число связей, то применяется некоторый вариант метода регуляриза-

ции А.Н. Тихонова [13]. Преимущество этого подхода состоит в возможности перехода к *B*-сплайн-вейвлет разложению с целью реализации кратномасштабного анализа [2] при редактировании сплайн-кривой. Последовательное осуществление метода лифтинга [3] в процессе кратномасштабного анализа продемонстрировано в [14]. В статье [15], наоборот, вейвлет-преобразование нового типа используется непосредственно для разложения исходной кривой на различные частотные составляющие. Высокочастотные шумы могут быть отфильтрованы в процессе реконструкции. При этом настоятельно требуется, чтобы алгоритм фильтрации ограничивался ликвидацией нежелательных дефектов и сохранял изогеометрические характеристики кривой [8]. Использование здесь эрмитовых сплайнов позволяет в явном виде, через значения сплайн-коэффициентов, учесть как геометрические ограничения на контрольные точки сплайна (пикеты автомобильной трассы), так и на тангенсы трассы (заданные направления касательных, например, при въезде на мостовое сооружение). Дополнительно ожидается ускорение вычислений на каждой итерации кратномасштабного анализа за счет уменьшения носителей базисных мультивейвлетов. В [14] представлен еще один метод ускорения вычислений, при котором все геометрические ограничения приводятся к одному уровню вейвлет-иерархии и минимизируется только кривизна полученной сплайн-кривой. Этот прием позволяет сохранить детали исходной кривой, которые обычно теряются в обычном методе вариационного моделирования.

Литература

1. **Добеши И.** Десять лекций по вейвлетам: Пер. с англ. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. 332 с.
2. **Чуи Ч.** Введение в вейвлеты: Пер. с англ. М.: Мир, 2001. 412 с.
3. **Столниц Э., Де Роуз Т., Салезин Д.** Вейвлеты в компьютерной графике: Пер. с англ. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002. 272 с.
4. **Jia R.-Q., Liu S.-T.** Wavelet bases of Hermite cubic splines on the interval // Advances Computational Mathematics. 2006. Vol. 25. P. 23-39.
5. **Шумилов Б.М., Эшаров Э.А.** Построение эрмитовых сплайн-вейвлетов // Вестник Томского государственного университета. Сер. Математика. Кибернетика. Информатика. Приложение. 2006. № 19. С. 260-266.

6. **Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л.** Методы сплайн-функций. М.: Наука, 1980. 352 с.
7. **Heil C., Strang G., Strela V.** Approximation by translate of refinable functions // Numer. Math. 1996. Vol. 73. P. 75-94.
8. Система автоматизированного проектирования автомобильных дорог. «ReCAD». Руководство по применению. Кн. 1–5. Томск, 2001. 428 с.
9. **Вершинин В.В., Завьялов Ю.С., Павлов Н.Н.** Экстремальные свойства сплайнов и задача сглаживания. Новосибирск: Наука, 1988. 102 с.
10. **Бойков В.Н., Шумилов Б.М.** Сплайны в трассировании автомобильных дорог. Томск: Изд-во ЦНТИ, 2001. 164 с.
11. **Де Бор К.** Практическое руководство по сплайнам: Пер. с англ. М.: Радио и связь, 1985. 304 с.
12. **Wang W., Zhang Y.** Wavelets-based NURBS simplification and fairing // Comput. Methods Appl. Mech. Engrg., 2009, doi: 10.1016 / j.cma. 2009.04.003.
13. **Тихонов А.Н., Арсенин В.Я.** Методы решения некорректных задач. М.: Наука, 1979. 285 с.
14. **Zhao G., Xu S., Li W., Teo O.-E.** Fast variation design of multiresolution curves and surfaces with B-spline wavelets // Comput. Aided Design. 2005. Vol. 37. P. 73-82.
15. **Шумилов Б.М., Эшаров Э.А.** Нестационарные сплайн-вейвлеты в ГИС и САПР линейно-протяженных пространственных объектов // Вестник Томского государственного архитектурно-строительного университета. 2006. № 1(12). С. 153-163.

Выбор оптимальной структуры композита с использованием генетического алгоритма

Ю.В. Геренг, Ю.Н. Сидоренко
Томский государственный университет

Рассматривается задача оптимизации параметров структуры армирования композита. Управляемыми параметрами являются объемное содержание армирующих элементов и тип структуры армирования материала. Критерий оптимальности записывается относительно характеристик жесткости и живучести композита. Для поиска оптимальных значений управляемых параметров использован генетический алгоритм. Показано, что метод обладает высокой скоростью сходимости уже при малых объемах популяции решений.

В основе технологий создания многих современных конструкционных материалов лежит идея объединения в единый комплекс компонентов, обладающих разными физико-механическими свойствами. В результате такого подхода стало возможным создание материалов (композитов), наилучшим образом приспособленных для работы в определенных условиях, способных удовлетворять именно тем требованиям, которые предъявляются к ним при эксплуатации.

Наличие возможности управления параметрами, влияющими на свойства композитов, в процессе их проектирования и производства определяет актуальность задач, связанных с оптимальным выбором таких параметров. Среди факторов, оказывающих влияние на механические свойства композитов, одними из наиболее важных являются механические свойства компонентов материала, параметры структуры армирования, образованной компонентами, и технологические параметры. Задачи оптимизации композитов являются, как правило, многопараметрическими, а их решение в общем случае осложняется необходимостью моделирования механического поведения материала в процессе всего его жизненного цикла: от создания до практического применения.

В последние годы при решении сложных (многопараметрических, многоэкстремальных) задач оптимизации стали применяться эволюционные подходы [1], в частности, широкую известность приобрели генетические алгоритмы (ГА). Такие методы решения задач оптимизации в той или иной степени основаны на имитации эволюционных процессов в биологических популяциях, подчиняющихся закону естественного отбора.

В работе рассматривается возможность использования генетических алгоритмов при оптимизации параметров структуры армирования волокнистого композита. Для оценки качества материала выбраны характеристики его жесткости и живучести.

Полагается, что элементы армирования (волокна) имеют одинаковый диаметр, а одноосное нагружение ведется в плоскости, нормальной к направлению армирования материала. Подобный вид нагружения для волокнистых композитов является наиболее опасным, поскольку может приводить к расслоению материала. В качестве управляемых параметров выбраны объемное содержание волокон и тип структуры армирования. Тип структуры определяется по наличию или отсутствию порядка в расположении волокон в объеме композита и может принимать одно из двух значений: регулярное или стохастическое армирование.

Поскольку при стохастическом армировании все механические характеристики композита являются случайными величинами, критерий оптимальности записывается относительно параметров, характеризующих распределение локальных значений модуля упругости (математическое ожидание и интерквартильная широта) [2]. При этом математическое ожидание представляет собой эффективный модуль упругости, а интерквартильная широта распределения может рассматриваться в качестве характеристики живучести материала, поскольку ее уменьшение свидетельствует о возрастании доли однотипных локальных конфигураций структурных элементов, что, в свою очередь, ведет к проявлению свойства равнопрочности у элементарных объемов материала и, как следствие, к снижению его живучести.

Для определения параметров распределения локальных значений модуля упругости формируется представительная выборка таких значений. Каждый элемент выборки представляет собой результат решения задачи определения механического отклика элементарного объема композита на внешнее контрольное нагружение. В математическом отношении данная задача описывается системой уравнений теории упругости, в которую входят уравнения равновесия, уравнения Коши и уравнения состояния. Задача решается численно с явным учетом конфигурации структурных элементов композита в пределах выбранного элементарного объема [3]. На внешних границах объема ставятся граничные условия в смешанном виде с учетом выбранного типа контрольного нагружения, на внутренних границах (границах раздела компонентов) используются условия равенства компонент вектора перемещений (условия склейки). Решение этой задачи позволяет найти поле компонент тензоров напряжений и деформаций в пределах эле-

ментарного объема композита и вычислить локальное значение эффективного модуля упругости материала в точке, отождествляемой с выбранным конкретным элементарным объемом. Повторение данной процедуры для точек, выбираемых с использованием генераторов случайных положений в пределах представительного объема материала, позволяет сформировать выборку локальных значений модуля упругости. Оценка представительности выборки выполняется по мере ее формирования. Полученная таким образом выборка хорошо описывается вейбулловским законом распределения.

Решение задач оптимизации с использованием эволюционных подходов сопряжено с необходимостью многократных вычислений значений целевой функции. В рассматриваемой задаче целевая функция определялась относительно параметров жесткости и живучести композита, и ее вычисление для каждого набора управляемых параметров (как это было показано выше) представляет собой решение крайне трудоемкой в вычислительном отношении задачи о нахождении эффективных механических свойств композита со стохастической структурой армирования.

Для повышения эффективности алгоритма нахождения оптимального решения реальный объект исследования (математическая модель композита со стохастической структурой) заменялся упрощенной моделью (метамоделью). В качестве метамодели использовалась полиномиальная кусочно-непрерывная аппроксимация целевой функции исходной задачи. Для построения такой аппроксимации значения целевой функции находились для нескольких наперед заданных комбинаций управляемых параметров (объемного содержания волокон для каждого типа структуры). Построенная метамодель использовалась в дальнейшем для вычисления значений целевой функции в рамках генетического алгоритма поиска оптимального решения.

Для нахождения комбинации управляемых параметров, при которой целевая функция достигает максимума, использовался один из простейших вариантов ГА с единственной популяцией решений. Каждое решение в популяции кодировалось битовой строкой, одна часть которой содержала информацию об объемном содержании волокон в композите, другая – о типе структуры армирования. На каждой итерации для популяции решений выполнялись обычные «генетические» процедуры: отбор (методом турнирной селекции), скрещивание кодов отобранных решений, мутация. Результатом выполнения данных процедур являлась замена исходной популяции решений новой популяцией («смена поколений»). Для каждого нового «поколения» проводилась оценка сходимости решения. В качестве критерия сходимости

использовалась величина максимального отклонения объемного содержания волокон в субпопуляции для каждого из типов структуры армирования от его среднего значения в субпопуляции. Оценка качества каждого решения в популяции выполнялась в рамках параллельного алгоритма с использованием стандарта MPI.

Описанный алгоритм использовался для поиска параметров структуры армирования композита TiNi-TiC, обеспечивающих ему максимальные показатели жесткости и живучести, в интервале объемного содержания армирующих элементов от 10 до 50%. Полученные результаты показали, что алгоритм обладает высокой скоростью сходимости уже при сравнительно малых объемах популяции решений. При этом увеличение объема популяции сопровождается увеличением количества итераций («поколений»), необходимых для достижения сходимости. Данное обстоятельство означает, что при решении подобных задач может быть найден некоторый «критический» объем популяции, обеспечивающий нахождение оптимального решения, превышение которого приводит лишь к увеличению вычислительных затрат, снижающих эффективность алгоритма. Показано, что в рамках принятых допущений оптимальной с точки зрения повышения жесткости и живучести композита будет стохастическая структура армирования с максимально высоким объемным содержанием включений (до 50%).

Литература

1. **Гладков Л.А., Курейчик В.В., Курейчик В.М.** Генетические алгоритмы. М.: Физматлит, 2006. 320 с.
2. **Кремер Н.М.** Теория вероятностей и математическая статистика: Учеб. для вузов. М.: ЮНИТИ-ДАНА, 2006. 573 с.
3. **Сидоренко Ю.Н.** Прогнозирование механических свойств стохастических композиционных материалов: Дисс. … канд. физ.-мат. наук. Томск, 2004. 142 с.

Новый тип эрмитовых сплайн-вейвлетов пятой степени*

Д.А. Турсунов, Б.М. Шумилов, Э.А. Эшаров, Э.А. Турсунов
Ошский государственный университет, Киргизия
Томский государственный архитектурно-строительный университет

Новый тип вейвлетов построен в базисе эрмитовых сплайнов пятой степени. Носителем материнских вейвлетов является отрезок [–1, 1]*, при этом один из них антисимметричный, а два других – симметричные.*

Введение

Вейвлеты используют в тех случаях, когда результат анализа некоего сигнала должен содержать не только перечисление его характерных частот (масштабов), но и сведения об определенных локальных координатах, при которых эти частоты проявляют себя. Это бывает важно при анализе и обработке нестационарных (во времени) или неоднородных (в пространстве) сигналов. Общий принцип построения базиса вейвлет-преобразования состоит в использовании масштабного преобразования и смещений. Именно за счет изменения масштабов вейвлеты способны выявить различие в характеристиках на разных шкалах, а путем сдвига проанализировать свойства сигнала в разных точках на всем изучаемом интервале. При анализе нестационарных сигналов, за счет свойства локальности, вейвлеты получают существенное преимущество перед преобразованием Фурье.

Введем обозначения: $\langle u, v\rangle := \int_R u(x)v(x)dx$, $u(x), v(x) \in L_2(R)$ – скалярное произведение; $\|f\|_2 = \sqrt{\langle f,f\rangle}$ – норма функции f в $L_2(\mathbf{R})$; Π_k – множество многочленов степени k, (k=0, 1, 2,…); V_n – пространство сплайнов пятой степени, удовлетворяющих следующим условиям, n>0:

а) $v \in C[0, 1] \cap C'[0, 1] \cap C''[0, 1]$;

б) $v(x) \in \prod_5, x \in (j/2^n, (j+1)/2^n), j=0, \ldots, 2^n-1$; $\dim(V_n)=2^{n+1}$.

И. Добеши построила полуортогональные вейвлеты неполиномиального типа с компактным носителем [1]. Чуи и Вонг построили полуортогональные сплайн-вейвлеты [2]. Вонг построил базис кубиче-

* Работа выполнена при поддержке РФФИ, проекты №09-07-90901, №09-01-90716.

ских сплайн-вейвлетов в пространстве Соболева [3]. Донован и др. построили ортогональные мультивейвлеты [4]. Недостатками построенных ранее вейвлетов является то, что они либо не имеют аналитического представления, либо расположены на достаточно широком носителе. И то и другое бывает чрезвычайно важно при их использовании.

Пусть ϕ_1, ϕ_2 и ϕ_3 – базисные эрмитовы сплайны пятой степени (рис. 1):

$$\phi_1(t)=\begin{cases}(t+1)^3\left(1-3t+6t^2\right) & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\left(1+3t+6t^2\right) & \text{при} \quad 0\le t\le 1,\end{cases}$$

$$\phi_2(t)=\begin{cases}(t+1)^3\left(t-3t^2\right) & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\left(t+3t^2\right) & \text{при} \quad 0\le t\le 1,\end{cases} \qquad \phi_3(t)=\begin{cases}(t+1)^3\dfrac{t^2}{2} & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\dfrac{t^2}{2} & \text{при} \quad 0\le t\le 1.\end{cases}$$

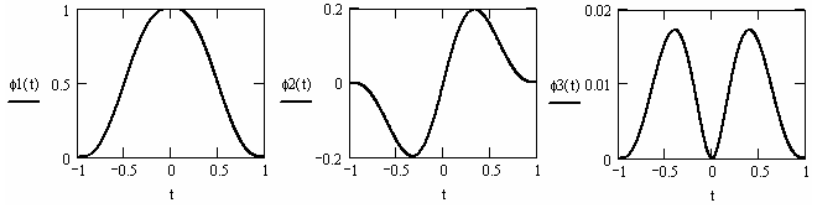


Рис. 1. Графики базисных эрмитовых сплайнов пятой степени: ϕ_1, ϕ_2 и ϕ_3

Дамен и др. построили биортогональные мультивейвлеты в базисе эрмитовых кубических сплайнов [5]. Отметим, что их конструкция базисных вейвлетов выглядит слишком сложно. В [6] описаны «ленивые» ортонормальные кубические эрмитовы сплайн-вейвлеты на суперкомпактном носителе [–1, 1], однако свойство полуортогональности для них не выполняется. В [7] было доказано, что полуортогональные кубические мультивейвлеты на суперкомпактном носителе можно построить относительно скалярного произведения $\langle u', v'\rangle$. Используя эту же идею, мы построили базисные вейвлеты на пространстве эрмитовых сплайнов пятой степени. В отличие от классических вейвлетов Чуи и Вонга, эти вейвлеты имеют меньший носитель, а в отличие от ранее построенных вейвлетов, они обладают свойством ортогональности относительно скалярного произведения $\langle u'', v''\rangle$.

Сплайн-вейвлеты

Пусть S представляет собой инвариантное пространство сдвигов, порожденное ϕ_1, ϕ_2 и ϕ_3. Функция g принадлежит пространству S тогда

и только тогда, когда существуют последовательности b_1, b_2 и b_3 на $\mathbf{Z}$, для которых выполняется равенство

$$g = \sum_{j\in Z}\left[b_1(j)\varphi_1(\cdot - j) + b_2(j)\varphi_2(\cdot - j) + b_3(j)\varphi_3(\cdot - j)\right]\cdot$$

Пусть $S_1=\{g(2\cdot)\colon g\in S\}$, тогда $S\subset S_1$. Мы потребуем выполнение равенств

$$\langle \psi''_1,\ \phi''_m(-j)\rangle = \langle \psi''_2,\ \phi''_m(-j)\rangle = \langle \psi''_3,\ \phi''_m(-j)\rangle = 0,\ m=1,2,3\ \ \forall j\in\mathbf{Z}. \qquad (1)$$

Здесь

$$\varphi_1''(t) = \begin{cases} 60t + 180t^2 + 120t^3 & \text{при } -1\le t\le 0,\\ -60t + 180t^2 - 120t^3 & \text{при } 0\le t\le 1,\end{cases}$$

$$\varphi_2''(t) = \begin{cases} -36t - 96t^2 - 60t^3 & \text{при } -1\le t\le 0,\\ -36t + 96t^2 - 60t^3 & \text{при } 0\le t\le 1,\end{cases} \quad \varphi_3''(t) = \begin{cases} 1 + 9t + 18t^2 + 10t^3 & \text{при } -1\le t\le 0,\\ 1 - 9t + 18t^2 - 10t^3 & \text{при } 0\le t\le 1.\end{cases}$$

Предположим, что

$$\psi(x) = \sum_{k\in Z}\left[b_1(k)\varphi_1(2x-k) + b_2(k)\varphi_2(2x-k) + b_3(k)\varphi_3(2x-k)\right],\ x\in R\cdot$$

Тогда для $\forall j\in\mathbf{Z}$ имеем:

$$\langle \psi'', \varphi_1''(-j)\rangle = -15b_1(2j-2) + 30b_1(2j) - 15b_1(2j+2) - 39b_2(2j-2) + 48b_2(2j-1) - 48b_2(2j+1) + 39b_2(2j+2) - 3b_3(2j-2) + 6b_3(2j) - 3b_1(2j+2);$$

$$\langle \psi'', \varphi_2''(-j)\rangle = 720b_1(2j-2) - 840b_1(2j-1) + 840b_1(2j+1) - 20b_1(2j+2) + 696b_2(2j-2) - 960b_2(2j-1) + 1728b_2(2j) - 960b_2(2j+1) + 696b_2(2j+2) + 53b_3(2j-2) - 14b_3(2j-1) + 14b_3(2j+1) - 53b_3(2j+2);$$

$$\langle \psi'', \varphi_3''(-j)\rangle = -240b_1(2j-2) + 840b_1(2j-1) - 1200\, b_1(2j) + 840b_1(2j+1) - 240b_1(2j+2) - 176b_2(2j-2) + 320b_2(2j-1) - 320b_2(2j+1) + 176b_2(2j+2) - 13b_3(2j-2) + 14b_3(2j-1) + 54b_3(2j) + 14b_3(2j+1) - 13b_3(2j+2).$$

Введем преобразование Лапласа последовательностей b_1, b_2 и b_3 на $\mathbf{Z}$, так что положительная или отрицательная степень z означает сдвиг на один шаг по шаблону узлов эрмитового сплайна вправо или влево соответственно. Тогда

$$q_{11}(z) = \sum_{j\in Z} b_1(2j+1)z^{2j+1},\ q_{12}(z) = \sum_{j\in Z} b_1(2j)z^{2j},$$

$$q_{21}(z) = \sum_{j\in Z} b_2(2j+1)z^{2j+1},\ q_{22}(z) = \sum_{j\in Z} b_2(2j)z^{2j},$$

$$q_{31}(z) = \sum_{j\in Z} b_3(2j+1)z^{2j+1},\ q_{32}(z) = \sum_{j\in Z} b_3(2j)z^{2j}\ .$$

Решение однородного функционального уравнения

$$Q(z)(q_{11}(z),\ q_{12}(z),\ q_{21}(z),\ q_{22}(z))^{\mathrm{T}} = 0,\ z\in C\setminus\{0\},\ j\in\mathbf{Z},$$

$$Q(z)=\begin{Bmatrix} 0 & -15z^2+30-15z^{-2} & 48z-48z^{-1} \\ -840z+840z^{-1} & 720z^2-720z^{-2} & -960z-960z^{-1} \\ 840z+840z^{-1} & -240z^2-1200-240z^{-2} & 320z-320z^{-1} \end{Bmatrix}$$

$$\begin{matrix} -39z^2+39z^{-2} & 0 & -3z^2+6-3z^{-2} \\ 696z^2+1728+696z^{-2} & -14z+14z^{-1} & 53z^2-53z^{-2} \\ -176z^2+176z^{-2} & 14z+14z^{-1} & -13z^2+54-13z^{-2} \end{matrix}\Bigg\}$$

будет означать, что $\langle \psi'', \phi''_m(-j)\rangle=0$, m=1,2,3.

Отсюда находим три независимых решения:

$q_{11}(z)=z^{-1}+z$, $q_{12}(z)=0$, $q_{21}(z)=0$, $q_{22}(z)=0$, $q_{31}(z)=-60z^{-1}-60z$, $q_{32}(z)=0$;

$q_{11}(z)=2(z^{-1}-z)$, $q_{12}(z)=0$, $q_{21}(z)=13(z^{-1}-z)$, $q_{22}(z)=16$, $q_{31}(z)=-24(z^{-1}+z)$, $q_{32}(z)=0$;

$q_{11}(z)=z^{-1}+z$, $q_{12}(z)=0$, $q_{21}(z)=-z^{-1}+z$, $q_{22}(z)=0$, $q_{31}(z)=-68(z^{-1}+z)$, $q_{32}(z)=16$.

Эти три решения порождают три материнских вейвлета ψ_1, ψ_2, ψ_3:

$$\psi_1(x)=\phi_1(2x+1)-\phi_1(2x-1)-60\phi_3(2x+1)-60\phi_3(2x-1),\ \psi_2(x)=2\phi_1(2x+1)-$$
$$-2\phi_1(2x-1)+13\phi_2(2x+1)+13\phi_2(2x-1)+16\phi_2(2x)-24\phi_3(2x+1)+24\phi_3(2x-1),$$
$$\psi_3(x)=\phi_1(2x+1)+\phi_1(2x-1)-\phi_2(2x+1)+\phi_2(2x-1)-68\phi_3(2x+1)-68\phi_3(2x-1)+16\phi_3(2x).$$

Носителем ψ_1, ψ_2, ψ_3 является отрезок [–1, 1]. Они удовлетворяют условию (1). При этом ψ_1 – антисимметричен, а ψ_2, ψ_3 – симметричны (рис. 2).

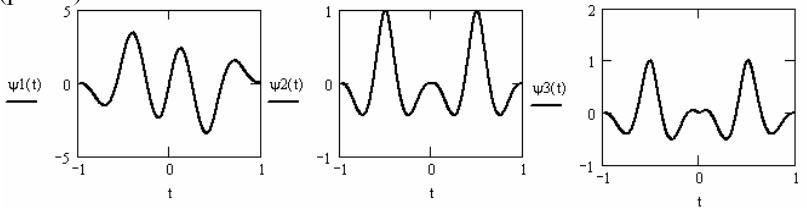


Рис. 2. Графики эрмитовых сплайн-вейвлетов пятой степени

Вывод

Полученные результаты позволяют высказать гипотезу о том, что для построения эрмитовых сплайн-вейвлетов ($2n$+1)-й степени с суперкомпактным носителем достаточно требовать полуортогональность со скалярным произведением $<u^{(n)}, v^{(n)}>$.

Литература

1. **Добеши И.** Десять лекций по вейвлетам. Москва; Ижевск: РХД, 2001.
2. **Chui C.K., Wang J.Z.** On compactly supported wavelets and a duality principle// Trans. Amer. Math. Soc. 1992. Vol. 330. P. 903-916.

3. **Wang J.Z.** Cubic spline wavelet bases of Sobolev spaces and multilevel interpolation // Appl. Comput. Harmonic Anal. 1996. Vol. 3. P. 154-163.
4. **Donovan G., Geronimo J.S., Hardin D.P., Massopust P.R.** Construction of orthogonal wavelets using fractal interpolation functions // SIAM J. Math. Anal. 1996. Vol. 27. P. 1158-1192.
5. **Dahmen W., Jia Han R.Q. and Kunoth A.** Biorthogonal multiwavelets on the interval: Cubic Hermite splines // Constr. Approx. 2000. Vol. 16. P. 221-259.
6. **Шумилов Б.М., Эшаров Э.А.** Построение эрмитовых сплайн-вейвлетов // Вестник ТГУ. Сер. Математика. Кибернетика. Информатика. Приложение. 2006. № 19. С. 260-266.
7. **Jia R.-Q., Liu S.-T.** Wavelet bases of Hermite cubic splines on the interval // Advances Computational Mathematics. 2006. Vol. 25. P. 23-39.

Схема «предиктор – корректор» с заданными свойствами для одномерных изотермических уравнений газовой динамики

С.В. Белов, В.В. Жаровцев
Научно-исследовательский институт прикладной математики и механики, Томск

Кроме аппроксимации и устойчивости на разностные уравнения и сетку накладываются дополнительные требования: минимальный сеточный шаблон, неотрицательная аппроксимация, минимальная аппроксимационная вязкость, повышенная устойчивость.

Систему нестационарных уравнений идеального изотермического газа с одной пространственной переменной запишем в виде

$$\varphi_t + \left(u\varphi + \psi\right)_x = 0\ ,\ \varphi = \left(\rho, \rho u\right)^T,\ \ \psi = \left(0, p\right)^T,\ \ p = c^2\rho\ , \quad (1)$$

где t – время; x – декартова координата; ρ – плотность; u – скорость; p – давление; c – изотермическая скорость звука; T – знак транспонирования. Для простоты изложения полагаем, что область интегрирования системы (1) есть прямоугольник, в котором вводится равномерная по пространственной координате и не равномерная по времени прямоугольная сетка с шагами $\Delta x = x_{i+0.5} - x_{i-0.5}$, $(i = 0,1,...,N)$, где N – число ячеек сетки по пространственной координате, и $\Delta t^{n+0,5} = t^{n+1} - t^n$ ($n = 0,1,\ldots,n_k$). Значения параметров газа определяются в центрах ячеек, а также на границах между соседними ячейками сетки. Используя обозначения $F(g^l_k, h^l_k, \ldots, q^l_k) = F^l_k$, где F – некоторая функция своих аргументов, разностный аналог системы (1) запишем так:

$$\varphi^{n+1}_{i+0,5} = \varphi^n_{i+0,5} - \Delta t^{n+0,5}\left[\left(\varphi\, u + \psi\right)^*_{i+1} - \left(\varphi\, u + \psi\right)^*_i\right] \Big/ \Delta x\ . \quad (2)$$

Вычисления по формулам (2) можно производить после того, как указан алгоритм нахождения как шага по времени, так и параметров газа на границах между соседними ячейками разностной сетки. Особый интерес для практики представляют разностные схемы, использующие минимальный сеточный шаблон $\left(t^{n+a}, x_{i+0,5+b}\right)$, $a = 0,1\,;\, b = -1,0,1$, поскольку в таких случаях в приграничных и внутренних точках сетки параметры газа рассчитываются по одним и

тем же формулам. Кроме этого, известно, что в ряде случаев за счет простоты и эффективности алгоритмов распараллеливания явные разностные схемы не уступают неявным. Все это учитывается при построении формул для расчета параметров газа на предварительном этапе. Для этого систему (1) перепишем в нескольких эквивалентных видах. Так, уравнение неразрывности эквивалентно каждому из соотношений

$$\rho_t + u\rho_x + \rho u_x = 0\,,\ \left(\ln\rho\right)_t + u\rho^{-1}\rho_x + u_x = 0\,,\ \left(\ln\rho\right)_t + u\left(\ln\rho\right)_x + u_x = 0\,, \quad (3)$$

а вместо закона сохранения импульса можно использовать любое из уравнений

$$\left(\rho u\right)_t + u\left(\rho u\right)_x + \left(\rho u\right)u_x + p_x = 0\,,\ \left(u\right)_t + u\left(u\right)_x + \rho^{-1}\left(p\right)_x = 0\,. \quad (4)$$

Привлекая подход, предложенный в [1] для определения ρ_i^* и u_i^*, построим разностные аналоги соответствующих уравнений из (3) и (4).

$$\rho_i^* = \rho_i^n - 0.5\left(\alpha_1+\beta_1\right)\left(\rho_{i+0,5}^n - \rho_{i-0,5}^n\right) - 0.5\left(\alpha_2-\beta_2\right)\rho_i^n\left(u_{i+0,5}^n - u_{i-0,5}^n\right)\Big/c_i^n\,,$$

$$\ln\left(\rho_i^*/\rho_i^n\right) = -0.5\left(\alpha_1+\beta_1\right)\left(\rho_{i+0,5}^n - \rho_{i-0,5}^n\right)\Big/\rho_i^n - 0.5\left(\alpha_2-\beta_2\right)\left(u_{i+0,5}^n - u_{i-0,5}^n\right)\Big/c_i^n\,,$$

$$\ln\left(\rho_i^*/\rho_i^n\right) = -0.5\left(\alpha_1+\beta_1\right)\ln\left(\rho_{i+0,5}^n/\rho_{i-0,5}^n\right) - 0.5\left(\alpha_2-\beta_2\right)\left(u_{i+0,5}^n - u_{i-0,5}^n\right)\Big/c_i^n\,, \quad (5)$$

$$\left(\rho u\right)_i^* = \left(\rho u\right)_i^n - 0.5\left(\alpha_3+\beta_3\right)\Big(\left(\rho u\right)_{i+0,5}^n - \left(\rho u\right)_{i-0,5}^n\Big) -$$

$$-\left(\alpha_4-\beta_4\right)\Big(p_{i+0,5}^n - p_{i-0,5}^n + \left(\rho u\right)_i^n\left(u_{i+0,5}^n - u_{i-0,5}^n\right)\Big)\Big/\left(2c_i^n\right),$$

$$u_i^* = u_i^n - 0.5\left(\alpha_3+\beta_3\right)\left(u_{i+0,5}^n - u_{i-0,5}^n\right) - 0.5\left(\alpha_4-\beta_4\right)\left(p_{i+0,5}^n - p_{i-0,5}^n\right)\Big/\left(\rho c\right)_i^n\,,$$

$$\alpha_{i,n}^k = \Delta t_k\left(u_i^n + c_i^n\right)\Big/\Delta x\,,\ \beta_{i,n}^k = \delta t_k\left(u_i^n - c_i^n\right)\Big/\Delta x\,,\ \omega_i^n = 0.5\left(\omega_{i+0.5}^n + \omega_{i-0.5}^n\right),\ \omega = (\rho, c, u)^T\,.$$

Число неповторяющихся сочетаний первых трех и последних двух разностных уравнений равно шести, поскольку из-за «рассогласованности» своих членов разностные уравнения в этих группах не эквивалентны друг другу. Таким образом, на минимальном сеточном шаблоне записано шесть семейств разностных схем (2), (5), аппроксимирующих систему (1) с первым порядком и зависящих от параметров Δt_j и δt_j $\left(j = \overline{1,4}\right)$.

Чтобы определить значения этих параметров, рассмотрим линейное приближение системы (1). У линеаризованных уравнений постоянные параметры отметим индексом «0». Введем новую функцию $\bar{\rho} = \rho c_0/\rho_0$. Тогда соответствующая системе (1) линейная система имеет вид

$$U_t + A U_x = 0\ ,\ A=\begin{pmatrix} u_0 & c_0 \\ c_0 & u_0 \end{pmatrix},\ U = (\overline{\rho}, u)^T\ , \qquad (6)$$

а единственное четырехпараметрическое семейство разностных схем для системы (6) можно записать так:

$$U_{i+0,5}^{n+1} = U_{i+0,5}^{n} - \frac{\Delta t}{\Delta x} A\left(U_{i+0,5}^{*} - U_{i-0,5}^{*}\right),\ U_i^{*} = U_i^n - B\left(U_{i+0,5}^{n} - U_{i-0,5}^{n}\right),$$

(7)

$$B = \frac{1}{2}\begin{pmatrix} \alpha_1 + \beta_1 & \alpha_2 - \beta_2 \\ \alpha_3 - \beta_3 & \alpha_4 + \beta_4 \end{pmatrix},\ \alpha_j = \frac{\Delta t_j}{\Delta x}\left(u_0 + c_0\right)\ \ \beta_j = \frac{\delta t_j}{\Delta x}\left(u_0 - c_0\right).$$

Подставив второе уравнение из (7) в первое, получим соотношения

$$U_i^{n+1} = D_{-1} U_{i-1}^n + D_0 U_i^n + D_1 U_{i+1}^n,\ D_{-1,1} = \frac{\Delta t}{\Delta x} AB \pm \frac{\Delta t}{2\Delta x} A\ ,\ D_0 = E - 2\frac{\Delta t}{\Delta x} AB\ , \qquad (8)$$

где E – единичная матрица. Для выделения схем с неотрицательной аппроксимацией найдем собственные значения матриц $D_{-1,1}$, D_0. В явном виде их удалось выписать, когда

$$\Delta t_4 = \Delta t_1\ ,\ \Delta t_3 = \Delta t_2\ ,\ \delta t_4 = \delta t_1\ ,\ \delta t_3 = \delta t_2\ ,\ \Delta t_2 = \Delta t_1\ ,\ \delta t_2 = \delta t_1\ . \qquad (9)$$

Теперь свободными остались два параметра. В результате достаточно громоздких выкладок имеем, что при выполнении условий

$$2\Delta t_1 \left|u_0 + c_0\right| \ge \Delta x\ ,\ 2\delta t_1 \left|u_0 - c_0\right| \ge \Delta x\ ,$$

$$\Delta t\left((u_0 + c_0)^2 (2\,\Delta t_1)/\Delta x\right) \le \Delta x,\quad \Delta t\left((u_0 - c_0)^2 (2\,\delta\, t_1)/\Delta x\right) \le \Delta x \qquad (10)$$

все собственные значения матриц $D_{-1,1}$, D_0 неотрицательны. То есть из восьмипараметрического семейства (7) выделено двухпараметрическое семейство разностных схем (параметры Δt_1, , δt_1) с неотрицательно определенными матрицами. В силу выполнения условий теоремы Фридрихса соотношения (9), (10) являются достаточными условиями устойчивости схем (7).

Для оставшегося двухпараметрического семейства схем первое дифференциальное приближение имеет вид

$$U_t + A U_x = \Delta x C U_{xx}\ ,\ C = A\left(B - A\Delta t/(2\Delta x)\right), \qquad (11)$$

где матрица C определяет величину аппроксимационной вязкости. Можно показать, что ее норма достигает своего минимума, когда

$$\Delta t_1 \left|u_0 + c_0\right| / \Delta x = 0.5\ ,\ \delta t_1 \left|u_0 - c_0\right| / \Delta x = 0.5\ . \qquad (12)$$

Тогда из (10) следует, что $\kappa \le 1$, где $\kappa = \Delta t\left(\left|u_0\right| + c_0\right)/\Delta x$ – число Куранта.

Операторы согласованного сглаживания для (2), (5) по верхнему и нижнему слоям запишем так [2]:

$$\overline{\varphi}^{\,n+1}_{i+0,5}=\left(P+F_n\right)\varphi^{\,n}_{i+0,5},\ \varphi^{\,n+1}_{i+0,5}=\left(I+F_b\right)\overline{\varphi}^{\,n+1}_{i+0,5},$$

$$F_n\varphi^{\,n}_{i+0,5}=\theta_n\left(f^n_{i+1}-f^n_i\right),\ F_b\overline{\varphi}^{\,n+1}_{i+0,5}=\theta_b\left(\overline{f}^{n+1}_{i+1}-\overline{f}^{n+1}_{i+1}\right),\ \Delta\rho_i=\rho_{i+0,5}-\rho_{i-0,5},$$

$$f_i=\begin{cases}\left(\varphi_{i+0,5}-\varphi_{i+0,5}\right), & \left(\Delta\rho_i\Delta\rho_{i+1}\right)<0\vee\left(\Delta\rho_i\Delta\rho_{i-1}\right)<0,\\ 0 \quad в \quad противоположном \quad случае,\end{cases}$$

где I – единичный оператор; P – один из операторов (2), (5). θ_n и θ_b – параметры сглаживания. Для схемы (7), (9), (12) удалось показать, что когда $\theta_n=3\sqrt{2}/2-2\approx 0{,}1213$, $\theta_b=\left(4-\sqrt{2}\right)/14\approx 0{,}1847$, то $\kappa\le 4-2\sqrt{2}\approx 1{,}171$.

Все отмеченные выше дополнительные свойства разностной схемы (2), (5) (первое и четвертое уравнения) были проверены на решении тестовой задачи, в качестве которой была выбрана задача о распаде произвольного разрыва, решение которой определяется давлением слева от разрыва. На рис. 1 и 2 представлены профили давлений в фиксированные моменты времени (сплошные кривые соответствуют точному решению, а штриховые – численному, когда $N=1000$). Численные результаты на рис. 1 соответствует расчетам, когда $\theta_n=\theta_b=0$, $\kappa=1$, а на рис. 2 – когда $\theta_n=0{,}1213$, $\theta_b=0{,}1847$, $\kappa=12/10$.

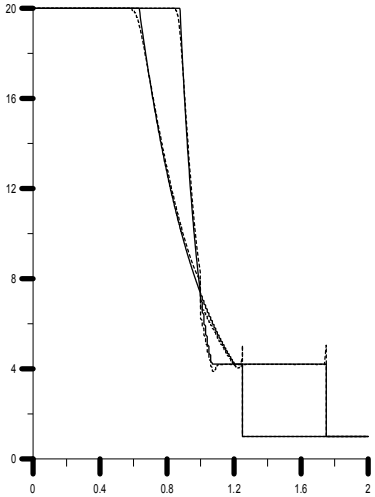


Рис. 1

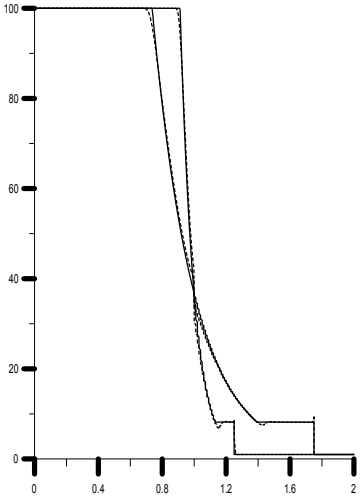


Рис. 2

Литература

1. **Жаровцев В.В.** Явные схемы с неотрицательной аппроксимацией на минимальном сеточном шаблоне для одномерных уравнений газовой динамики //Вычислительная гидродинамика. Томск: Изд-во Том. ун-та, 1999. С. 84-88.
2. **Жмакин А.И., Фурсенко А.А.** Об одной монотонной разностной схеме сквозного счета // ЖВММФ. 1980. Т. 20, № 4. С. 1021-1031.

Обоснование корректности полинейного рекуррентного метода решения разностных эллиптических уравнений

А.А. Фомин, Л.Н. Фомина
Кемеровский государственный университет

Рассматривается матрично-векторная форма записи алгоритма итерационного полинейного рекуррентного метода решения пятидиагональных систем линейных алгебраических уравнений с матрицами положительного типа. Путем пошаговых устойчивых преобразований выводятся общий вид и структура результирующего оператора. Приводится обоснование корректности метода.

Как известно, решение краевых задач тепло- и массопереноса связано с разностной дискретизацией их исходных дифференциальных постановок, что, в свою очередь, приводит к возникновению систем линейных алгебраических уравнений (СЛАУ) вида $A\vec{\Phi} = \vec{f}$. Здесь A – матрица системы; $\vec{\Phi}$ – искомый вектор решения; $\vec{f}$ – вектор правой части системы. Подобные матрицы обладают высоким порядком и ленточной структурой [1]. Для многомерных задач полученные таким образом СЛАУ разрешаются, как правило, итерационными методами.

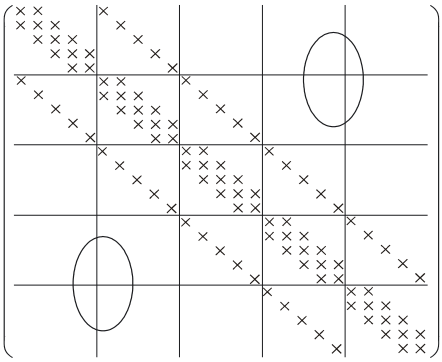


Рис. 1. Структура матрицы A

В работах [2, 3] излагается алгоритм и рассматриваются результаты тестовых расчетов итерационного полинейного рекуррентного метода для случаев, когда A – матрица положительного типа. Данный метод

демонстрирует свою высокую эффективность, однако в силу того, что в общем случае записать его явно в канонической форме $P_{k+1}(\vec{\Phi}^{k+1}-\vec{\Phi}^{k})=\vec{f}-A\vec{\Phi}^{k}$ [1] не представляется возможным, корректность каждого расчета необходимо фактически доказывать вычислением нормы невязки для очередного приближения решения. Само по себе выполнить такое вычисление не сложно, более того, именно по величине нормы невязки и определяется сходимость вектора решения. Однако в целях теоретической завершенности изложения этого метода представляется необходимым в общем случае обосновать его корректность, т.е. показать, что в случае выполнения условия $\left\|\vec{\Phi}^{k+1}-\vec{\Phi}^{k}\right\|_{k\to\infty}\to 0$ полученное решение удовлетворяет исходной системе линейных алгебраических уравнений.

Для двумерных задач матрица *A* имеет пятидиагональную структуру (рис. 1). Общее число неизвестных и, следовательно, размер матрицы *A* равны числу $N=n\times m$. Как известно, такая матрица может быть разбита на отдельные квадратные клетки, количество которых, для определенности, будет $n\times n$, тогда размер каждой клетки – $m\times m$. Здесь и далее для наглядности и не в ущерб общности изложения материала выбрана разностная сетка 5×5. Понятно, что в рамках рассматриваемого случая только клетки на диагонали и околодиагональные будут отличны от нуля, все остальные – нулевые. Поклеточная структура матрицы *A* и состав двух клеток (для примера) имеют вид

$$A=\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15}\\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25}\\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35}\\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45}\\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} \end{pmatrix},\quad A_{11}=\begin{pmatrix} a_{P_{11}}-a_{N_{11}} & 0 & 0 & 0\\ -a_{S_{12}} & a_{P_{12}}-a_{N_{12}} & 0 & 0\\ 0 & -a_{S_{13}} & a_{P_{13}}-a_{N_{13}} & 0\\ 0 & 0 & -a_{S_{14}} & a_{P_{14}}-a_{N_{14}}\\ 0 & 0 & 0 & -a_{S_{15}} & a_{P_{15}} \end{pmatrix},\quad A_{12}=\begin{pmatrix} -a_{E_{11}} & 0 & 0 & 0 & 0\\ 0 & -a_{E_{12}} & 0 & 0 & 0\\ 0 & 0 & -a_{E_{13}} & 0 & 0\\ 0 & 0 & 0 & -a_{E_{14}} & 0\\ 0 & 0 & 0 & 0 & -a_{E_{15}} \end{pmatrix}$$

Первые два шага полинейного рекуррентного метода представляют собой воздействие на исходную систему двумя невырожденными операторами: $M^{(12)}_{(+)}M^{(11)}_{(+)}(A\vec{\Phi}-\vec{f})=0$. При этом только диагональные клетки матриц операторов $M^{(11)}_{(+)}$ и $M^{(12)}_{(+)}$ отличны от нуля, из них первые клетки в верхнем ряду имеют следующий вид:

$$M_{11(+)}^{(11)}=\begin{pmatrix}1 & 0 & 0 & 0 & 0\\ a_{S_{12}}/a_{P_{11}} & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\end{pmatrix},\quad M_{11(+)}^{(12)}=\begin{pmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & a_{S_{13}}/\alpha_{P_{12}} & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\end{pmatrix},$$

остальные диагональные клетки – единичные матрицы. Следовательно, клетки произведения имеют следующую структуру:

$$\left(M_{(+)}^{(12)}M_{(+)}^{(11)}A\right)_{11}=\begin{pmatrix}\times\times 0 0 0\\ 0\times\times 0 0\\ 0 0\times\times 0\\ 0 0\times\times\times\\ 0 0 0\times\times\end{pmatrix},\quad \left(M_{(+)}^{(12)}M_{(+)}^{(11)}A\right)_{12}=\begin{pmatrix}\times 0 0 0 0\\ \times\times 0 0 0\\ \times\times\times 0 0\\ 0 0 0\times 0\\ 0 0 0 0\times\end{pmatrix}.$$

Пусть $M_{(+)}^{(12)}M_{(+)}^{(11)}A=G_{(+)}^{(12)}+L_{(+)}^{(12)}$, причем матрица $L_{(+)}^{(12)}$ состоит из нулей, кроме второй клетки в верхнем ряду, которая имеет всего лишь один ненулевой элемент – коэффициент «внешаблонного» узла [3]:

$L_{12(+)}^{(12)}=\begin{pmatrix}0 0 0 0 0\\ 0 0 0 0 0\\ \times 0 0 0 0\\ 0 0 0 0 0\\ 0 0 0 0 0\end{pmatrix}$, а матрица $G_{(+)}^{(12)}$ содержит все остальные элементы матрицы $M_{(+)}^{(12)}M_{(+)}^{(11)}A$. Используя данное представление, исходная система перепишется в виде $G_{(+)}^{(12)}\vec{\Phi}+L_{(+)}^{(12)}\vec{\Phi}=\left[\prod_{j=2}^{1}M_{(+)}^{(1j)}\right]\vec{f}$.

При помощи одного из способов аппроксимации значения искомой функции во «внешаблонном» узле производится замена члена, содержавшего его: $L_{(+)}^{(12)}\vec{\Phi}\to L_{(+)}^{(12)}\vec{\Phi}^{(k)}+\theta B_{(+)}^{(12)}\Delta\vec{\Phi}^{(k+1)}$, - при этом алгоритм принимает итерационный характер. После проведения подобных преобразований на первой расчетной линии (верхний ряд клеток) исходное уравнение можно переписать в виде

$$\mathcal{G}^{(2)}\vec{\Phi}^{(k+1)}+\theta\mathcal{B}^{(2)}\Delta\vec{\Phi}^{(k+1)}+\mathcal{L}^{(2)}\vec{\Phi}^{(k)}=\mathcal{M}^{(2)}\vec{f},$$

где эквивалентные преобразования группируются в матрицах $\mathcal{G}^{(2)}$, $\mathcal{L}^{(2)}$, $\mathcal{M}^{(2)}$, а приближенные – только в матрице $\mathcal{B}^{(2)}$. Причем проводимые преобразования устойчивы, сохраняют свойство построчного диагонального преобладания и противоположность знаков диагональных и внедиагональных элементов.

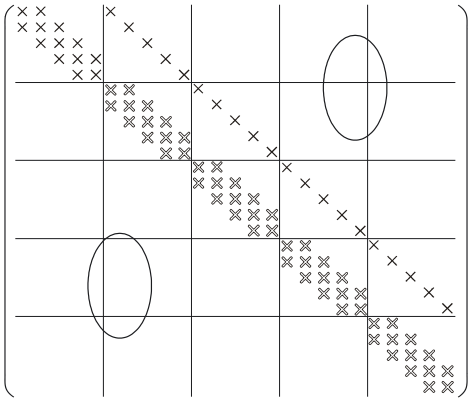


Рис. 2

В конце проходов по всем расчетным линиям (рядам клеток) исходная система примет вид $\mathcal{G}^{(n)}\vec{\Phi}^{(k+1)}+\theta\mathcal{B}^{(n)}\Delta\vec{\Phi}^{(k+1)}+\mathcal{L}^{(n)}\vec{\Phi}^{(k)}=\mathcal{M}^{(n)}\vec{f}$,

где эквивалентные преобразования содержатся в матрицах $\mathcal{G}^{(n)}$, $\mathcal{L}^{(n)}$, $\mathcal{M}^{(n)}$, а приближенные – только в матрице $\mathcal{B}^{(n)}$, причем матрица $\mathcal{G}^{(n)}+\theta\mathcal{B}^{(n)}$ по свой структуре четырехдиагональная и положительного типа (рис. 2), откуда следует разрешимость полученной системы.

В случае сходимости $\Delta\vec{\Phi}^{(k+1)}\to 0$ влияние члена, содержащего приближенные преобразования, станет несущественным. Поскольку операторы $\mathcal{G}^{(n)}$, $\mathcal{L}^{(n)}$ и $\mathcal{M}^{(n)}$ – суть чисто эквивалентные преобразования, не меняющие решение исходной системы, то путем обратной цепочки преобразований можно получить явный вид оператора воздействия на первоначальную систему уравнений.

$$\left\{E_{(0)}^{(n)}+E_{(p)}^{(n)}H^{(n)}\left[E-\left(\prod_{j=m\text{-}1}^{1}M_{(+)}^{(n-1\,j)}+\prod_{j=2}^{m}M_{(-)}^{(n-1\,j)}\right)\right]\right\}\times$$

$$\times\left\{E_{(0)}^{(n-1)}+E_{(p)}^{(n-1)}H^{(n-1)}\left[E-\left(\prod_{j=m\text{-}1}^{1}M_{(+)}^{(n-2\,j)}+\prod_{j=2}^{m}M_{(-)}^{(n-2\,j)}\right)\right]\right\}\times\ \ ...$$

$$...\ \ \times\left\{E_{(0)}^{(2)}+E_{(p)}^{(2)}H^{(2)}\left[E-\left(\prod_{j=m\text{-}1}^{1}M_{(+)}^{(1\,j)}+\prod_{j=2}^{m}M_{(-)}^{(1\,j)}\right)\right]\right\}\times\left(A\vec{\Phi}^{(*)}-\vec{f}\right)=0.$$

Понятно, что решение подобной системы является также решением исходной системы, при этом фигурные скобки представляют собой комбинацию невырожденных элементарных нижних и верхних матриц преобразований, которая не приводит к возникновению других дополнительных решений. Следовательно, решения исходной и преобразованной систем совпадают.

На основании проведенных исследований показано, что алгоритм метода путем пошаговых устойчивых преобразований переводит исходную систему уравнений к виду $W\left(\vec{\Phi}^{k+1}-\vec{\Phi}^{k}\right)=R\left(A\vec{\Phi}^{k}-\vec{f}\right)$, где $W=W(\theta)$ – удобно разрешаемый оператор с четырехдиагональной почти верхнетреугольной матрицей; R – оператор невырожденных эквивалентных преобразований; θ – итерационный параметр компенсации. При этом матрицы операторов W и R выписываются явным образом. Нетрудно видеть, что в этом случае вопрос о корректности метода решается естественным образом.

Литература

1. **Ильин В.П.** Методы неполной факторизации для решения алгебраических систем. М.: Физматлит, 1995.
2. **Фомин А.А., Фомина Л.Н.** Полинейный рекуррентный метод решения СЛАУ с пятидиагональной матрицей // Четвертая Сибирская школа-семинар по параллельным и высокопроизводительным вычислениям (Томск, 9–11 октября 2007 г.). Томск: Дельтаплан, 2008. С. 192-200.
3. **Фомина Л.Н.** Использование полинейного рекуррентного метода с переменным параметром компенсации для решения разностных эллиптических уравнений // Вычислительные технологии. 2009. Т. 14, № 4. С. 108-120.

Численный метод расчета явления термического бара в озере Байкал

Б.О. Цыденов
Томский государственный университет

Численными методами моделируются возникновение и развитие конвективных течений под влиянием весеннеелетнего термобара на озере Байкал. Используется негидростатическая модель, в которой уравнение состояния связывает плотность с температурой и давлением. Установлено возникновение циркуляционного течения вблизи берега и его смещение к центру озера с течением времени.

Термический бар (термобар) представляет собой узкую зону в глубоком озере умеренных широт, в которой происходит погружение имеющей наибольшую плотность воды от поверхности до дна.

С физической точки зрения, причиной формирования термобара является так называемый эффект уплотнения при смешении вод, т. е. аномальное изменение плотности воды. Известно, что плотность воды определенного солевого состава, находящейся при фиксированном давлении, достигает максимума при некоторой температуре – температуре максимальной плотности (ТМП). Поэтому если смешиваются две водные массы, имеющие общую боковую границу, и температура одной выше ТМП, а другой ниже ТМП, то в результате получается смесь, которая будет тяжелее как первого, так и второго объемов. Естественно, более тяжелая вода должна опускаться, вследствие чего в месте смешения образуется как бы барьер для горизонтального перемещения воды. То есть термобар препятствует обмену водных масс между прибрежными и центральным районами озера, являясь в то же время зоной конвергенции этих масс, т.е. гидрологическим фронтом. По мере опускания более плотных вод их место занимают все новые и новые порции смеси, которая получается из подтягивающихся с обеих сторон к фронту теплых и холодных вод.

Для сохранения уникальности Байкала и его экосистемы необходимо понимание всех физических механизмов, участвующих в процессах водообмена и формировании качества его вод. С одной стороны, важность изучения термобара как явления, которое может оказать существенное влияние на процессы распространения загрязнения, состоит в том, что интенсивные нисходящие течения, возникающие между двумя конвективными ячейками, могут привести к быстрому распространению загрязнения из поверхностных слоев до очень больших

глубин. С другой стороны, установлено, что придонные воды Байкала «моложе» и богаче кислородом, чем воды основного глубинного ядра. Все это вызывает интерес к исследованию термобара.

Целью данной работы является численное моделирование конвективных течений для исследования явления термического бара в озере Байкал.

В основу исследования положена двумерная негидростатическая модель в приближении Буссинеска для конвективного течения:

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}=-\frac{1}{\rho_0}\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(K_x\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_y\frac{\partial u}{\partial y}\right);$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}=-\frac{1}{\rho_0}\cdot\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left(K_x\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_y\frac{\partial v}{\partial y}\right)-\frac{\rho}{\rho_0}g;$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0;$$

$$\frac{\partial T}{\partial t}+\frac{\partial uT}{\partial x}+\frac{\partial vT}{\partial y}=\frac{\partial}{\partial x}(D_x\frac{\partial T}{\partial x})+\frac{\partial}{\partial y}(D_y\frac{\partial T}{\partial y});$$

$$\rho(T,p)\approx\rho_m(p)\left\{1-\varphi(p)[T-T_m(p)]^2\right\},$$

где $T_m(p)=3{,}985694-0{,}020617\cdot p$; $\rho_m(p)=\rho_0+4{,}916021\cdot10^{-2}\cdot p$; $\varphi(p)=8{,}572628\cdot10^{-6}-7{,}061491\cdot10^{-9}\cdot p$; u, v – составляющие скорости по осям x и y; T – температура; c_0 – характерная плотность воды; p – давление; коэффициенты K_x , K_y и D_x , D_y характеризуют интенсивность диффузионного переноса импульса и тепла в соответствующем направлении.

Начальные условия $t=0$ соответствуют состоянию покоя $u=v=0$ и заданным полям температуры $T=T_0(y)$, по которым определяется гидростатическое распределение давления с учетом уравнения состояния. У берега $x=0$ ставятся граничные условия $u=0$; $v=0$; $T_x=0$, на открытой границе $x=L_x$: $u=0$; $v_x=0$; $T_x=0$; на дне ставится условие отсутствия теплообмена и задается связь касательных напряжений с придонной скоростью (квадратичный закон трения).

Решение конвективно-диффузионного уравнения основано на конечно-разностном методе контрольного объёма. Расчетная область разбивается на некоторое число непересекающихся контрольных объемов таким образом, что каждая узловая точка содержится в одном контрольном объеме. Заданная сетка определяется множеством узлов и множеством сеточных функций. Дифференциальное уравнение интегрируют по каждому контрольному объему. Следующий шаг состоит в аппроксимации полученных интегральных соотношений конечно-

разностными, и в результате получается совокупность разностных уравнений. Перед применением метода контрольного объема для получения дискретизации исходной системы строится шахматная сетка. Давление и температура определяются в узловых точках построенной сетки, а компоненты скорости определяются в точках, расположенных на гранях контрольного объема.

Численный алгоритм нахождения поля течения основан на разностной схеме Кранка–Никольсона [1]. Конвективные слагаемые в уравнениях аппроксимируются по противопотоковой схеме Upwind. Для расчета поля течения использована процедура SIMPLE [2]. Алгебраические системы решаются методом нижней релаксации (вычисление компонент скорости *u, v*) и явным методом Булеева (нахождение температуры и давления) [3].

Программа была протестирована для случая квадратной каверны с использованием результатов исследований В.И. Полежаева [4] для следующих краевых условий:

$$t = 0 : u = v = 0; T = T_0;$$

$$x = 0 : u = v = 0; T = T_L; \qquad x = L_x : u = v = 0; T = T_0;$$

$$y = 0; u = v = 0; \frac{\partial T}{\partial y} = 0; \qquad y = L_y : u = v = 0; \frac{\partial T}{\partial y} = 0.$$

Полученные результаты совпадают с картиной течения, приведенной в работе [4] (рис. 1).

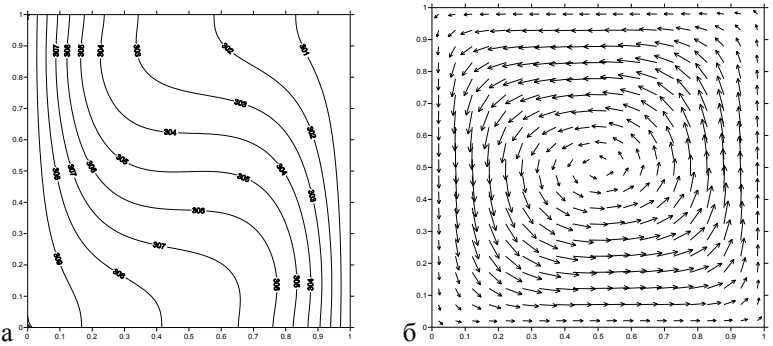


Рис. 1. Изотермы (*а*) и линии тока (*б*), полученные при тестировании, Pr = 0,7, Ra = 10^4

В численных расчетах сделана попытка воспроизвести реальные условия озера Байкал. Путем блокировки некоторых контрольных объемов (задавая нулевое значение скорости в выключенной зоне и используя очень большие значения коэффициента вязкости в этой зо-

не) прямоугольной неравномерной сетки [2] расчетную область приблизили к прибрежному профилю озера Байкал, взятому из работы [5]. Протяженность расчетной области Lx = 10 км, а глубина H = 900 м примерно соответствует средним глубинам южного бассейна Байкала. Используется неравномерная ортогональная сетка 126×90 с измельчением шагов у берега (x=0). В результате h_x меняется от 25 до 200 м, ($h_y = 10$ м). Начальное вертикальное распределение температуры взято на основе натурных наблюдений (март) [5].

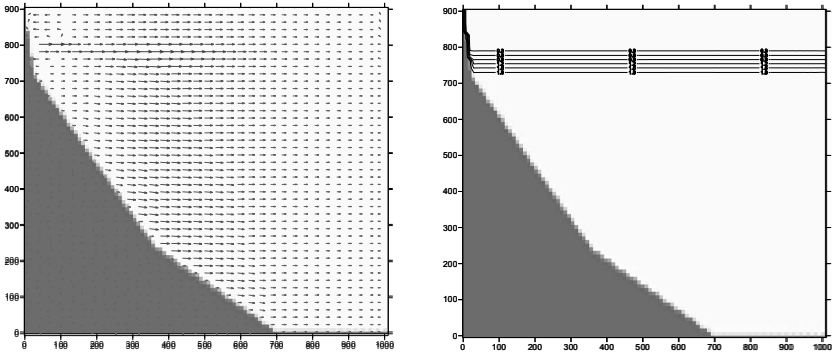


Рис. 2. Поле течений, изотермы через 24 ч соответственно

Из рис. 2 видно, что циркуляционное течение возникает у берега, и с течением времени начинает смещаться к центру озера.

Анализ результатов численных расчетов показывает, что процесс формирования циркуляционного течения согласуется с описанием поля температур и характера движений, полученным в результате обработки и интерпретации натурных наблюдений.

Литература

1. **Самарский А.А.** Теория разностных схем. М.: Наука, 1989. С. 257–279
2. **Патанкар С.** Численные методы решения задач теплообмена и динамики жидкости. М.: Энергоатомиздат, 1984. 149 с.
3. **Ильин В.П.** Методы неполной факторизации для решения алгебраических систем. М.: Физматлит, 1995. 288 с.

4. **Полежаев В.И., Бунэ А.В., Верезуб Н.А. и др.** Математическое моделирование конвективного тепломассообмена на основе уравнений Навье–Стокса. М.: Наука, 1987. 270 с.
5. **Shimaraev M.N., Verbolov V.I., Granin N.G., Sherstyankin P.P.** Physical Limnology of Lake Baikal: a Review. Irkutsk: Okayama, 1994.

Приближенное решение дифференциальных уравнений четвертого порядка с помощью эрмитовых сплайн-вейвлетов пятой степени*

Д.А. Турсунов, Э.А. Эшаров, А.Т. Бекмуратов
Ошский государственный университет, Киргизия
Томский государственный архитектурно-строительный университет

Используется новый тип эрмитовых сплайн-вейвлетов пятой степени для построения приближенного решения задачи Дирихле для дифференциальных уравнений четвертого порядка. Вейвлеты построены в базисе эрмитовых сплайнов пятой степени. Построенные вейвлеты принадлежат пространству C2, носителем материнских вейвлетов является отрезок [–1, 1].

Введение

В данной статье построены базисные вейвлеты в пространстве эрмитовых сплайнов пятой степени на отрезке [0, 1]. Построенные базисные вейвлеты применяются для численного решения задачи Дирихле для дифференциальных уравнений четвертого порядка.

Введем обозначения: **R**, **Z**, **N** – множество действительных, целых и натуральных чисел соответственно; $L_2(\mathbf{R})$ – линейное пространство квадратично-интегрируемых вещественных функции на **R**; $\langle u,v\rangle := \int_{\mathrm{R}} u(x)v(x)dx,\ u(x),v(x)\in L_2(\mathrm{R})$ – скалярное произведение; если $\langle u,v\rangle=0$, то функции *u, v* называются ортогональными; $\|f\|_2 = \sqrt{\langle f,f\rangle}$ – норма функции *f* в $L_2(\mathbf{R})$; $L_2(0, 1)$ – линейное пространство квадратично-интегрируемых вещественных функций на интервале (0, 1). Скалярное произведение в $L_2(0,1)$ имеет вид $\langle u,v\rangle := \int_0^1 u(x)v(x)dx,\ u(x),v(x)\in L_2(0,1)$; $H^1(0,1)$ – пространство функций $u(x)\in C(0,1)$ таких, что $u''(x)\in L_2(0,1)$; $H_0^1(0,1)$ – замыкание множества $\{u\in C[0,1]\cap C'[0,1]\cap C''[0,1]$: $u(0)=u(1)=0$, $u'(0)=u'(1)=0\}$ в $H^1(0, 1)$; Π_k – множество многочленов степени *k*, (*k* = 0, 1,

* Работа выполнена при поддержке РФФИ, проекты №09-07-90901, №09-01-90716.

2, …); V_n – пространство сплайнов пятой степени, удовлетворяющих следующим условиям, $n>0$:

а) $v \in C[0,1] \cap C'[0,1] \cap C''[0,1]$; б) $v(0)=v(1)=v'(0)=v'(1)=0$;

в) $v\big|_{(j/2^n,(j+1)/2^n)} \in \Pi_5\big|_{(j/2^n,(j+1)/2^n)}$, $j=0,\ldots,2^n-1$; $\dim(V_n)=3*2^n-1$.

Добеши [1] построила полуортогональные вейвлеты неполиномиального типа с компактным носителем. Коэн и др. [2] адаптировали ортогональные вейвлеты Добеши на отрезок [0,1]. Чуи и Вонг [3] построили полуортогональные сплайн-вейвлеты. В [4] эти сплайн-вейвлеты были адаптированы к отрезку [0,1]. Вонг [5] построил базис кубических сплайн-вейвлетов в пространстве Соболева. Донован и др. [6] построили ортогональные мультивейвлеты.

Недостатками построенных ранее вейвлетов является то, что они либо не имеют аналитического представления, либо расположены на достаточно широком носителе. И то и другое бывает чрезвычайно важно при их использовании для приближенного решения интегродифференциальных уравнений.

В [7] было доказано, что полуортогональные кубические мультивейвлеты на суперкомпактном носителе можно построить относительно скалярного произведения $\langle u', v' \rangle$.

Ранее нами построено приближенное решение задачи Дирихле для дифференциальных и интегродифференциальных уравнений второго порядка с помощью эрмитовых кубических сплайн-вейвлетов. В данной статье, используя эту же идею, с помощью нового типа эрмитовых сплайн-вейвлетов пятой степени построено приближенное решение задачи Дирихле для дифференциального уравнения четвертого порядка.

Пусть ϕ_1, ϕ_2 и ϕ_3 – базисные эрмитовы сплайны пятой степени:

$$\phi_1(t)=\begin{cases}(t+1)^3\left(1-3t+6t^2\right) & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\left(1+3t+6t^2\right) & \text{при} \quad 0\le t\le 1,\end{cases}$$

$$\phi_2(t)=\begin{cases}(t+1)^3\left(t-3t^2\right) & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\left(t+3t^2\right) & \text{при} \quad 0\le t\le 1,\end{cases}$$

$$\phi_3(t)=\begin{cases}(t+1)^3\dfrac{t^2}{2} & \text{при} \quad -1\le t\le 0,\\ (1-t)^3\dfrac{t^2}{2} & \text{при} \quad 0\le t\le 1.\end{cases}$$

Нетрудно заметить, что множество

$$\Phi_n:=\{\phi_1(2^n-j): j=1,\ldots,2^n-1\}\cup\{\phi_2(2^n-j): j=1,\ldots,2^n-1\}\cup\{\phi_3(2^n-j): j=0,\ldots,2^n\} \quad (1)$$

является базисом для V_n. Элементы Φ_n обозначим через $\{v_1,\ldots, v_{3*2^{n}-1}\}$.

Пусть Ψ_n – множество вейвлетов, но пока не конкретизируются: $\Psi_n:=\{\psi_1(2^n-j): j=1,\ldots,2^n-1\}\cup\{\psi_2(2^n-j): j=1,\ldots,2^n-1\}\cup\{\psi_3(2^n-j): j=0,\ldots,2^n\}$, (2) и W_n – линейное пространство, натянутое на Ψ_n, очевидно, что $\dim(W_n)=3*2^n-1$. Ниже мы докажем следующее равенство:

$$\int_0^1 w''(x)v''(x)dx = 0,\ \forall w\in\Psi_n,\ \forall v\in\Phi_n\ . \tag{3}$$

Из этого следует, что $V_n\cap W_n=\{0\}$. Кроме того, будет показано, что $V_{n+1}\supseteq V_n+W_n$ и $\dim(V_{n+1})=\dim(V_n)+\dim(W_n)$. Это означает, что $V_{n+1}=V_n\oplus W_n$.

Следовательно, мы получим разложение

$$H_0^1(0,1): H_0^1(0,1) = V_1\oplus W_1\oplus W_2\oplus\ldots. \tag{4}$$

Напомним, что $\Phi_1=\{v_1, v_2, v_3, v_4, v_5\}$; элементы Ψ_n обозначим $\Psi_n=\{w_{3*2^n},\ldots\}$.

Пусть $g_k:=v_k/\|v''_k\|_2$ при $k=1,2,3,4,5$ и $g_k:=w_k/\|w''_k\|_2$ при $k>5$. Тогда $\|g''_k\|_2=1$ при $n\in\mathbf{N}$. $(g''_k)_{k\in\mathbf{N}}$ является последовательностью Рисса в $L_2(0, 1)$.

Сплайн-вейвлеты

В данном разделе мы построим базисные вейвлеты на пространстве эрмитовых сплайнов пятой степени.

Пусть ϕ_1, ϕ_2 и ϕ_3 – сплайны пятой степени, определенные во введении. Очевидно, что они удовлетворяют условиям: $\phi_1, \phi_2, \phi_3\in C^2$, $\phi_1(0)=1$ $\phi'_1(0)=0$, $\phi''_1(0)=0$, $\phi_2(0)=0$ $\phi'_2(0)=1$, $\phi''_2(0)=0$, $\phi_3(0)=0$ $\phi'_3(0)=0$, $\phi''_3(0)=1$.

Следовательно, эрмитовы интерполяция для функции $f\in C^2(\mathbf{R})$ имеет следующий вид:

$$u=\sum_{j\in Z} f(j)\phi_1(\cdot-j)+f'(j)\phi_2(\cdot-j)+f''(j)\phi_3(\cdot-j), \tag{5}$$

$$\forall j\in Z:\ u(j)=f(j),\ u'(j)=f'(j),\ u''(j)=f''(j). \tag{6}$$

Используем построенные нами сплайн-вейвлеты:

$$\psi_1(x)=\phi_1(2x+1)-\phi_1(2x-1)-60\phi_3(2x+1)-60\phi_3(2x-1),$$

$$\psi_2(x)=2\phi_1(2x+1)-2\phi_1(2x-1)+13\phi_2(2x+1)+13\phi_2(2x-1)+16\phi_2(2x)-24\phi_3(2x+1)+$$
$$+24\phi_3(2x-1),$$

$$\psi_3(x)=\phi_1(2x+1)+\phi_1(2x-1)-\phi_2(2x+1)+\phi_2(2x-1)-68\phi_3(2x+1)-68\phi_3(2x-1)+16\phi_3(2x).$$

Применение

В этом разделе мы используем построенные вейвлеты для решения дифференциальных уравнений вида

$$\frac{d^4u}{dx^4}+p(x)u=f(x),\quad 0<x<1 \tag{7}$$

с граничным условием $u(0)=u(1)=u'(0)=u'(1)=0,$ (8)
где $p_k(x)$, $f(x)$ – заданные непрерывные функции. Коэффициенты уравнения (7) удовлетворяют условиям

$$0\le p_k(x)\le c_k,\ k = 0, 1, 2, 3,\ x\in[0, 1]. \quad (9)$$

Пусть $a(u,v)$ обозначает билинейную форму, $u, v \in H_0^1(0,1)$:

$$a(u,v) = \int_0^1 u''(x)v''(x)dx + \int_0^1 p(x)u(x)v(x)dx.$$

Тогда вариационная запись (7)-(8) имеет следующий вид:

$$a(u,v) = \langle f, v\rangle,\ \forall v \in H_0^1(0,1).$$

Соответствующая задача аппроксимации Галеркина: найти $u_n\in V_n$, при котором

$$a(u_n,v)=\langle f,v\rangle\ \forall v\in V_n. \quad (10)$$

По лемме Лакса–Милграмма (см. [8. С. 60]) задача (10) имеет единственное решение. Мы предлагаем использовать найденное выше множество вейвлетов $G=\{g_1,\ldots,g_2^{n+1}\}$ как базис для V_n. С этим базисом для V_n задача (10) может быть дискретизирована следующим образом:

$$\sum_{k=1}^{2^{n+1}} a(g_j, g_k)c_k = \langle g_j, f\rangle, \qquad j = 1,\ldots,2^{n+1}.$$

Обозначим матрицу $A_n=(a(g_j,g_k))_{1\le j,k\le 2^{n+1}}$. Можно доказать, что число обусловленности матрицы A_n равномерно ограничено (не зависит от n).

Вывод

Численные результаты показывают, что точность улучшается с повышением n.

Литература

1. **Добеши И.** Десять лекций по вейвлетам. Москва; Ижевск: РХД, 2001.
2. **Cohen A., Doubechies I., Vial P.** Wavelets on the interval and fast wavelet transforms // Appl. Comput. Harmonic Anal. 1993. Vol. 1. P. 54-81.
3. **Chui C.K., Wang J.Z.** On compactly supported wavelets and a duality principle // Trans. Amer. Math. Soc. 1992. Vol. 330. P. 903-916.
4. **Chui C.K., Quak E.** Wavelets on a bounded interval // Numerical Methods in Approximation Theory eds. D. Braess and L.L. Schumaker. Birkhauser, Brasel, 1992. Vol. 9. P. 53-75.

5. **Wang J.Z.** Cubic spline wavelet bases of Sobolev spaces and multilevel interpolation // Appl. Comput. Harmonic Anal. 1996. Vol. 3. P.154-163.
6. **Donovan G., Geronimo J.S., Hardin D.P., Massopust P.R.** Construction of orthogonal wavelets using fractal interpolation functions SIAM // J. Math. Anal. 1996. Vol. 27. P. 1158-1192.
7. **Jia R.-Q., Liu S.-T.** Wavelet bases of Hermite cubic splines on the interval // Advances Computational Mathematics. 2006. Vol. 25. P. 23-39.
8. **Brenner S.C., Scott L.R.** The Mathematical Theory of Finite Element Methods. New York: Springer. 1994.

О кластеризации в анализе движения в изображении с неподвижной камеры

А.И. Шапошников, Е.В. Шапошникова
Томский институт академии ВЭГУ

В [1] представлен один из подходов к решению задачи кластеризации (разбиения) ряда изображений, образующих видеоряд, на участки одного цвета. Предлагаемый метод позволяет получать кластеры, пересекающиеся с несколькими кадрами. Однако кластеризация может быть проведена не всегда. В предлагаемой работе исследуются условия проводимости кластеризации, и решается проблема непрерывности для получаемых результатов.

Введение

Задача анализа изображения неподвижной камеры вызвана большим числом прикладных задач, например слежения, управления движением и др. В качестве основных этапов в этом анализе выделяются фильтрация изображения, выделение движущихся объектов и слежение за выделенными объектами – трекинг (tracking) [2]. Несмотря на значительные продвижения в каждой из этих задач в отдельности (отделение теней и бликов, выделение движущегося множества [3], трекинг выделенных объектов, анализ группы вращений и т.д.), задача анализа движений объектов произвольного ролика (например, из [4]) еще не решена. Как отмечено в [2], «практическое использование вычислительной техники ограничено, и исследования должны продолжаться».

В этой работе обсуждены техника кластеризации видеоизображений на основе Mean Shift и некоторые результаты ее применения

Фильтрация и кластеризация

Дадим некоторые определения. Неопределяемым понятием будем считать кадр видеоизображения, т.е. то, что мы видим на экране монитора. Последовательность кадров назовем видеорядом. В пределах этой статьи рассматриваем только кадры, снятые с одной и той же неподвижной камеры. Каждому кадру сопоставим его номер в последовательности. Пиксель – это элемент кадра видеоизображения, соответствующий некоторой точке на мониторе. Каждый пиксель отождест-

вим с точкой **(*x,y,t,r,g,b*)** в пространстве $\mathbf{R}^6$, где ***x,y*** – координаты точки на экране; ***t*** – номер кадра; ***r,g,b*** – компоненты цвета точки экрана.

Процедуру MeanShift-кластеризации рассмотрим, следуя [6]. Выбираем ***h*** – шаг окна (как правило, ***h*=16** вполне подходит) и для каждой точки **(*x,y,t,r,g,b*)** строим рекуррентно последовательность центроидов C_i. Рассмотрим результаты кластеризации двух последовательных кадров видеоролика, снятого с неподвижной камеры [7] (рис. 1).

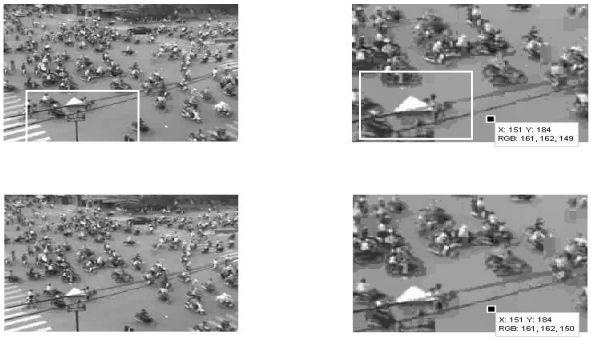


Рис. 1. Кластеризация кадров

Как видим, полотно дороги после кластеризации имеет на обоих кадрах одинаковый цвет (161, 162, 150). Согласованность касается и более тонких объектов (рис. 2, выделенный прямоугольник).



Рис. 2. Выделение объектов

Объект – мотоциклист движется, что заметно по изменению и первой координаты, и цвета брюк на исходных кадрах. На обработанных

кадрах брюки имеют разные координаты, но одинаковый цвет (153,155,144), т.е. они попали в единый кластер, расположенный в разных кадрах.

Влияние скоростей объектов

При проведении такой схемы кластеризации и фильтрации в один кластер могут попасть точки, принадлежащие совершенно разным объектам, которые имеют одинаковые координаты в разных кадрах.

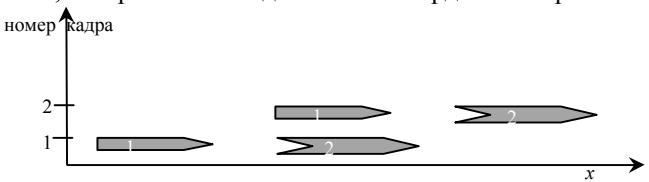


Рис. 3. Использование ограниченного числа кадров

Рассмотрим рис. 3. Здесь плоскость кадра представлена одной пространственной координатой ***x***. По вертикали откладывается номер кадра. Из-за низкой частоты кадров и высоких скоростей *1* и *2*, объект *1* оказался во втором кадре на том месте, на котором был объект *2*. Цвета объектов схожие, значит, существуют пиксели $\boldsymbol{p_1}$ и $\boldsymbol{p_2}$, принадлежащие разным объектам, которые будут классифицированы в один кластер при любом значении шага. Результат неправильный.

Чтобы обеспечить правильную кластеризацию, надо согласовать частоту кадров, масштаб по ***t*** и скорости наблюдаемых объектов.

При включении в видеоролик промежуточных кадров ситуация меняется (рис. 4).

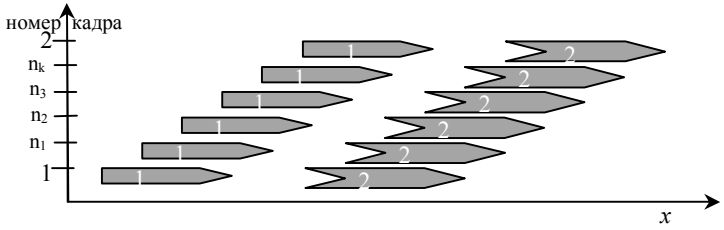


Рис. 4. Включение дополнительных промежуточных кадров

В этом случае даже без изменения первоначальных настроек мы получим правильный результат. При частоте ***n*** кадров в секунду получаем ограничение на скорости объектов в кадре не более ***nh*** пикселей

в секунду. Таким образом, мы получаем допустимое для кластеризации пикселей расположение объектов в кадрах.

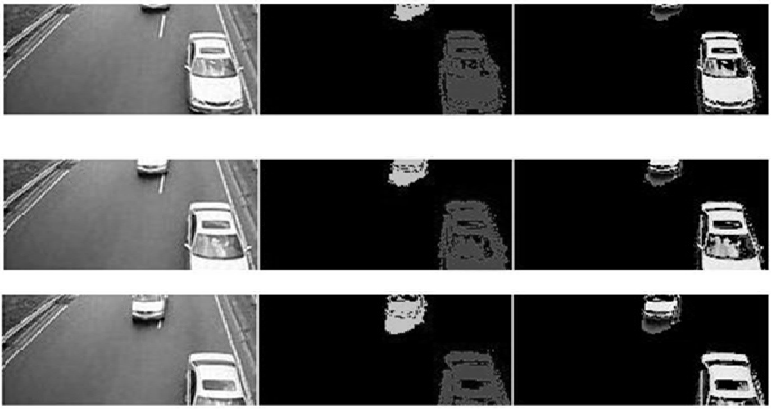


Рис. 5. Ролик с автомобилем

Соответствующий модуль был реализован на MatLab. Обрабатывался ролик размером 160×120 пикселей, 8 с, с легковыми машинами, движущимися в кадре. Ролик был снят камерой наблюдения, размещенной на автомобильной магистрали. Смещение автомобилей в соседних кадрах составляло не более 13 пикселей за кадр. Шаг окна ***h=16*** пикселей. Общее время обработки сюжета составило 354 с. Все автомобили были правильно кластеризованы и раскрашены каждый в свой цвет.

На рис. 5 показаны три последовательных кадра. Слева – исходное изображение. В центре – результат кластеризации. Как видим, пиксели корпуса каждого автомобиля на всех трех кадрах составляют единый кластер. Справа – те же кластеры без раскрашивания всех элементов кластера в единый цвет (пиксели имеют свои цвета).

Литература

1. **Шапошников А.И.** Выделение движущихся объектов в анализе изображения неподвижной камеры// Информационные технологии и математическое моделирование: Материалы VIII Всероссийской научно-практической конференции с международным участием (13–4 ноября 2009 г.). Томск: Изд-во Том. ун-та, 2009. Ч. 2. С. 297-301.

2. **Cucchiara R., Grana C., Piccardi M., Prati A.** Detecting moving objects, ghosts, and shadows in video streams // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2003. Vol. 25, № 10. P. 1337–1342.
3. **Гаганов В., Конушин А.** Сегментация движущихся объектов в видеопотоке. http://cgm.computergraphics.ru/content/view/67.
4. http://YouTube.com.
5. **Comaniciu D., Meer P.** Mean Shift Analysis and Applications // IEEE Int. Conf. Computer Vision (ICCV'99). Kerkyra, Greece, 1999. P. 1197-1203.
6. http://youtube.com/watch?v=gbPLV2kB4-Y&feature=related

Исследование неоднородной структуры биологического объекта при пропускании электрического тока небольшой силы

Е.С. Шерина
Томский государственный университет

Для исследования неоднородной структуры биологических объектов используется томография. В работе рассматриваются прямая и обратная задачи электроимпедансной томографии, которые позволяют воспроизводить распределение импеданса внутри биологического объекта. Решение осуществляется численно с использованием метода конечных объемов.

Область применения томографии охватывает медицинскую диагностику, геофизику, промышленную интроскопию и т.д. В медицине томографические обследования позволяют получить данные об анатомической структуре и функционировании органов и систем организма, помогают выявлять аномалии в участках органов и тканях. Электроимпедансная томография (ЭИТ) позволяет воспроизводить внутреннюю структуру биологических объектов. Исследования ЭИТ осуществляются с использованием электрического тока.

Прямая задача ЭИТ заключается в нахождении распределения потенциала электрического поля внутри биологического объекта при заданном распределении проводимости и конфигурации поверхностных источников тока. Обратная задача ЭИТ – в нахождении распределения электрического сопротивления (импеданса) внутри биологического объекта.

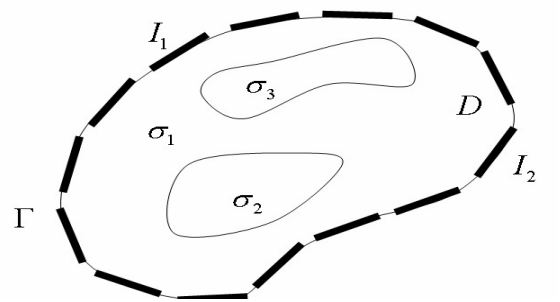


Рис. 1. Область исследования

Рассматривается биологический объект (рис. 1), через который пропускается электрический ток небольшой силы частотой не более 100–150 кГц через расположенные локально на поверхности электроды. Объект находится в воздушной среде, имеет достаточно гладкую границу Г и характеризуется значениями коэффициента электрической проводимости σ, который зависит от координат (x,y). I_1 и I_2 – прикрепленные на поверхности объекта электроды с площадью поверхности S_1 и S_2 соответственно.

Математическая постановка задачи получается с использованием уравнения Максвелла в проводниках, закона Ома для неподвижных проводников, закона Кирхгоффа и имеет следующий вид:

$$\frac{\partial}{\partial x}\left(\sigma(x,y)\frac{\partial\varphi(x,y)}{\partial x}\right)+\frac{\partial}{\partial y}\left(\sigma(x,y)\frac{\partial\varphi(x,y)}{\partial y}\right)=0,\quad (x,y)\in D, \tag{1}$$

$$\left.\frac{\partial\varphi(x,y)}{\partial\vec{n}}\right|_{\Gamma_0}=0, \tag{2}$$

$$I_{1,2}=\pm J_n S=\pm\sigma(x,y)S\left.\frac{\partial\varphi(x,y)}{\partial\vec{n}}\right|_{\Gamma_{1,2}}. \tag{3}$$

Причем должен выполняться закон сохранения $I_1+I_2=0$. $\vec{n}$ – единичный вектор внешней нормали; J_n – плотность тока на границе.

Дискретные аналоги дифференциального уравнения и граничных условий получаются методом конечного объема на неструктурированной треугольной сетке, которая строится в исследуемой области.

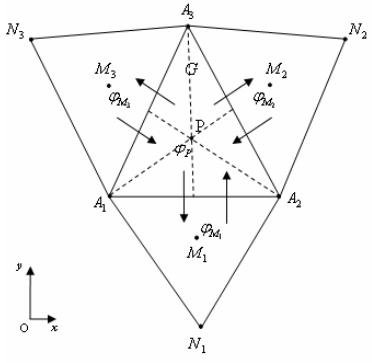


Рис. 2. Фрагмент сетки

Рассматривается фрагмент сетки (рис. 2), на котором изображены четыре конечных объема. $G=\Delta A_1A_2A_3$, направление обхода – против часовой стрелки. Уравнение (1) интегрируется по конечному объему G, окружающему узел P:

$$\iint_G\left(\frac{\partial}{\partial x}\left(\sigma\frac{\partial\varphi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial\varphi}{\partial y}\right)\right)dG=0$$

По теореме Гаусса–Остроградского в двумерном случае получается следующее уравнение:

$$\iint_G\left(\frac{\partial}{\partial x}\left(\sigma\frac{\partial\varphi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial\varphi}{\partial y}\right)\right)dG=$$

$$= \oint\limits_{\partial G} (\vec{n}, \sigma grad\varphi) dl = \oint\limits_{\partial G} \sigma \frac{\partial \varphi}{\partial n} dl = 0,$$

где n – единичный вектор внешней нормали к границе треугольной ячейки. Иначе уравнение запишется в виде

$$\int\limits_{A_1A_2} \sigma \frac{\partial \varphi}{\partial n_{12}} dl + \int\limits_{A_2A_3} \sigma \frac{\partial \varphi}{\partial n_{23}} dl + \int\limits_{A_3A_1} \sigma \frac{\partial \varphi}{\partial n_{31}} dl = 0.$$

В результате для получения разностного уравнения необходимо аппроксимировать интегралы по ребрам, используется формула средних прямоугольников:

$$\int\limits_{A_1A_2} \sigma \frac{\partial \varphi}{\partial n_{12}} dl + \int\limits_{A_2A_3} \sigma \frac{\partial \varphi}{\partial n_{23}} dl + \int\limits_{A_3A_1} \sigma \frac{\partial \varphi}{\partial n_{31}} dl \approx$$

$$\approx \left(\sigma \frac{\partial \varphi}{\partial n_{12}} \right)_{m_{12}} \Delta l_{12} + \left(\sigma \frac{\partial \varphi}{\partial n_{23}} \right)_{m_{23}} \Delta l_{23} + \left(\sigma \frac{\partial \varphi}{\partial n_{31}} \right)_{m_{31}} \Delta l_{31} = 0,$$

где $\left(\sigma \frac{\partial \varphi}{\partial n_{ij}} \right)_{m_{ij}}$ – производная по направлению внешней нормали к конечному объему G на ребре A_iA_j, вычисленная в середине ребра; Δl_{ij} – длина ребра A_iA_j, m_{ij} – середина ребра.

Аппроксимация производных в последнем выражении производится в промежуточных точках C_p и C_{Mk}, отложенных по нормали n_{ij} к ребру A_iA_j на расстоянии Δn_{ij} и Δn_{ijk} соответственно от A_iA_j. С помощью разложения функции φ по формуле Тейлора в точках C_p и C_{Mk} получается разностный аналог производной

$$\left(\frac{\partial \varphi}{\partial n_{ij}} \right)_{m_{ij}} = \frac{\varphi(C_{M_k}) - \varphi(C_P)}{\Delta n_{ij} + \Delta n_{ijk}} + O(\Delta n).$$

Предполагается, $\Delta n_{ij} = \Delta n_{ijk}$, $\varphi(C_P) \approx \varphi(P)$, $\varphi(C_{M_k}) \approx \varphi(M_k)$. Кроме того, $\Delta n_{ij} = \frac{2S_{ij}}{\Delta l_{ij}}$.

В итоге получается разностный аналог уравнения (1):

$$\sigma \frac{\varphi(M_1) - \varphi(P)}{4S_{12}} (\Delta l_{12})^2 + \sigma \frac{\varphi(M_2) - \varphi(P)}{4S_{23}} (\Delta l_{23})^2 + \sigma \frac{\varphi(M_3) - \varphi(P)}{4S_{31}} (\Delta l_{31})^2 = 0. \quad (4)$$

Для аппроксимации граничных условий в рассматриваемой области D (см. рис. 1) к граничным ячейкам достраиваются симметричные ячейки так, чтобы прямая, соединяющая центры масс данных контрольных объемов, была параллельна вектору нормали. Для граничного условия (2), выражающего контакт тела с воздухом, получаем следующий разностный аналог:

$$\varphi(M_k) - \varphi(P) = 0. \quad (5)$$

А условие (3) контакта тела с электродом аппроксимируется в виде

$$\sigma \frac{\varphi(M_k) - \varphi(P)}{2\Delta n_{ij}} \Delta l_{ij} = I. \qquad (6)$$

Разностная задача (4)–(6) имеет первый порядок аппроксимации, устойчива и её решение сходится к точному решению с первым порядком точности.

Полученная разностная задача решается методом бисопряженных градиентов Bi-CGSTAB без предобуславливания.

В работе рассматривается тестовая задача для уравнения Лапласа в круге единичного радиуса, на которой показана сходимость численного решения к точному решению с первым порядком точности. Также проведен численный эксперимент для модели биологического объекта. Найдено распределение потенциала электрического поля с помощью пакета Fluent при коэффициенте электрической проводимости σ=0,5 и силе тока $I_{1,2}$=±2,5. Выполнено численное решение задачи с теми же условиями.

Обратная задача – тип задач, часто возникающий во многих разделах науки, когда значения параметров модели должны быть получены из наблюдаемых данных. Обратная задача ЭИТ является коэффициентной обратной задачей. Реконструкция неизвестного распределения проводимости внутри исследуемого объекта производится на основе данных измерения разности потенциалов между точками его поверхности. Составляется функционал вида

$$\delta\sigma = D^{-1}(\varphi^* - \varphi),$$

где D^{-1} – оператор преобразования отклонения вычисленного значения потенциала φ от измеренного φ^* в вариацию проводимости σ. Данные для решения обратной задачи получаются при решении множества прямых задач с различными конфигурациями электрического тока и при всевозможных парах электродов.

ЭИТ – новый и не слишком известный томографический метод исследования. У современного электроимпедансного томографа существует проблема точности получаемого изображения, соответствия его действительности. Не удается обнаружить образования (например, опухоли), размер которых меньше 1% от размера области (томосреза). Решение такой проблемы требует рассмотрения новых методов решения задачи и использования более мощной вычислительной техники.

Литература

1. **Пеккер Я.С., Бразовский К.С.** Электроимпедансная томография. Томск: Изд-во НТЛ, 2004. 192 с.

2. **Фирсов Д. К.** Метод контрольного объема на неструктурированной сетке в вычислительной механике : Учеб. пособие. Томск, 2007. 46 с.

Генетический алгоритм организации мультипрограммного функционирования распределенных вычислительных систем*

А.В. Ефимов
Сибирский государственный университет
телекоммуникаций и информатики, Новосибирск

Рассмотрена проблема организации мультипрограммного функционирования распределенных вычислительных систем (ВС). Предложен генетический алгоритм оптимизации функционирования распределенных ВС при обработке наборов параллельных программ с нефиксированными параметрами. Представлены результаты моделирования.

В архитектурном плане распределенная вычислительная система (ВС) представляется множеством взаимодействующих элементарных машин, оснащенных средствами коммуникаций и внешними устройствами. Элементарная машина (ЭМ) – это основной функциональный и структурный элемент ВС; конфигурация ЭМ допускает варьирование в широких пределах – от процессорного ядра до ЭВМ. Все основные ресурсы распределенных ВС (арифметико-логические устройства, память, средства управления и коммуникаций) являются логически и технически рассредоточенными. Число ЭМ в распределённых ВС допускает варьирование от десятков до сотен тысяч.

Перспективными считаются мультипрограммные режимы функционирования ВС [1] (обработка наборов задач и обслуживание потока задач), которые позволяют организовать разделение ресурсов между несколькими одновременно выполняющимися задачами.

Исследования пользовательских запросов показывают, что более 80% параллельных программ, выполняемых с использованием распределенных ВС, обладают свойством адаптивности – способностью подстраиваться под размер выделенной подсистемы из ЭМ. Существующие системы управления ресурсами распределенных ВС позволяют запускать адаптирующиеся программы, однако в них отсутствует возможность учитывать предпочтения пользователя на ранги выделяемых подсистем ЭМ.

* Работа выполнена при поддержке РФФИ (гранты № 09-07-90403, 09-07-13534, 09-07-00095, 08-07-00022), в рамках ФЦНТП (ГК 02.740.11.0006).

Множество вариантов подсистем ЭМ, допустимых для выполнения параллельной программы, может быть представлено в виде вектора рангов подсистем и сопоставленных им приоритетов. Это позволяет учитывать специфику параллельной задачи при оптимизации функционирования распределенных ВС.

Рассмотрим формальную постановку задачи.

Имеется распределенная ВС, состоящая из N ЭМ. На вход системы поступает набор из L задач. Каждая задача $j \in J = \{1, 2, \ldots, L\}$ является параллельной адаптирующейся программой и характеризуется вектором параметров

$$P_j = (p_{j1}, p_{j2}, \ldots, p_{jq_j}),\ i = 1, 2, \ldots, q_j,$$

где параметр $p_{ji} = (r_{ji}, t_{ji}, \mu_{ji})$ описывает i-й вариант конфигурации ресурсов, допустимой для реализации задачи. Параметр r_{ji} – ранг задачи (количество ЭМ, необходимое для её решения, $r_{ji} \in C = \{1, 2, \ldots, N\}$); t_{ji} – время, требуемое для решения задачи с использованием r_{ji} ЭМ; μ_{ji} задаёт приоритет i-й конфигурации ресурсов, $\mu_{ji} \in (0, 1]$. Чем выше значение μ_{ji}, тем более предпочтительным является выделение для решения задачи r_{ji} элементарных машин на время t_{ji}. Параметр q_j – количество параметров j-й задачи.

Считаем, что количество ЭМ в распределенной ВС не достаточно, чтобы одновременно решить все задачи набора.

Требуется построить расписание S решения параллельных задач на распределенной ВС. Для каждой задачи необходимо определить момент времени τ_j начала решения её ветвей и их распределение по элементарным машинам. Пусть x_j – номер выбранного параметра (конфигурации ресурсов) задачи $j \in J$ для её решения ($x_j \in \{1, 2, \ldots, q_j\}$).

Обозначим через $x_{ji} \in C = \{1, 2, \ldots, N\}$ – номер ЭМ, на которую распределена ветвь $i \in \{1, 2, \ldots, r_{jx_j}\}$ задачи $j \in J$.

Введем функцию $J(t) = \{j \in J \mid \tau_j \le t \le \tau_j + t_j\}$ – множество задач, решаемых на распределенной ВС в момент времени t.

Будем называть расписание

$$S = (\tau_1, \tau_2, \ldots, \tau_L; x_1, x_2, \ldots, x_L; x_{11}, x_{12}, \ldots, x_{1r_{1x_1}}, \ldots, x_{L1}, x_{L2}, \ldots, x_{Lr_Lx_L})$$

допустимым, если оно удовлетворяет ограничениям.

1) В любой момент времени на ресурсах распределенной ВС решается не более N ветвей параллельных задач: $\sum_{j \in J(t)} r_{jx_j} \le N,\ \forall t \in \boldsymbol{R}$.

2) Ветви параллельных задач решаются на разных элементарных машинах:

$$\prod_{j\in J(t)}\prod_{j'\in J(t)\setminus\{j\}}(x_{ji}-x_{j'i'})\neq 0,\ \forall t\in \boldsymbol{R},\ i=1,2,\ldots,\ r_{jx_j},$$

$i'=1, 2, \ldots, r_{j'x_{j'}}$ Обозначим через Ω множество допустимых расписаний.

В качестве показателей оптимальности расписаний будем использовать время $T(S)$ окончания решения последней задачи и суммарную «удовлетворенность» пользователей $M(S)$:

$$T(S)=\max_{j\in J}\{\tau_j+t_j\}\ ,\ M(S)=\sum_{j=1}^{L}\mu_{jx_j}\ .$$

Введем составной показатель $F(S)$ эффективности расписания S:

$$F(S)=\alpha*M(S)/M_{\max}+\beta*T_{\min}/T(S)\ ,$$

где $T_{\min}$ – нижняя оценка времени решения задач набора; $M_{\max}$ – абсолютный максимум суммарного значения «предпочтений» пользователей для данного набора задач; α, β – весовые коэффициенты целевых функций.

Итак, требуется найти допустимое расписание $S\in\Omega$, доставляющее максимум целевой функции $F(S)$. Формально

$$F(S)=\alpha*M(S)/M_{\max}+\beta*T_{\min}/T(S)\ \rightarrow \max_{S\in\Omega} \qquad (1)$$

при ограничениях:

$$\sum_{j\in J(t)} r_{jx_j}\leq N\ ,\ \forall t\in \boldsymbol{R}, \qquad (2)$$

$$\prod_{j\in J(t)}\prod_{j'\in J(t)\setminus\{j\}}(x_{ji}-x_{j'i'})\neq 0,\ \forall t\in \boldsymbol{R},\ i=1,2,\ldots,\ r_{jx_j}\ ,\quad i'=1,2,\ldots,\ r_{j'x_{j'}}\ , \qquad (3)$$

$$x_{ji}\in C,\ j=1,2,\ldots,L,\ i=1,2,\ldots,\ r_{jx_j}\ , \qquad (4)$$

$$\tau_j\in \boldsymbol{R},\ j=1,2,\ldots,L. \qquad (5)$$

Задача (1)–(5) относится к дискретной оптимизации и является трудноразрешимой.

Разработан приближенный алгоритм решения задачи (1)–(5). В основу алгоритма положены эволюционные (генетические) методы оптимизации. Изначально строятся несколько допустимых расписаний («особей»), путем распределения задач набора по пакетам («генам»), с применением алгоритмов упаковки объектов в контейнеры (Best Fit, First Fit, Next Fit). При этом для каждой задачи при каждом распределении случайным образом назначается один из вариантов параметров (описанных вектором значений с приоритетами). Допустимое распи-

сание предполагает последовательное выполнение на распределенной ВС пакетов, объединяющих несколько задач, решаемых параллельно. Совокупность полученных расписаний формирует начальную «популяцию». Последующие шаги формирования итогового расписания эмитируют жизнедеятельность популяции и состоят из следующих этапов: развитие текущей популяции или образование «потомства» с использованием кроссинговера (алгоритма перетасовки генов) и «мутации» – формирования новой популяции. Процесс повторяется до тех пор, пока на протяжении определенного количества популяций не будет получено «потомство» (новая популяция) с лучшими значениями целевых функций. Таким образом, генетический алгоритм позволяет строить расписания, обеспечивающие субминимум времени выполнения задач набора и субмаксимум относительной «удовлетворенности» пользователей.

Алгоритм реализован на языке программирования C++. Моделирование и численные эксперименты разработанного алгоритма осуществлялись с использованием ресурсов пространственно-распределенной мультикластерной вычислительной системы Института физики полупроводников СО РАН и Центра параллельных вычислительных технология ГОУ ВПО «Сибирский государственный университет телекоммуникаций и информатики». Поступающие задачи генерировались с помощью модели рабочей загрузки, предложенной W. Cirne и F. Berman [2] на основе статистики загрузки некоторых промышленных ВС. Приоритеты параметрам программ (требуемым ресурсам) задавались случайным образом с использованием равномерного распределения. Рассматривались наборы с количеством задач от 100 до 10000.

Результаты моделирования показали, что время поиска расписаний генетическим алгоритмом компенсируется уменьшением суммарного времени выполнения всех параллельных задач набора.

Литература

1. **Хорошевский В. Г.** Архитектура вычислительных систем. М.: МГТУ им. Н.Э. Баумана, 2008. 520 с.
2. ***Cirne, W.*** A Model for Moldable Supercomputer Jobs / Walfredo Cirne, Francine Berman // 15th Intl. Parallel & Distributed Processing Symposium. April 2001.
3. (http://www.cs.huji.ac.il/labs/parallel/workload/models.html)

Алгоритмы децентрализованной диспетчеризации параллельных программ в пространственно-распределенных вычислительных системах*

М.Г. Курносов, А.А. Пазников
Институт физики полупроводников СО РАН, Новосибирск

Рассматриваются алгоритмы децентрализованной диспетчеризации в пространственно-распределенных вычислительных системах параллельных программ с целью минимизации времени их обслуживания. Приводятся результаты натурных экспериментов на действующей мультикластерной вычислительной системе.

Введение

Сегодня для решения сложных задач науки и техники широкое распространение получили пространственно-распределенные вычислительные системы (ВС). Они представляют собой макроколлективы пространственно рассредоточенных ВС (вычислительных подсистем), взаимодействующих между собой через локальные и глобальные сети связи [1]. К таким системам относятся GRID-системы и мультикластерные ВС.

При организации функционирования распределенных ВС возникает задача диспетчеризации параллельных программ, поступающих в систему. Для каждой программы необходимо определить, на каких вычислительных ресурсах каких подсистем она будет выполняться. В средствах диспетчеризации должны учитываться динамичность состава систем и переменная загрузка их ресурсов.

Существующие программные средства диспетчеризации параллельных программ (GridWay, Common Scheduler Framework, Nimrod/G, Condor-G, GrADS, AppLeS) являются централизованными [2]. В таких системах отказ управляющего узла может привести к неработоспособности всей ВС. Кроме того, применение централизованных систем диспетчеризации в большемасштабных ВС ограничено вычислительной сложностью поиска требуемых ресурсов.

При децентрализованной схеме диспетчеризации в пространственно-распределенной ВС функционирует коллектив диспетчеров, при-

* Работа выполнена при поддержке РФФИ (гранты № 08-07-00018, 09-07-00095, 09-07-13534) и Совета по грантам Президента РФ (грант № НШ-2121.2008.9).

нимающий решение о выделении ресурсов для программ пользователей. Это позволяет достичь живучести ВС, т.е. ее способности продолжать работу при отказах отдельных вычислительных подсистем.

Постановка задачи

Пусть распределенная ВС состоит из H вычислительных подсистем и включает N элементарных машин (ЭМ), n_i – количество ЭМ в составе подсистемы $i \in S = \{1, 2, \ldots, H\}$.

Известны значения $b_{ij} = b(i, j, m)$ – пропускной способности канала связи между подсистемами $i, j \in S$ при передаче сообщений размером m байт ($[b(i, j, m)]$ = байт/с).

На каждой подсистеме присутствует диспетчер, который поддерживает очередь параллельных программ и осуществляет поиск вычислительных ресурсов для их выполнения.

Коллектив диспетчеров представлен в виде ориентированного графа $G = (S, E)$, в котором вершинам соответствуют диспетчеры, а ребрам – логические связи между ними. Наличие дуги $(i, j) \in E$ означает, что диспетчер i может отправлять ресурсные запросы диспетчеру j. Множество всех вершин j, входящих в дуги (i, j), образует локальную окрестность $L(i) = \{j \in S: (i, j) \in E\}$ диспетчера i.

Пользователь направляет программу и запрос на выделение ресурсов одному из диспетчеров. Диспетчер в соответствии с реализованным алгоритмом осуществляет поиск (суб)оптимальной подсистемы $i^* \in S$ для выполнения программы пользователя.

Считаем, что программа характеризуется рангом r – количеством параллельных ветвей, ожидаемым временем t выполнения программы и суммарным размером z исполняемых файлов и данных ($[z]$ = байт).

Децентрализованный алгоритм диспетчеризации

При поступлении программы в очередь диспетчера i он опрашивает диспетчеры $j \in L(i)$ из локальной окрестности и получает значения текущего количества задач в их очередях и оценку времени t_j, через которое программа может быть запущена на ресурсах подсистемы j в случае передачи задачи в её очередь. Через систему мониторинга определяются значения пропускных способностей b_{ij} каналов связи между подсистемами. Выбирается подсистема $i^* = \operatorname{argmin}\{T(j)\}$, $j \in L(i)$, $T(i) = z / b(i, j, z) + t_j + t$, обеспечивающая минимальное значение времени T обслуживания параллельной программы. Время обслуживания включает время доставки файлов и данных до выделенной подсисте-

мы, время ожидания в очереди подсистемы и время выполнения программы.

Затем программа пересылается в очередь подсистемы i^*. Важно отметить, что при передаче задачи в очередь подсистемы i^*, для нее аналогично периодически будет осуществляться поиск ресурсов. Это обеспечивает адаптацию под динамически изменяющуюся загрузку ресурсов пространственно-распределенных ВС.

Программный пакет GBroker

В Центре параллельных вычислительных технологий ГОУ ВПО «Сибирский государственный университет телекоммуникаций и информатики» (ЦПВТ ГОУ ВПО «СибГУТИ») ведется разработка программного пакета GBroker децентрализованной диспетчеризации программ в распределенных ВС.

В пакет (рис. 1) входят диспетчер gbroker, который устанавливается на каждой из подсистем и взаимодействует с локальным менеджером задач, клиентская часть gclient, реализующая интерфейс между пользователем и распределенной ВС, и модуль netmon мониторинга производительности каналов связи на уровне стека протоколов TCP/IP.

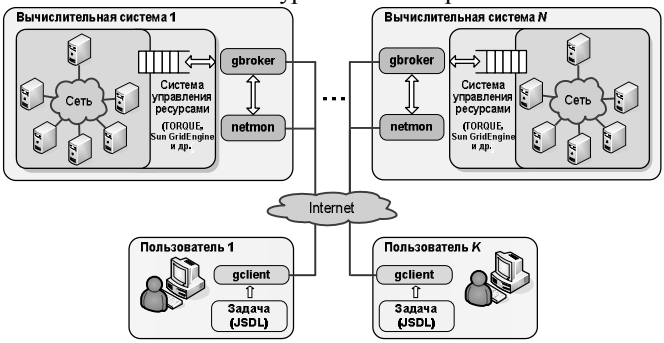


Рис. 1. Функциональная структура пакета GBroker

Администратор настраивает локальные окрестности диспетчеров gbroker, указывая, какие диспетчеры с какими могут обмениваться программами из своих очередей, и настраивает сервис netmon.

Пользователь формирует задание, состоящее из параллельной MPI-программы и паспорта на языке ресурсных запросов JSDL, и отправляет его средствами gclient любому из диспетчеров gbroker. Диспетчер в соответствии с описанным алгоритмом выбирает подсистему, на которой затем выполняется программа.

Результаты экспериментов

На мультикластерной системе ЦПВТ ГОУ ВПО «СибГУТИ» проведено исследование разработанного алгоритма диспетчеризации и программного средства. Использовались кластеры следующих конфигураций: Xeon16 (4 узла – 2 x Intel Xeon 5150), Xeon32 (4 узла – 2 x Intel Xeon E5345), Xeon80 (10 узлов – 2 x Intel Xeon E5420). Сеть связи между кластерами – Fast Ethernet.

В качестве тестовых задач использовались MPI-программы из пакета NAS Parallel Benchmarks.

На выбранную подсистему поступал стационарный поток из M параллельных задач. Ранг r_i параллельной программы распределен по закону Пуассона с параметрами $\lambda = \{0{,}25r_0, 0{,}5r_0, 0{,}75r_0, r_0\}$, где $r_0 = 80$ – максимальное количество процессорных ядер. Заявки на обслуживание поступали через интервалы времени t, экспоненциально распределенные с параметрами $\mu = \{0{,}1$ с, 0,2 с, 0,3 с, 0,4 с$\}$.

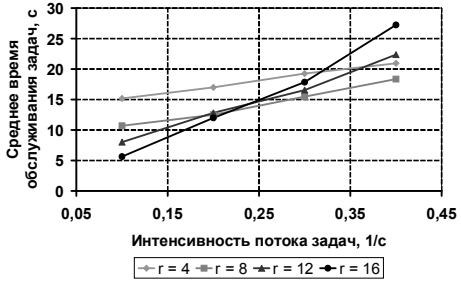


Рис. 2. Зависимости среднего времени обслуживания задач от интенсивности λ потока задач

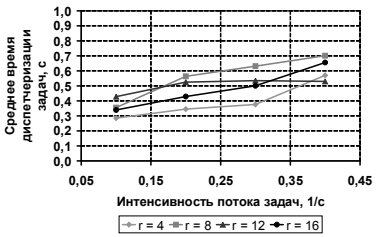


Рис. 3. Зависимости среднего времени диспетчеризации задач от интенсивности λ потока задач

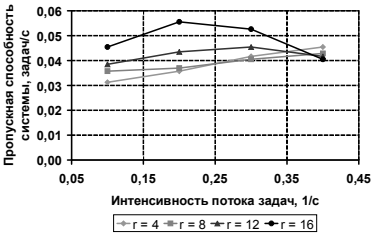


Рис. 4. Зависимости пропускной способности системы от интенсивности λ потока задач

Среднее время диспетчеризации (рис. 2, 3) задач незначительно по сравнению со временем их обслуживания. Пропускная способность системы (рис. 4) растет при увеличении интенсивности потока поступ-

ления задач. Снижение пропускной способности при больших значениях интенсивности обусловлено образованием очередей в системе пакетной обработки заданий.

Заключение

Децентрализованная диспетчеризация параллельных программ в распределенных ВС позволяет повысить живучесть последних. Среднее время обслуживания задач при децентрализованной диспетчеризации сопоставимо с централизованной. Время диспетчеризации достаточно мало по сравнению со временем выполнения программ.

В дальнейшем планируется выполнить интеграцию с программным пакетом GridWay. Планируется ввести учет производительности процессорных ядер на подсистемах и реализовать возможность выделения группы подсистем для решения параллельной программы.

Литература

1. **Хорошевский В. Г.** Архитектура вычислительных систем. М.: МГТУ им. Н. Э. Баумана, 2008.
2. **Gaweda I., Wilk C.** Grid Brokers and Metaschedulers. Market Overview // Technical Report. Kraków, 2006.

Расчет показателей эффективности функционирования распределенных вычислительных систем при решении задач потока с отказами*

В.А. Павский, К.В. Павский
КемТИ, Кемерово
Институт физики полупроводников СО РАН, Новосибирск

Предлагается математическая модель для расчета показателей осуществимости решения задач потоков на распределенных вычислительных системах (ВС) в переходном режиме функционирования. В их основе лежат вероятностные процессы теории массового обслуживания. Полученные решения представлены в аналитическом виде.

На вычислительную систему (ВС) [1] поступает пуассоновский поток простых (последовательных) задач с интенсивностью α, из которых формируется пакет. Количество задач в пакете ограничивается числом n элементарных машин ВС, выделяемых для решения поступающих задач. Если пакет сформирован, то очередная задача получает отказ и считается потерянной. Как только ВС освобождается, она приступает к обслуживанию пакета (пусть даже и не до конца сформированного) и начинается формирование очередного пакета. Время решения каждой задачи в системе является случайной величиной, распределенной по экспоненциальному закону с интенсивностью решения β. По окончании решения всех задач ВС переходит к решению задач из очередного пакета. Требуется проанализировать эффективность работы системы.

Обозначим через $P_k(t)$ – вероятность того, что в момент времени $t \in [0,\infty)$ пакет состоит из k нерешенных задач, причем $P_0(0)=1$ (т.е. в начальный момент времени пакет был пустым); $k \in E_0^n = \{0,1,...,n\}$.

Используя методы ТМО [2,3], получаем систему дифференциальных уравнений

* Работа выполнена при поддержке РФФИ (гранты №09-07-00185, №08-08-00300), Совета по грантам Президента РФ (грант №НШ-2121.2008.9).

$$\begin{cases} \dfrac{d}{dt}P_0(t) = -\alpha \cdot P_0(t) + \beta \cdot \sum\limits_{k=1}^{n} P_k(t), \\ \dfrac{d}{dt}P_k(t) = -(\alpha+\beta)\cdot P_k(t) + \alpha \cdot P_{k-1}(t), \quad k \in E_1^{n-1}, \\ \dfrac{d}{dt}P_n(t) = -\beta \cdot P_n(t) + \alpha \cdot P_{n-1}(t). \end{cases} \tag{1}$$

с начальными условиями

$$P_0(0) = 1 \ , \ P_k(0) = 0. \tag{2}$$

Условие нормировки, являющееся следствием системы уравнений (1), имеет вид:

$$\sum_{k=0}^{n} P_k(t) = 1 \ , \ \forall t \in [0, \infty) \ . \tag{3}$$

Преобразуем первое уравнение системы (1) с учетом условия (3):

$$\frac{d}{dt}P_0(t) = -(\alpha+\beta)\cdot P_0(t) + \beta \ . \tag{4}$$

Общее решение системы (1) при начальных условиях (2) записывается в виде

$$P_0(t) = \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}\cdot e^{-(\alpha+\beta)t} \ . \tag{5}$$

$$P_k(t) = \frac{\alpha^k \cdot \beta}{(\alpha+\beta)^{k+1}} - \frac{\alpha^k \cdot \beta}{(\alpha+\beta)^{k+1}} \cdot e^{-(\alpha+\beta)t} +$$

$$+ e^{-(\alpha+\beta)t} \cdot \left(\frac{\alpha^{k+1} \cdot t^k}{k! \cdot (\alpha+\beta)} - \alpha^k \cdot \beta \cdot \sum_{r=1}^{k-1} \frac{t^r}{r! \cdot (\alpha+\beta)^{k-r+1}} \right) . \tag{6}$$

$$P_n(t) = \frac{\alpha^n}{(\alpha+\beta)^n} + \frac{\alpha^{n-1}\cdot\beta}{(\alpha+\beta)^n} \cdot e^{-(\alpha+\beta)t} - \frac{\alpha^n}{(n-1)!(\alpha+\beta)} \cdot e^{-(\alpha+\beta)t} \times$$

$$\times \left(t^{n-1} + (n-1)! \cdot \sum_{k=1}^{n-1} \frac{t^{n-k-1}}{\alpha^k \cdot (n-k-1)!} \right) + \alpha^{n-1} \cdot \beta \cdot e^{-(\alpha+\beta)t} \cdot \sum_{r=1}^{n-2} \frac{t^r}{r! \cdot (\alpha+\beta)^{n-r}} +$$

$$+ \alpha^{n-1} \cdot \beta \cdot e^{-(\alpha+\beta)t} \cdot \sum_{r=1}^{n-2} \sum_{k=1}^{r} \frac{t^{r-k}}{\alpha^k \cdot (\alpha+\beta)^{n-r} \cdot (r-k)!} \ . \tag{7}$$

Иллюстрация к характеру изменения значений вероятностей $P_k(t)$ от времени t представлена на рис. 1.

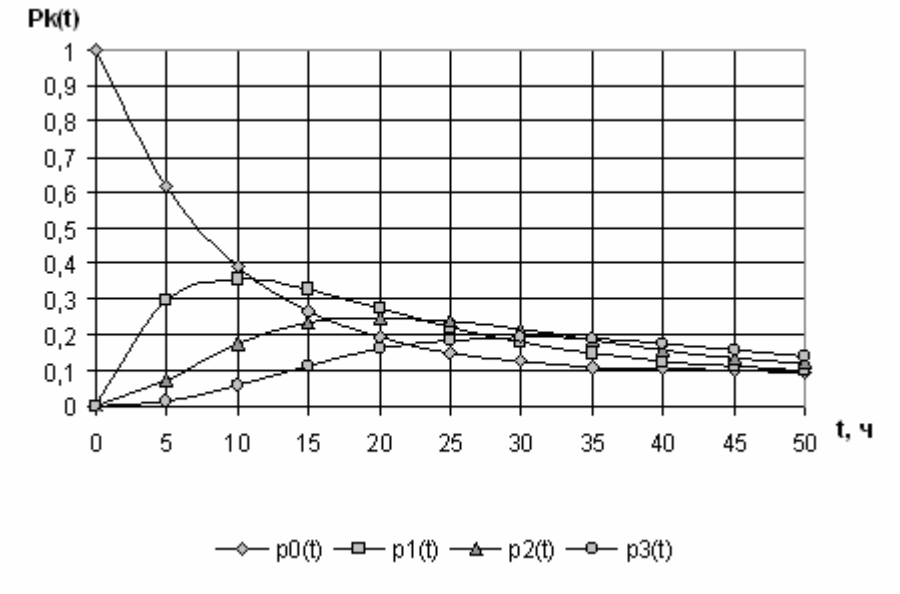


Рис. 1. Зависимость вероятностей $P_k(t)$, $k = 0,1,2,3$ от времени t при $\alpha = 0,1$ 1/ч, $\beta = 0,01$ 1/ч

Из рис. 1 видно, что стационарный режим функционирования ВС при решении задач наступает через достаточно длительный промежуток времени, $t > 45$ ч. Следовательно, для вычислительных систем, функционирующих в реальных условиях, пренебрежение зависимостью показателей эффективности от времени не допустимо.

Так как система M/M/1/n всегда эргодична, то стационарный режим ($\lim\limits_{t\to\infty} P_k(t) = P_k > 0$, $\lim\limits_{t\to\infty} P_k'(t) = 0$) всегда существует и

$$P_0 = \frac{\beta}{\alpha+\beta}\ ,\ P_k = \frac{\alpha^k \cdot \beta}{(\alpha+\beta)^{k+1}}\ ,\ k \in E_1^{n-1}\ ,\ P_n = \frac{\alpha^n}{(\alpha+\beta)^n}\ . \quad (8)$$

Как следует из (3), вероятности $P_k(t), k \in E_0^n$ составляют ряд распределения и, следовательно, представляют базис, через который выражаются показатели, характеризующие эффективность процесса решения (осуществимость решения [1. С.477], [4. С.32]) задач на ВС в момент времени t , $t \in [0,\infty)$. Рассмотрим те из них, которые можно использовать при экспресс-анализе функционирования ВС:

1) вероятность того, что пакет не пуст:

$$P_{\geq 1} = 1 - P_0 \ ;$$

2) вероятность отказа в обслуживании задачи

$$P_{OTK} = P_n \ ;$$

3) средний объем пакета

$$n = \left[\frac{\ln P_{OTK}}{\ln \alpha - \ln(\alpha + \beta)} \right], \qquad (9)$$

где $[x]$ – целая часть числа x ;

4) среднее число задач в пакете

$$M = \frac{\alpha \cdot \beta}{(\alpha + \beta)^2} \cdot \sum_{k=1}^{n-1} k \cdot \frac{\alpha^{k-1}}{(\alpha + \beta)^{k-1}} + n \cdot \left(\frac{\alpha}{\alpha + \beta} \right)^n \ ; \qquad (10)$$

5) дисперсия числа задач в пакете

$$D = \frac{\alpha \cdot \beta}{(\alpha + \beta)^2} \cdot \sum_{k=1}^{n-1} k^2 \cdot \left(\frac{\alpha}{\alpha + \beta} \right)^{k-1} + n^2 \cdot \left(\frac{\alpha}{\alpha + \beta} \right)^n - M^2 \ . \qquad (11)$$

Литература

1. **Хорошевский В.Г.** Архитектура вычислительных систем. М.: МГТУ им. Баумана, 2008. 520 с.
2. **Клейнрок Л.** Теория массового обслуживания. М.: Машиностроение, 1979. 432 с.
3. **Вишневский В.М.** Теоретические основы проектирования компьютерных сетей. М.: Техносфера, 2003. 512 с.
4. **Павский В.А., Павский К.В., Хорошевский В.Г.** Вычисление показателей живучести распределенных вычислительных систем и осуществимости решения задач // Искусственный интеллект. 2006. №4. С. 28-34.

Представление алгоритмов в технологии фрагментированного программирования

В.А. Перепелкин
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассмотрен вопрос о представлении алгоритмов в языке фрагментированного программирования с точки зрения пользователя, компилятора и исполняющего окружения. Сформулированы требования к языку и представлена неформальная модель фрагментированной программы.

Введение

Актуальной задачей является упрощение процесса создания эффективных параллельных программ [1. С. 4; 2]. Один из подходов для задач численного моделирования – это технология фрагментированного программирования [3], направленная на то, чтобы освободить программиста от низкоуровневого программирования коммуникаций, синхронизаций и распределения ресурсов во времени, а также обеспечить автоматически необходимые динамические свойства прикладной параллельной программы. Работа посвящена рассмотрению вопроса представления программ в этой технологии.

Фрагментированное программирование

Технология фрагментированного программирования предназначается для реализации сверхбольших задач численного моделирования. Она состоит в том, что параллельный алгоритм представляется как совокупность двух частей. Первая, так называемая «алгоритмическая», часть содержит представление алгоритма – данные программы, вычисления над этими данными и допустимый способ исполнения программы. Данные и вычисления агрегируются из так называемых фрагментов данных и вычислений, и такая фрагментированная структура программы сохраняется во время исполнения. Это позволяет осуществлять различные манипуляции с программой во время исполнения, например перераспределять фрагменты, планировать вычисления. Вторая часть, «рекомендательная», содержит информацию, связанную с реализацией алгоритма на конкретном вычислителе, – информацию о структуре данных и вычислений, рекомендуемый способ исполнения и

отображения программы на ресурсы суперкомпьютера. Такое представление называется фрагментированной программой (ФП).

Если алгоритм запрограммирован верно, то первой, алгоритмической, части достаточно, чтобы исполнить программу. Но эффективное ее исполнение затруднительно вследствие сложности задачи отображения программы на ресурсы параллельного суперкомпьютера во времени. Для этого требуется информация из второй, реализационной, части программы [4]. По сути, она содержит основные решения, которые требуется принять для эффективного исполнения программы.

Исполняет ФП специальное окружение, называемое исполнительной системой (ИС). ИС осуществляет отображение ФП на ресурсы суперкомпьютера и исполняет ее в соответствии с имеющимся аппаратным обеспечением и заданными рекомендациями. ИС реализует все низкоуровневые операции по управлению ресурсами, обмену данными, синхронизации и балансировке загрузки процессоров.

Создание ФП происходит на высоком уровне, слабо связанном с конкретным аппаратным окружением, жестким порядком исполнения подзадач и распределением ресурсов. Это обуславливает переносимость ФП. Для исполнения такой программы на другом классе параллельных компьютеров требуется реализовать новую ИС. Изменений в саму программу вносить не требуется, требуется только ее перекомпиляция.

Требования к представлению алгоритма

К представлению фрагментированных алгоритмов предъявляются следующие требования. Пользователь должен задавать свой алгоритм на высоком уровне, без описания низкоуровневых операций управления процессами, пересылками данных и синхронизации. Транслятор анализирует программу, оптимизирует ее, настраивает на конкретный параллельный компьютер и осуществляет предварительное планирование вычислений с учетом как структуры программы, так и рекомендаций пользователя. Представление алгоритма должно содержать всю необходимую для этого информацию. В результате транслятор генерирует исполняемую программу. Далее ИС выполняет сгенерированную программу в соответствии с составленным планом. Она осуществляет оптимизацию и принятие решений времени исполнения (распределение и перераспределение ресурсов, планирование вычислений).

ФП должна быть в значительной степени параметризована. Фрагментация данных на конкретные фрагменты данных осуществляется в нужной степени грануляции, в зависимости от имеющихся ресурсов.

Конкретные процедуры могут быть выбраны в зависимости от имеющихся библиотек и особенностей аппаратного обеспечения. Различные классы параллельных компьютеров могут использовать различные рекомендации.

Фрагментированная программа

Определим ФП как кортеж $<DF, CF, <, n, w, p, G>$.

DF – это множество фрагментов данных. Фрагмент данных – это ресурс, хранящий значения переменных алгоритма. Например, фрагмент данных может представлять некоторую область сетки в конечно-разностной схеме.

CF – это множество фрагментов вычислений. Каждый фрагмент вычислений применяется к определенным фрагментам данных и должен сработать один раз.

< – это бинарное отношение частичного порядка на множестве *CF*, которое определяет допустимый порядок срабатывания фрагментов вычислений. Если < – линейный порядок, то выполнение ФП может быть только последовательным.

Пусть $F=CF \cup DF$, а $\boldsymbol{R}$ – множество вещественных чисел.

$n: F \times F \rightarrow \boldsymbol{R}$ – функция, задающая соседство фрагментов данных и вычислений. Например, фрагменты вычислений являются соседними для фрагментов данных, которые они обрабатывают. Соседние фрагменты нужно размещать в соседних ПЭ, чтобы уменьшить накладные расходы на пересылку данных. ПЭ считаются соседними, если существует коммуникационный канал, непосредственно их соединяющий. Функция *n* задает оценку того, насколько важно размещать данную пару фрагментов рядом.

$w: F \times F \rightarrow R$ – функция веса, определяющая ресурсоемкость фрагмента. Для фрагментов данных – это их размер, а для фрагментов вычислений – оценка времени выполнения.

Функции *n* и *w* используется при планировании исполнения и динамической балансировке загрузки.

$p: F \rightarrow R$ – функция приоритета. Задает предпочтения в порядке исполнения фрагментов вычислений. Если заданный порядок < позволяет, то на исполнение первыми назначаются фрагменты вычислений с большим приоритетом (или работающие с фрагментами данных большего приоритета). Задание приоритетов позволяет направлять фронт вычислений в нужную сторону.

$G=\{ g_i \mid g_i \subset F \}$ – множество групп фрагментов. В группы объединяются фрагменты данных и вычислений, образующие относительно независимую часть программы. Информация о группах используется при планировке исполнения ФП.

Множество *DF* и частично упорядоченное множество *CF* составляют алгоритмическую часть ФП. Это тот минимум, который необходим для корректной работы программы. Остальные компоненты составляют рекомендательную часть ФП. Для одной ФП могут быть определены несколько наборов рекомендательных составляющих. Наиболее подходящий из них будет выбран при исполнении ФП.

Заключение

Представление фрагментированных алгоритмов не содержит низкоуровневых операций управления процессами, а лишь задает общий граф программы с дополнительной информацией о ее структуре. Рекомендательная информация предназначена для использования транслятором и ИС. Параметризация программы относительно размеров фрагментов данных и выбора процедур связана с конкретным синтаксическим представлением и в статье не рассматривается.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988. С. 128.
2. **Kale L.V., Bohm E., Mendes C.L. et al.** Programming Petascale Applications with Charm++ and AMPI // Petascale Computing: Algorithms and Applications, CRC Press. 2008. P. 421-441.
3. **Malyshkin V.E.** Fragmented Programming of Library Parallel Numerical Subroitines // Proceedings of the 7-th Int. conference on New Trends in Software Methodologies, Tools and Techniques, IOS Press. 2007. Vol. 193. P. 425-430.
4. **Kraeva M.A., Malyshkin V.E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers // Int. Journal on Future Generation Computer Systems. 2001. Vol. 17, №. 6, P. 755-765.

Подсистема автоматического обнаружения логических ошибок в MPI-программах информационно-вычислительного портала Кемеровского государственного университета

А.Ю. Власенко
Кемеровский государственный университет

На отладку параллельных программ в модели передачи сообщений зачастую затрачивается больше времени, чем на создание исходного кода. Поэтому требуются специальные инструментальные системы, служащие для обнаружения логических ошибок в программах пользователя. В настоящей работе описывается подсистема автоматического контроля корректности MPI-программ, входящей в информационно-вычислительный портал КемГУ.

При разработке параллельных программ для вычислительных систем с распределенной памятью возникает большое количество семантических (логических) ошибок. В дополнение ко всем ошибкам, известным при создании последовательных программ, здесь появляются также ошибки, обусловленные недетерминированным поведением совокупности одновременно работающих процессов. Все множество таких ошибок можно разделить на локальные (для обнаружения которых каждому процессу не требуется информация от других) и глобальные (включающие 2 и более процессов). К локальным ошибкам, свойственным лишь MPI-приложениям, можно отнести отсутствие в программе вызова необходимой функции (например, функции ожидания завершения неблокирующей коммуникации или завершающей для MPI-программы функции), некорректное использование внутренних объектов MPI-реализации (коммуникаторов, типов данных и т.д.), перекрывание областей памяти в неблокирующей коммуникации и основной ветви MPI-программы и многие другие ошибки. В числе глобальных можно назвать взаимные блокировки, ошибки соревнования (гонки данных), ошибки несоответствия в параметрах коммуникационных функций, вызванных различными процессами параллельной программы, и некоторые другие типы ошибок.

Существующие в данное время инструментальные системы обнаружения семантических ошибок можно классифицировать по используемым подходам. Автоматический анализ корректности параллельных программ во время исполнения выделяется среди других тем, что

посредством применения эвристических алгоритмов средство отладки может обнаружить ошибки, обусловленные недетерминизмом, для проведения анализа достаточно одного запуска параллельной программы и нет необходимости создавать программный код эталонной версии, также не нужно составлять модель программы на каком-либо псевдоязыке и ожидать, пока система произведет перебор всех вариантов исполнения параллельного приложения. Использующие этот метод инструментальные средства имеют существенные архитектурные недостатки. Так, в разработке Штутгартского центра высокопроизводительных вычислений – системе MARMOT поиском всех типов ошибок занимается выделенный узел вычислительного кластера, называемый Debug Server. При большом числе процессов параллельного приложения он может стать узким местом, негативно влияющим на производительность исполняемой программы. В системе ITAC (Intel Trace Analyzer and Collector) выявлением ошибок всех типов занимаются дополнительные потоки, каждый из которых работает в пределах MPI-процесса на узле кластера. Такая архитектура оптимальна для поиска локальных ошибок, но в поиске глобальных задействуются все узлы кластера, несмотря на то, что данную работу можно возложить на выделенный сервер.

Таким образом, актуальной является задача разработки программной системы автоматического контроля корректности MPI-приложений, которая бы несущественно влияла на производительность исполняемой параллельной программы и была неограниченно масштабируемой по числу вычислительных узлов кластерной системы.

Представляемая в данной работе система имеет следующую архитектуру [1]. Среди узлов вычислительного кластера, зарезервированных для задачи пользователя, выделяется 2 типа. Первый тип – узлы-рабочие, на каждом из которых работает MPI-процесс, исполняющий программу пользователя, и служебный поток, производящий поиск локальных ошибок. Второй тип – узлы-анализаторы, которые занимаются обнаружением глобальных ошибок. Весь процесс обработки коммуникационной функции типа «точка–точка» можно представить функциональной диаграммой (рис. 1).

При вызове функции ее параметры перехватываются при помощи профилировочного интерфейса MPI-реализации, и служебный поток производит поиск локальных ошибок. В настоящее время разработаны алгоритмы для проверки повторного использования запросов обмена неблокирующими функциями и перекрывания областей памяти между вызванной операцией и вызванными ранее не блокирующими посыл-

ками. Параллельно с этим параметры функции отправляются посредством механизма сокетов узлу-анализатору. Тот, в свою очередь, заполняет необходимые структуры данных и на основе анализа ранее сделанных записей в этих структурах делает вывод о наличии либо отсутствии глобальных ошибок. К ним относятся ошибки несоответствия, изменение данных во время неблокирующей коммуникации основной ветвью программы (в случае MPI_Recv), некоторые типы дедлоков и гонок данных. После передачи служебного сообщения на узел-анализатор основной поток на узле-рабочем производит непосредственно саму вызванную коммуникацию и по ее завершении отсылает узлу-анализатору сообщение об этом. Отсюда следует, что дополнительная нагрузка на сеть в этом случае будет заключаться в отправке параметров коммуникации на узел-анализатор (без самих данных, которые передаются/принимаются в данной операции) и отправке сигнала о завершении коммуникационной операции.

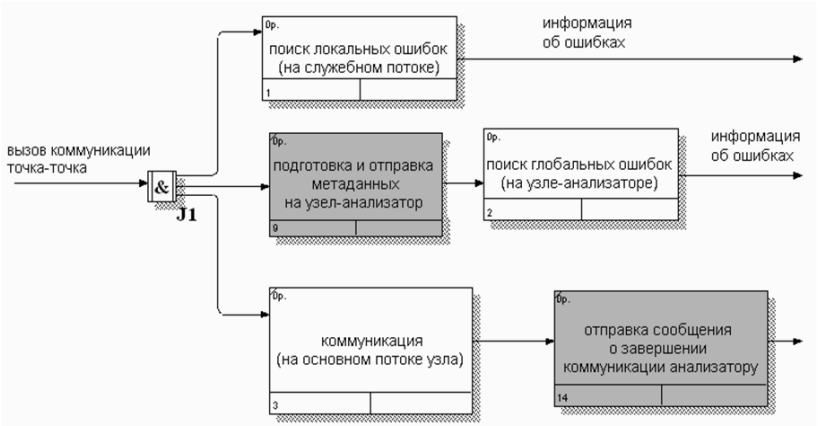


Рис. 1. Функциональная модель системы

По окончании работы программы на основе анализа структур данных, заполняемых на узлах-рабочих и анализаторах, делается вывод о наличии таких ошибок, как непарные коммуникации (например, MPI_Send, для которого не было вызвано соответствующего MPI_Recv), отсутствие функций ожидания или тестов на завершение для неблокирующих коммуникаций, зависание MPI-процесса на некоторой функции.

Система планируется к интеграции в информационно-вычислительный портал Кемеровского государственного университета [2]. В настоящее время пользователь имеет возможность загрузить на портал исходные коды своего приложения с файлами начальных дан-

ных и создать задание на компиляцию, выбрав необходимые библиотеки и компилятор. Для того чтобы воспользоваться системой автоматического контроля корректности, в число этих параметров также следует добавить возможность слинковать программу пользователя с библиотекой собственной разработки, осуществляющей все дополнительные действия на узле-рабочем. После того как пользователь отправил задание на компиляцию и получил с портала исполняемый файл, он может создать задание на запуск, выбрав вычислительный кластер и указав количество узлов для MPI-программы. Если требуется использовать описанную систему, то пользователю также необходимо указать число узлов-анализаторов среди общего числа узлов кластера, которые резервируются для задачи. Когда из базы данных менеджер вычислительных ресурсов выбирает данное задание на запуск и отправляет на требуемый кластер, то по протоколу ssh на выбранном числе узлов запускается процесс узлов-анализаторов, а на остальных – MPI-реализация загружает параллельную программу. Найденные ошибки узлы-анализаторы и служебные потоки записывают в текстовые файлы, которые после работы MPI-приложения собираются на одном из анализаторов в один файл и помещаются вместе с файлами результатов в базу данных.

Литература

1. **Власенко А.Ю.** Модель масштабируемой системы автоматического контроля корректности параллельных программ // Вестник НГУ. Сер. Информационные технологии. Новосибирск: Изд-во НГУ, 2009. Т. 7, вып. 4. С. 53-65.
2. **Афанасьев К.Е.** Информационный портал параллельных вычислений для организации учебно-научного процесса [Электронный ресурс]/ К.Е. Афанасьев, А.В. Демидов, С.В. Стуколов //http://tm.ifmo.ru/tm2008/db/doc/get_thes.php?id=330. 2008.

Задание управления во фрагментированных программах сетями Петри и проверка корректности управления

В.В. Ларин
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Представлены средства для задания во фрагментированной программе управления и проверки корректности этого управления при помощи сетей Петри. Рассматривается проблема отображения вычислительного алгоритма на ресурсы вычислительной системы и поиск следующих ошибок управления: дедлок, недостижимость нужного результата, недетерминированность результата, бесконечные циклы.

Фрагментированная программа (ФП) [1] представляется в виде набора фрагментов, каждый из которых является совокупностью вычислительной процедуры, входных и выходных переменных. Программистом задается порядок исполнения фрагментов.

Если есть алгоритм, вычисляющий некоторую функцию, то для задания ФП, реализующей алгоритм, необходимо описать множество переменных, множество вычислительных фрагментов, реализующих соответствующие операции, управление, определяющее порядок исполнения фрагментов, и задать отображение переменных и фрагментов на вычислительные ресурсы компьютера.

Отдельного рассмотрения заслуживает проблема задания управления в ФП. Она тесно связана с задачей отображения переменных и фрагментов на вычислительные ресурсы. На средства задания управления предъявляются следующие требования: удобство и возможность контроля корректности управления.

В работе исследуется применение сетей Петри (СП) для задания управления. Пусть переходы СП соответствуют фрагментам вычислений, а места СП – переменным. Наличие фишки в месте говорит о готовности переменной. Это позволяет задать необходимое потоковое управление. Ограничения в ресурсах могут требовать задания прямого управления, ужесточающего потоковое. Так, если при вычислении выражения $x = a \cdot b + c \cdot d$ ресурсы памяти не ограничены, то программа для вычисления выражения выглядит, как представлено на рис. 1, а её управляющая сеть представлена на рис. 2. Если же компьютер имеет

только ячейки памяти *x,m1,m2,* то программа изменится (рис. 3). Изменится также и управляющая сеть (рис. 4).

m1=a
m2=b
m3=c
m4=d
*m5=m1*m2*
*m6=m3*m4*
m7=m5+m6

Рис. 1. Программа без ограничений

m1=a
m2=b
*x=m1*m2*
m1=c
m2=d
*m1=m1*m2*
x=x+m1

Рис. 2. Программа с ограничениями

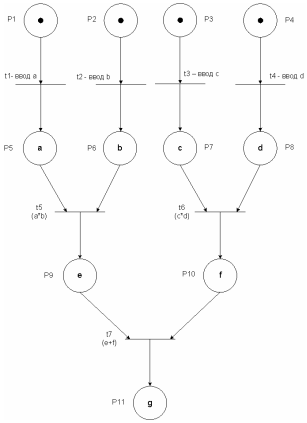


Рис. 3. Управляющая сеть Петри для программы без ограничений

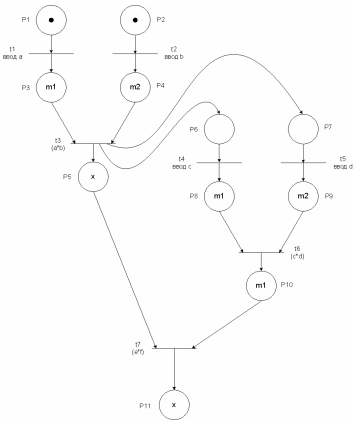


Рис. 4. Управляющая сеть Петри для программы с ограничениями

Видно, что из-за ограничения по ресурсам управляющая сеть перестраивается, и некоторые места сети уже не соответствуют переменным алгоритма. Так, места *P*6 и *P*7 отвечают за освобождение ячеек памяти *m*1 и *m*2. СП для программы без ограничений задает потоковое управление, СП для программы с ограничением по ресурсам задает более строгое потоковое управление. С точки зрения пользователя, программа выполняет один и тот же алгоритм, и значит обе управляющие сети должны вести себя одинаковым образом, т.е. если в первой сети переход A срабатывает после перехода B, то и во второй сети переход A должен сработать после B. Соблюдение этого условия можно автоматически проверять.

Для задания программы нужно описать отображение переменных и вычислений на ресурсы машины. Для программы без ограничений оно представлено в табл. 1, а для программы с ограничениями – в табл. 2.

Таблица 1. Программа без ограничений

Переменные алгоритма	Вычислительные ресурсы
A	M1
B	M2
C	M1
D	M2
E	X
F	M1
G	X

Таблица 2. Программа с ограничениями

Переменные алгоритма	Вычислительные ресурсы
A	M1
B	M2
C	M3
D	M4
E	M5
F	M6
G	M7

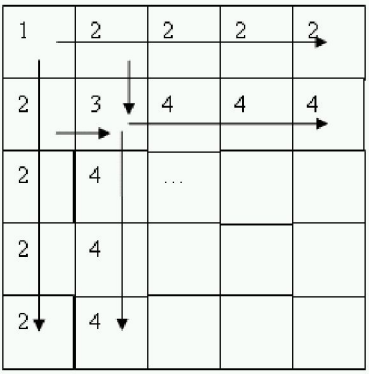


Рис. 5. Взаимодействия фрагментов в алгоритме LU-разложения

Рассмотрим задание управления на примере алгоритма фрагментированного LU-разложения [2]. На рис. 5 показана структура взаимодействия вычислительных фрагментов. Квадрат матрицы означает фрагмент вычислений.

Исходя из этих зависимостей, зададим управление СП (рис. 6). Приведенное управление статическое, но бывает, что нужно рекомендовать динамически генерируемое управление. Такая необходимость возникает, когда нельзя статически принять оптимальное решение по управлению. Для ускорения работы программы выгодно увеличение числа фрагментов, готовых к вычислениям. В этих целях фрагменты, соответствующие переходам, помеченным цифрой 2, должны исполниться быстрее помеченных цифрой 1. Однако, увеличивая число исполняющихся фрагментов, нужно помнить об ограничении ресурсов.

При записи реальных программ размер СП сильно увеличивается. Для решения этой проблемы в управляющей СП были выделены блоки повторяющихся операций.

Был разработан язык задания СП, в котором места ассоциируются ресурсными переменными, а переходы – с фрагментами вычислений.

Таким образом, был разработан язык записи ФП, который состоит из управляющей СП, таблицы отображения переменных на ресурсы и таблицы фрагментов.

При задании управления необходимо обеспечивать его корректность. Корректное управление – управление, обеспечивающее получение целевых переменных в ходе вычислений. Из-за некорректного управления при исполнении могут возникнуть следующие ошибки: дедлок, недостижимость нужного результата, недетерминированность результата, бесконечные циклы. При нахождении этих ошибок рассматривается вопрос о достижимости конечного результата.

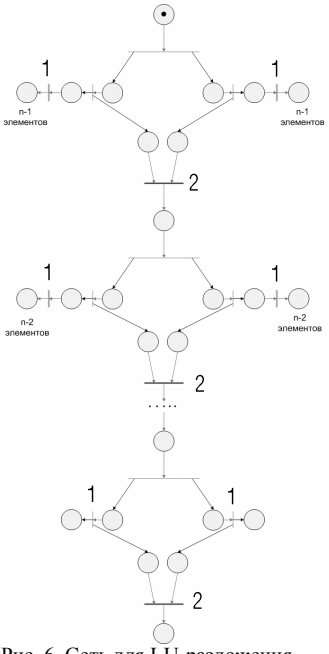


Рис. 6. Сеть для LU-разложения

Для проверки управления на корректность строится дерево достижимости, представляющее собой множество всех состояний программы. При анализе управления дерево достижимости проверяется на наличие требуемого состояния программы.

Разработана многопоточная реализация алгоритма построения дерева достижимости. Каждый поток строит свое поддерево, производится динамическая балансировка загрузки.

Заключение. Были разработаны язык для задания управления и частичного отображения ресурсов, транслятор СП, модуль проверки управления на ошибки, параллельная реализация алгоритма построения дерева достижимости.

В дальнейшем планируется разработка средства, реализующего удобное задание управления для программ, имеющих большую управляющую сеть; разработка визуальной среды для задания управления во ФП; интеграция с другими частями системы фрагментированного программирования.

Литература

1. **Arykov S., Malyshkin V.** Asynchronous Language and System of Numerical Algorithms Fragmented Programming // In proc. of the PaCT 2009, Novosibirsk, Russia, August 31 – September 4, 2009. Springer, LNCS 5698. 2009. P. 1-7.
2. **Чаюк К.Г**. Фрагментированный алгоритм LU-разложения и его параллельная реализация // Четвертая Сибирская школа-семинар по параллельным и высокопроизводительным вычислениям. Томск: Дельтаплан, 2008. С. 157-167.

Runtime-система FP RTS управления исполнением фрагментированных программ на мультикомпьютере

Г.А. Щукин
Новосибирский государственный технический университет

Разработанная runtime-система FP RTS обеспечивает параллельное исполнение фрагментированных программ на мультикомпьютере. В статье показано устройство FP RTS и приведены тесты исполнения фрагментированных вычислительных алгоритмов.

Введение

Развитие и распространение высокопроизводительных параллельных вычислительных систем ставит задачу не только создания соответствующего эффективного параллельного программного обеспечения для решения вычислительных задач, но и оптимального использования вычислительного оборудования.

Одной из технологий, позволяющей не только упростить процесс создания параллельных программ и вывести его на новый уровень, но и обеспечить оптимальное использование вычислительных ресурсов, является технология фрагментированного программирования [1, 2].

Во фрагментированном программировании программа состоит из фрагментов данных (аналог переменных программы) и фрагментов кода (действия, выполняемые над данными); через применение определенных фрагментов кода к определенным фрагментам данных получаются фрагменты вычислений, а исполнение программы заключается в запуске всех ее фрагментов вычислений.

FP RTS

Наличие runtime-системы [3] для исполнения фрагментированных программ позволяет автоматически использовать динамические свойства фрагментированной программы: подстройка под ресурсы вычислительной системы для оптимального исполнения через изменение размера и числа фрагментов данных; динамическая балансировка нагрузки через перераспределение фрагментов вычислений между вычислительными узлами; коммуникации (обмен данными) на фоне вычислений. Таким образом, программист выполняет только описание фрагментированной программы, тогда как исполнение программы вынесено отдельно в runtime-систему.

Разработанная автором runtime-система FP RTS осуществляет динамическое управление исполнением фрагментированных программ на вычислительных машинах широкого класса – мультипроцессорах (SMP) и мультикомпьютерах (MPP, кластеры). Для функционирования runtime-системы на каждом вычислительном узле запускается менеджер FP RTS (рис. 1), который можно разделить на 3 подсистемы.

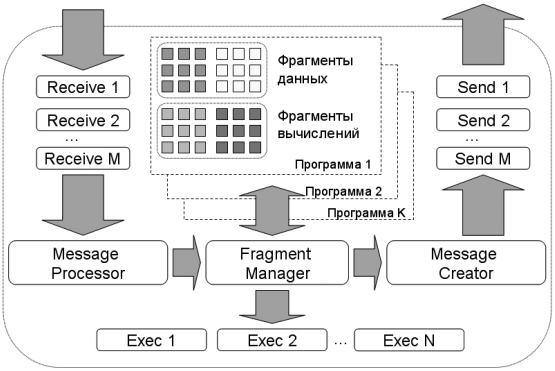


Рис. 1. Структура FP RTS на одном вычислительном узле

Подсистема хранения содержит исполняющиеся в текущий момент фрагментированные программы.

В начале работы каждой программы FP RTS осуществляет автоматическое распределение фрагментов данных и вычислений по узлам. Таким образом, на каждом узле хранится только часть фрагментов данных и вычислений из всей программы; некоторые фрагменты данных могут дублироваться на разных узлах. Во время исполнения программы фрагменты данных и вычислений могут перемещаться между узлами.

Подсистема исполнения содержит пул исполнительных потоков для запуска фрагментов вычислений на узле.

Каждый фрагмент вычислений имеет входные и выходные фрагменты данных. Минимальное условие для запуска фрагмента вычислений – готовность всех его входных фрагментов данных. После исполнения фрагмента вычислений вычисленные им выходные фрагменты данных могут быть переданы на вход другим фрагментам вычислений. Кроме зависимостей по данным, порядок исполнения фрагментов вычислений в общем случае не детерминирован, но может регулироваться с помощью системы динамических приоритетов.

На каждом узле готовые к исполнению фрагменты собираются в группы (группировка по фрагментам кода) и для каждой группы вы-

числяется динамический приоритет на основе приоритетов содержащихся в ней фрагментов (приоритет фрагмента может зависеть от его «важности», вычислительного веса и т.п.). Приоритет также вычисляется и для всей программы. В итоге на исполнение берутся фрагменты с максимальным приоритетом из программы и группы с максимальными приоритетами.

Подсистемы хранения и исполнения в одиночку могут обеспечить функционирование FP RTS и запуск фрагментированных программ на машинах с общей памятью (SMP).

Сетевая подсистема отвечает за взаимодействие между менеджерами FP RTS на разных узлах, а также за пересылку фрагментов данных и вычислений между узлами. Вся информация передается в виде сообщений. На каждом узле запускаются отдельные потоки для отправки и приема сообщений. Потоков отправки/приема может быть несколько, что обеспечивает параллельные отправку/прием нескольких сообщений.

Для описания и запуска фрагментированных программ в FP RTS разработан специальный интерфейс (API) – набор классов и функций, использующихся для описания фрагментов вычислений и фрагментированных программ, запуска/остановки runtime-системы и загрузки в нее программ для исполнения. Интерфейс может использоваться как программистами фрагментированных программ, так и компилятором/транслятором с языка фрагментированного программирования высокого уровня.

Результаты экспериментов

Тестирование проводилось на кластере НКС-30Т; на каждом вычислительном узле кластера находятся два 4 ядерных процессора Intel Xeon E5450 3 GHz с общей памятью 16 Gb. На каждом вычислительном узле запускалось по одному менеджеру FP RTS (процессу).

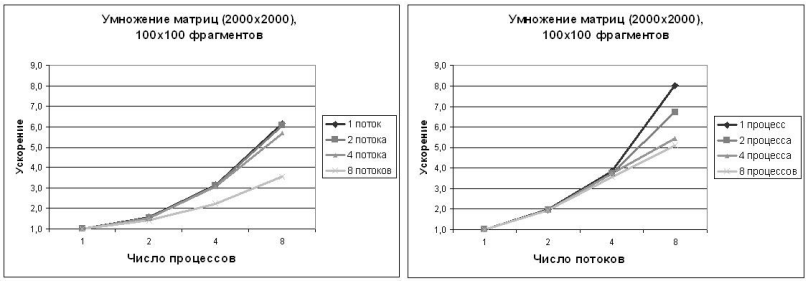


Рис. 2. Умножение матриц: ускорение

Фрагментированное умножение матриц представлено на рис. 2.

Ускорение достигается как при увеличении числа вычислительных узлов, так и при увеличении числа исполнительных потоков. Ускорение тем лучше, чем больше доля вычислений и, соответственно, меньше доля накладных расходов на управление.

Фрагментированное LU-разложение представлено на рис. 3-4.

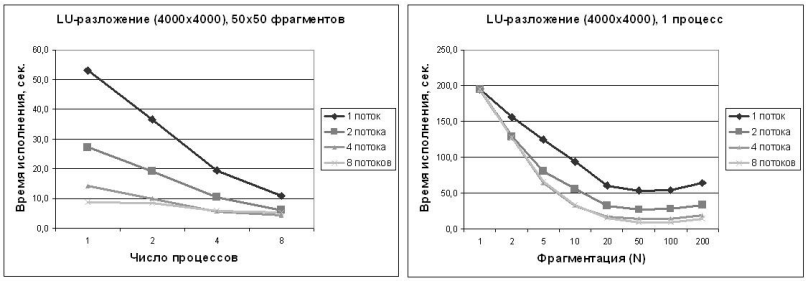


Рис. 3. LU-разложение: время исполнения и фрагментация

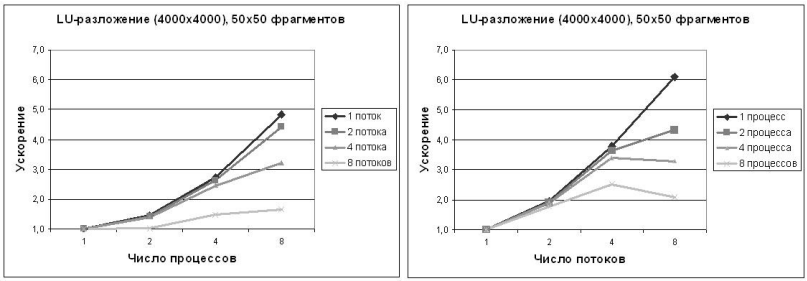


Рис. 4. LU-разложение: ускорение

За счет фрагментации время исполнения программы существенно снижается. Несмотря на предусматриваемый фрагментированным алгоритмом большой объем передаваемых между узлами фрагментов данных, достигается неплохое ускорение при увеличении числа процессов и потоков. Некоторое падение ускорения при большом числе процессов и потоков можно объяснить снижением относительной доли вычислений для параллельного исполнения всеми потоками, а также техническими ограничениями вычислительного узла (невозможностью предоставить ресурсы всем потокам).

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988. С. 128.
2. **Malyshkin V.E.** Fragmented Programming of Library Parallel Numerical Subroutines // Proceedings of the 7-th Int. conference on New Trends in Software Methodologies, Tools and Techniques, IOS Press, 28-30 September, 2007. Dubai, 2007. Vol. 193. P. 425-430.
3. **Kalgin K.V., Malyshkin V.E., Nechaev S.P., Tschukin G.A.** Runtime System for Parallel Execution of Fragmented Subroutines// PACT-2007, LNCS. 2007. Vol. 4671. P. 544-552.

Примеры применения сплайнов, сохраняющих интеграл, при обработке изображения

О.П. Федорова
Томский государственный университет

Рассматриваются примеры применения сплайн-функций при работе с изображением.

Рассмотрим задачу сжатия и восстановления изображения. Изображение определим как функцию $f(x, y)$, где x, y – координаты на плоскости. Значение функции f в любой точке, задаваемой парой координат (x, y), называется интенсивностью или уровнем серого изображения в этой точке. Пусть изображение $f(x, y)$ задано в прямоугольной и ограниченной области D. Будем строить аппроксимацию на представлении кубического сплайна дефекта 1 по переменным x и y через базисные сплайны [1]

$$S(x,y)= \sum_{p=i-1}^{i+2} \sum_{q=j-1}^{j+2} z_{p,q}(f) B_p(x) B_q(y) \qquad (1)$$

на сетке, которая представляет собой множество точек пересечения двух семейств параллельных прямых, проведенных параллельно осям координат через точки $-\infty < x_0 < x_1 < \ldots < x_n < \infty$ и $-\infty < y_0 < y_1 < \ldots < y_m < \infty$. При этом прямоугольник $D = [x_0; x_n] \times [y_0; y_m]$ разбивается на прямоугольные ячейки $\Omega_{ij} = [x_i; x_{i+1}] \times [y_j; y_{j+1}]$, $i = \overline{0, n-1}$; $j = \overline{0, m-1}$. Здесь $B_i(x)$, $B_j(y)$ – базисные кубические сплайны [1]. При аппроксимации изображения $f(x, y)$ для вычисления коэффициентов $z_{ij}(f)$ построим формулу по аналогии с формулой, точной на кубических многочленах [1]. Получим для коэффициентов сплайна (1) на Ω_{ij} следующее выражение:

$$z_{pq} = \frac{16}{9} f_{p,q} - \frac{2}{9} f_{p-1,q} - \frac{2}{9} f_{p+1,q} -$$

$$- \frac{2}{9} f_{p,q-1} + \frac{1}{36} f_{p-1,q-1} + \frac{1}{36} f_{p+1,q-1} -$$

$$- \frac{2}{9} f_{p,q+1} + \frac{1}{36} f_{p-1,q+1} + \frac{1}{36} f_{p+1,q+1}.$$

Цифровое изображение состоит из конечного числа элементов, которые представляются в виде матрицы чисел. Координаты (x, y) – целые числа: номер строки и номер столбца матрицы. Элементы матрицы – значения яркости в градациях серого, принимают значения от 0 до 255. Если используется модель RGB представления цветов, то изображение – это совокупность из трех матриц. Рассмотрим пример черно-белого изображения (см. рис. 1, *в*), для которого n = 726 и m=473.

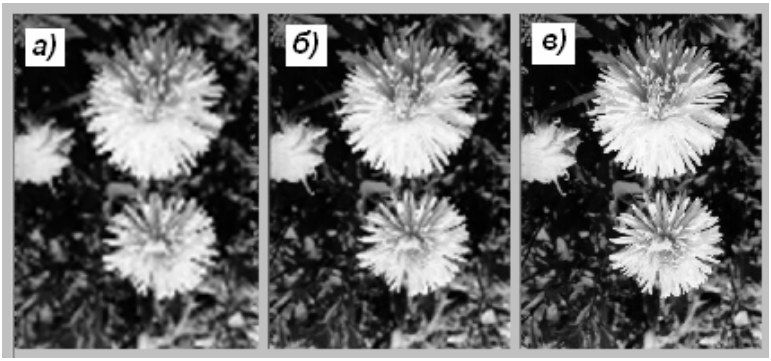


Рис. 1. Исходное изображение (*в*) и восстановленные (*а*) и (*б*)

На рис. 1, *а* и *б* показано изображение, восстановленное с помощью сплайна при шаге дискретизации $\Delta x = \Delta y = 6$ и $\Delta x = \Delta y = 4$, соответственно. Ошибка, определяемая как стандартное отклонение, составляет примерно 6% в случае *а* и 4% – в случае *б*.

Важное значение при работе с изображением имеет распределение уровня интенсивности, вычисляемое для заданного изображения. Для приближения распределений можно использовать сплайны, сохраняющие интеграл. Получаемые в этом случае функции можно рассматривать как некоторые распределения.

Пусть задана функция одной переменной $f(x), x \in D \subset R$ такая, что существует конечный интеграл от функции по области D:

$$\left| \int_D f(x)\,dx \right| < M \ .$$

Сплайн $S(x)$ на отрезке $[x_0; x_n]$ выражается линейной комбинацией нормализованных базисных B сплайнов

$$S(x) = \sum_i z_i(f) B_i(x). \qquad (2)$$

Выберем $z_i(f)$ так, чтобы значения определенного интеграла по области задания приближаемой функции и сплайна были равными [2. С. 62].

На промежутке $[x_i; x_{i+1}]$ кубический сплайн $S(x)$ дефекта 1 (2) запишется в виде

$$S(x) = \sum_{p=i-1}^{i+2} z_p B_p(x) .$$

Для нахождения коэффициентов z_i получена [2. С. 63] система линейных уравнений:

$$a_i z_{i-1} + b_i z_i + c_i z_{i+1} + d_i z_{i+2} = \int_{x_i}^{x_{i+1}} f(x)dx \ , \ i = \overline{0, n-1} \ , \qquad (3)$$

здесь $a_i = \int_{x_i}^{x_{i+1}} B_{i-1}(x)dx$, $b_i = \int_{x_i}^{x_{i+1}} B_i(x)dx$, $c_i = \int_{x_i}^{x_{i+1}} B_{i+1}(x)dx$,

$$d_i = \int_{x_i}^{x_{i+1}} B_{i+2}(x)dx.$$

Система (3) дает n условий для определения n+3 коэффициентов. Дополнительные условия можно задать различным способом, исходя из знаний о свойствах приближаемой функции $f(x)$. В настоящей работе использованы условия равенства функции и сплайна в точках x_0, x_{n-1}, x_n. Получившаяся система разрешается методом прогонки

$$z_i = W_i z_{i+1} + Q_i z_{i+2} + R_i .$$

Исследования прогоночных коэффициентов W_i и Q_i показывают, что для i=6, 7, 8, … значения W_i отличается от –1,101020, а Q_i отличается от значения –0,101020 на величину, меньшую 10^{-6}. Этот факт может быть использован при практическом применении сплайна (2). При приближении распределения яркости вычисления нужно проводить в точках составляющих 1/2, 1/4, 3/4, 1/8, 3/8, 5/8, 7/8 шага, поэтому для сокращения времени, затрачиваемого на вычисление значения сплайна, можно использовать таблицы значений базисных сплайнов в этих точках: $B_{i-1}(x_i+h/2) = B_{i+2}(x_i+h/2) = 1/48$, $B_i(x_i+h/2) = B_{i+1}(x_i+h/2) = 23/48$, $B_{i-1}(x_i+h/4) = B_{i+2}(x_i+3h/4) = 9/128$, $B_{i-1}(x_i+3h/4) = B_{i+2}(x_i+h/4) = 1/384$, $B_i(x_i+h/4) = B_{i+1}(x_i+3h/4) = 235/384$, $B_i(x_i+3h/4) = B_{i+1}(x_i+h/4) = 121/384$.

На рис. 2 штриховой линией показан полигон распределения яркости, построенный по 726*423 пикселям изображения (см. рис. 1, *в*). Кружками даны значения сплайна, построенного по 16 промежуткам.

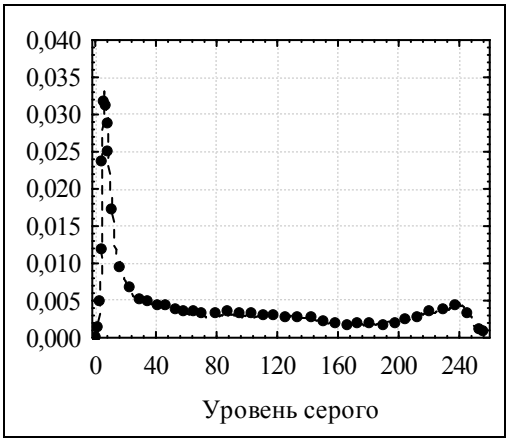


Рис. 2. Полигон распределения яркости черно-белого изображения

Ошибка, вычисляемая по среднему квадратичному отклонению, в данном примере около 5%. Таким образом, для восстановления исходного распределения с достаточной точностью можно сохранять 19 значений коэффициентов сплайна вместо 256 значений частот яркости отдельных пикселей.

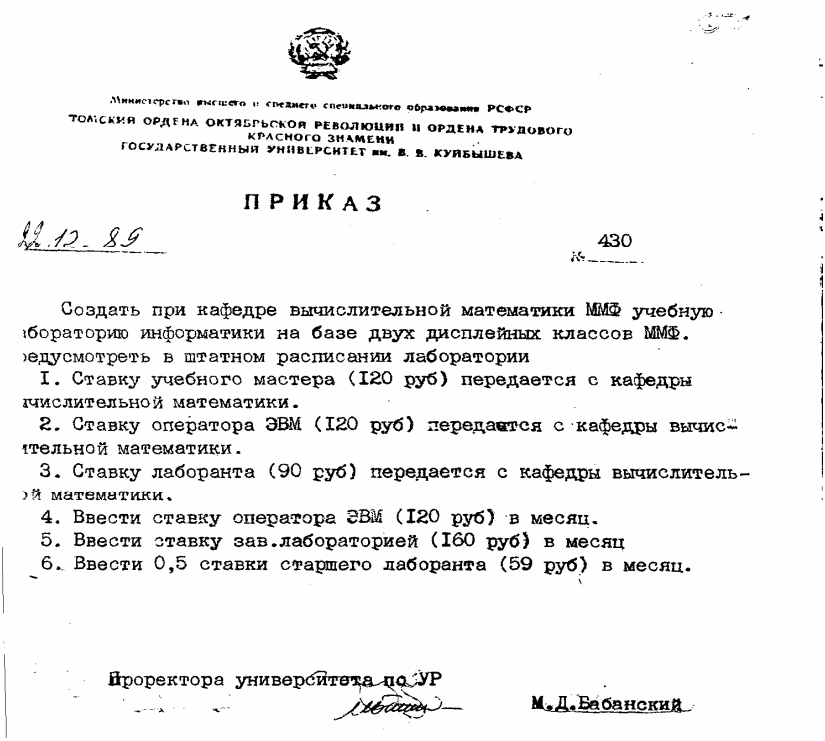
Литература

1. **Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л.** Методы сплайн-функций. М.: Наука, 1980. 352 с.
2. **Федорова О.П.** Об одном подходе к приближению функций сплайнами. // Вестник Томского университета. Математика и механика. 2008. № 2(3). С. 61-66.

Учебной лаборатории кафедры вычислительной математики и компьютерного моделирования 20 лет

М.Д. Михайлов, К.Д. Овчинникова
Томский государственный университет

22 декабря 2009 г. исполнилось 20 лет учебной лаборатории кафедры вычислительной математики и компьютерного моделирования (УЛ КВМиКМ) ММФ. Приказ №430 от 22.12.89 г. (фото 1) проректора ТГУ по учебной работе Михаила Демьяновича Бабанского гласил: «Создать при кафедре вычислительной математики ММФ учебную лабораторию информатики на базе двух дисплейных классов ММФ». В штатном расписании лаборатории было предусмотрено две ставки оператора ЭВМ, ставка зав. лабораторией, 0,5 ставки старшего лаборанта, ставка лаборанта и ставка учебного мастера.



Министерство высшего и среднего специального образования РСФСР
ТОМСКИЙ ОРДЕНА ОКТЯБРЬСКОЙ РЕВОЛЮЦИИ И ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. В. В. КУЙБЫШЕВА

ПРИКАЗ

22.12.89 430

Создать при кафедре вычислительной математики ММФ учебную лабораторию информатики на базе двух дисплейных классов ММФ.
Предусмотреть в штатном расписании лаборатории
1. Ставку учебного мастера (120 руб) передается с кафедры вычислительной математики.
2. Ставку оператора ЭВМ (120 руб) передается с кафедры вычислительной математики.
3. Ставку лаборанта (90 руб) передается с кафедры вычислительной математики.
4. Ввести ставку оператора ЭВМ (120 руб) в месяц.
5. Ввести ставку зав.лабораторией (160 руб) в месяц
6. Ввести 0,5 ставки старшего лаборанта (59 руб) в месяц.

Проректора университета по УР М.Д.Бабанский

Фото 1

К этому времени изменилась технология преподавания дисциплин на кафедре вычислительной математики. Вместо калькуляторов и машин типа ЕС-ЭВМ в учебный процесс стали внедряться ПЭВМ. В 1989 г. парк машин учебной лаборатории пополнился персональными компьютерами «ДВК-3» и «Искра» в количестве 10 и 8 ПЭВМ соответственно.



ДВК-3

Машины были переданы заводом математических машин (ЗММ) в обмен на выпускников кафедры, распределенных на работу в эту организацию. Изменились планы преподавания базовых курсов «Информатика» и «Методы вычислений». Проведение практических занятий по этим дисциплинам было ориентировано на персональную вычислительную технику.

Постепенно на смену ПК «ДВК-3» и «Искра» пришли машины класса IBM второго, третьего и четвертого поколений, а также Pentium'ы. В конце 90-х годов прошлого столетия факультет существенно обновил парк персональных компьютеров. К этому времени в УЛ КВМиКМ было три дисплейных класса (ДК): 314, 316, 319 (ауди-

тории второго учебного корпуса). Ниже приведена комплектация классов того периода персональными компьютерами.

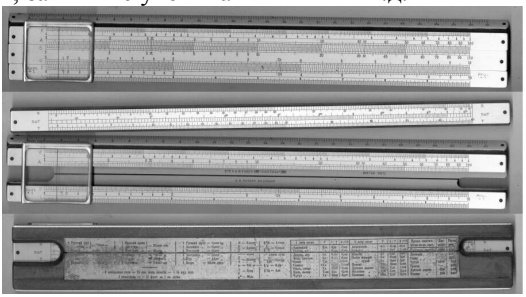
Конфигурация аудитории 319: 10 персональных компьютеров «Intel-Celeron» (тактовая частота 1200 МГц, оперативная память 260 Мб, жесткий диск (винчестер) 40 Гб).

Конфигурация аудитории 316: 10 персональных компьютеров Athlon (процессор АМД Athlon XP, тактовая частота 1800 МГц, оперативная память: 225 Мб – 6 штук, оперативная память 128 Мб – 4 штуки, жесткий диск (винчестер) 40 Гб).

Конфигурация аудитории 314: четыре ПЭВМ Duron (процессор АМД Duron, тактовая частота 650 МГц, оперативная память 128 Мб, жесткий диск (винчестер) 40 Гб); одна ПЭВМ Pentium (процессор Pentium MMX, тактовая частота 200 МГц, оперативная память 64 Мб, жесткий диск (винчестер) 5 Гб); одна ПЭВМ Pentium (тактовая частота 150 МГц, оперативная память: 32 Мб, жесткий диск (винчестер) 808 Мб); две ПЭВМ Pentium (тактовая частота 166 МГц, оперативная память: 48 Мб, жесткий диск (винчестер) 1 Гб); одна ПЭВМ Pentium (тактовая частота 133 МГц, оперативная память 32 Мб, жесткий диск (винчестер) 695 Мб).

В 2007 г. в аудиториях 314 и 316 проведен капитальный ремонт и каждый дисплейный класс пополнился современными ПЭВМ. Кроме 12 машин для работы студентов в классе установили персональный компьютер и для преподавателя, ведущего занятия. В классах появились интерактивные доски, проекторы.

С каждым годом увеличивается объем работы УЛ КВМиКМ. Наряду с учебными занятиями по базовым курсам «Компьютерные науки» и «Методы вычислений» (практика) это чтение лекций по спецкурсам кафедры с использованием презентаций, защита курсовых и дипломных работ, проведение тестирования знаний у студентов ММФ, научных семинаров, занятий с учениками ФМШ и т.д.



Логарифмическая линейка

Заметим, что совершенствование вычислительной техники, ее использование в учебных занятиях в ТГУ шло параллельно с развитием вычислительной математики. И продолжительность этого процесса не ограничивается двумя десятилетиями, а перешагнуло второе столетие. За это время от работы на обычных счетах, логарифмической линейке, дифференциальном анализаторе и интеграфе пользователи техники университета сделали гигантский шаг к современным персональным машинам и суперкомпьютерам.



Николай Никанорович Горячев
(1883–1940)

Значительную роль в исследованиях по вычислительной математике в Томском университете сыграл выпускник физико-математического факультета МГУ Н.Н. Горячев.

Еще в 1920 г., впервые в Томске, Николай Никанорович начал читать на физико-математическом факультете курс лекций «Методы приближенных вычислений». В своих исследованиях по астрономии ученый постоянно использовал вычислительную математику и математическую статистику для усовершенствования вычислительной схемы эволюций орбит тел Солнечной системы.

В 1931 г. на физико-математическом факультете ТГУ открывается механико-математическое отделение, на котором появилась специальность по прикладной математике. По этой специальности, наряду с классическими курсами, читались лекции по теории конечных разностей, приближенному интегрированию дифференциальных уравнений, вычислительной технике и др.

Важно отметить, что в начале 30-х годов ТГУ был единственным вузом в стране, где на механико-математическом отделении готовились специалисты по прикладной (вычислительной) математике.

Из сказанного следует, что интерес к вычислительной математике и ее применению возник в ТГУ еще в начале прошлого столетия.

Трудно переоценить вклад в развитие вычислительной математики профессора Льва Александровича Вишневского, который, работая в ТГУ с 1925 г., много внимания уделял подготовке специалистов по прикладной (вычислительной) математике. Он стал инициатором создания при университете вычислительного бюро. Здесь начинались работы по расчету траекторий полета снарядов. Для этого использовались вычислительные схемы метода Адамса и обратное интерполирование. С 1932 по 1937 г. Лев Александрович был директором научно-исследовательского института математики и механики (НИИММ) при ТГУ, руководил отделом прикладной математики.



Лев Александрович Вишневский
(1887 – 1938)

Одними из первых работ того времени по разностным уравнениям явились статья А.К. Минятова «Сравнительное исследование дифференциальных и разностных уравнений» и публикация М.С. Горохова «Связь графического метода интегрирования с численным». Обе статьи вышли в свет в «Известиях НИИММ».

В сентябре 1957 г. на механико-математическом факультете Томского государственного университета по приказу министра высшего образования СССР В.П. Елютина была открыта кафедра прикладной и вычислительной математики (КПиВМ), первая за Уралом (фото 2).

П Р И К А З № 209.

по Томскому госуниверситету им. В. В. Куйбышева

„29" октября 1957 г.

§ 2.-

В соответствии с приказом МВО СССР № 984 от 21.IX.57г. организовать в составе механико-математического факультета кафедру прикладной и вычислительной математики в следующем составе:

доц.Г.А.Бюлер, асс.Семухина Н.В., Михайлов А.М., ст.лаб. Сазонова Э.С.

Отделу кадров оформить личное дело доц.Бюлера для утверждения в МВО.

Ректор Томского государственного университета профессор-доктор (А.Бунтин)

Фото 2

П Р И К А З № 53

по Томскому госуниверситету им. В. В. Куйбышева

„6" апреля 1959 г.

§ - 1.

В целях выполнения приказа Министерства об организации учебных вычислительных лабораторий:

1. Передать комнату № 211 кафедре прикладной и вычислительной математики к 6.IV.59г.
2. Декану ГГФ доц.Юртусову М.П. освободить комнату № 211 к этому же сроку.

Ректор Томского государственного университета профессор-доктор /А.Бунтин/

Фото 3

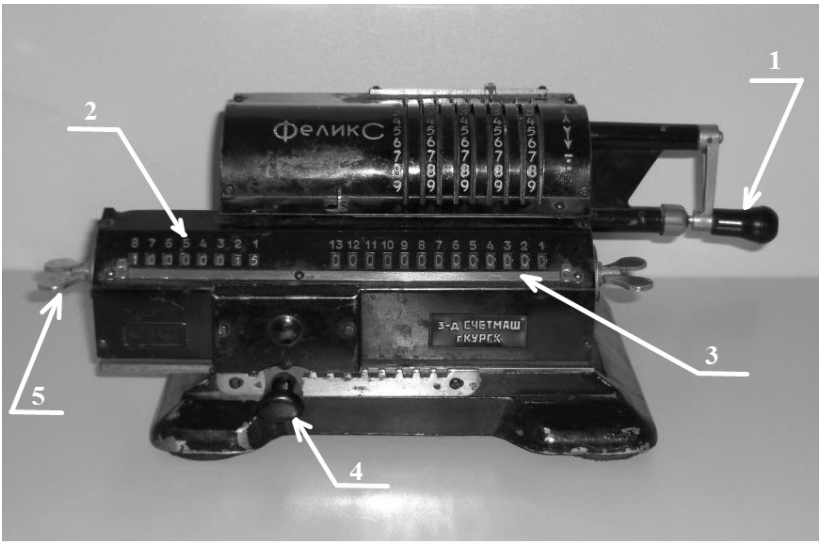
6 апреля 1959 г. приказом №53 ректора ТГУ А. Бунтина был открыт кабинет вычислительной техники при КПиВМ (фото 3):

«В целях выполнения приказа Министерства об организации учебных вычислительных лабораторий:

1) Передать комнату №211 кафедре прикладной и вычислительной математики к 6 апреля 1959 г.

2) Декану ГГФ Кортусову М.П. освободить комнату №221 к этому же сроку».

В конце пятидесятых – начале шестидесятых годов студенты ММФ выполняли здесь задания по «Методам вычислений» на арифмометрах «Феликс» и «Рейн-металлах», составлявших в то время парк кабинета. Арифмометр является простейшей цифровой машиной, производящей математические операции с числами.



Арифмометр «Феликс»

Авторы надеются, что некоторое количество строк, посвященных выполнению арифметических операций на арифмометре, заинтересует нынешнее поколение студентов. Читайте и сравнивайте возможности «Феликса» и современных ПЭВМ. Начнем с операции сложения. Для ее выполнения набираются исходные данные с помощью установочных рычагов. Сначала происходит набор первого слагаемого, которое вращением рукоятки *1* на один оборот в направлении, указанном стрелкой со значками «+» и «*», оказывается в счетчике результатов. А в счетчике оборотов *2* будет находиться *1*. Точно так же набирается

второе слагаемое, и с ним производится аналогичное действие. В счетчике результатов *3* получается сумма двух слагаемых.

Чтобы произвести операцию вычитания, поступают следующим образом. Сначала в счетчик результатов *3* переносится уменьшаемое (аналогично первому слагаемому), затем на установочных рычагах набирается вычитаемое и рукоятка *1* поворачивается на один оборот в направлении, указанном стрелкой со значками «–» и «:». Результат операции располагается в счетчике результатов *3*.

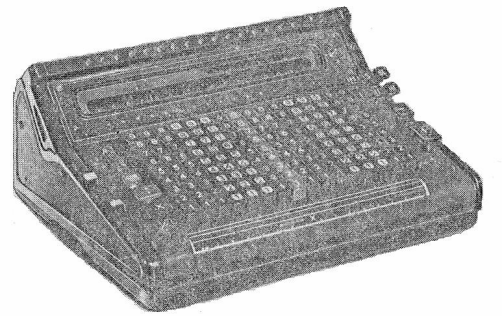
С помощью установочных рычагов на арифмометре можно набрать число, имеющее 9 разрядов. Счетчик результатов содержит 13 разрядов.

При выполнении операции умножения сначала набирают на арифмометре один из сомножителей. Далее, вращая рукоятку *1* в том же направлении, что и при сложении, делают столько оборотов, какова последняя цифра второго множителя. В счетчике результатов *3* будет стоять первый множитель, умноженный на последнюю цифру второго сомножителя, в счетчике оборотов *2* получается число, равное количеству сделанных оборотов. Далее каретка передвигается на один разряд вправо при помощи рычажка *4,* и по указанному правилу осуществляется умножение на предпоследнюю цифру второго сомножителя. Вновь перемещение каретки на один разряд вправо и т.д. вплоть до умножения на первую цифру второго сомножителя. Результат получается в счетчике результатов *3*, а второй сомножитель будет стоять в счетчике оборотов *2*.

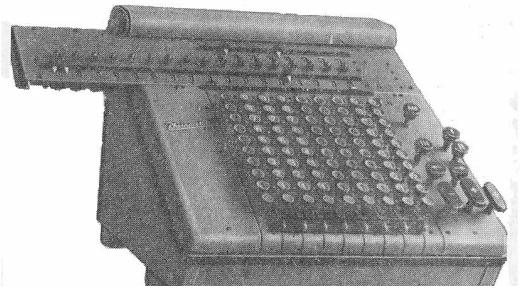
Для выполнения операции деления надо сначала передвинуть каретку вправо на такое количество разрядов, которое равно числу разрядов в частном. Затем перенести делимое в счетчик результатов *3*. Старший разряд делимого устанавливается на установочных рычагах как можно левее, но чтобы при этом все делимое полностью перешло в счетчик результатов *3*. Далее гасится в счетчике оборотов *2* цифра *1* с помощью барашка *5*. После этого с помощью установочных рычагов набираем делитель. При наборе делителя необходимо следить за тем, чтобы его старший разряд стоял против старшего разряда делимого, если делимое больше делителя. В противном случае на один разряд правее старшего разряда делимого (при этом и делимое, и делитель рассматриваются как целые числа, равные номеру самого левого из использованных установочных рычагов). Операция деления осуществляется повторным вычитанием. Рукоятка вращается в том же направлении, что и при вычитании, до тех пор, пока делитель, число установленное на рычагах, не станет больше числа в счетчике результатов. Если будет сделан лишний оборот, то можно услышать звонок. Тогда

делается оборот в обратном направлении. Далее каретка перемещается на один разряд влево, и процесс повторяется. Это продолжается до тех пор, пока каретка не займет крайнее левое положение. Частное будет находиться в счетчике оборотов *2*, а в счетчике результатов *3* – остаток.

Арифмометр приносит пользу при проведении расчетов: производится меньше записей на бумаге, вычисления носят менее утомительный характер, время каждого вычисления уменьшается и зависит от скорости вращения рукоятки пользователем.



Клавишная вычислительная машина «Мерседес»



Клавишная вычислительная машина «Рейн-металл»

Интересный факт: за рабочий день на арифмометре можно выполнить около 750 операций умножения пятизначных чисел на четырехзначные и около 570 операций деления таких же чисел.

«Рейн-металл», «Мерседес» и другие машины незначительно отличаются от арифмометра. В этих клавишных машинах вместо рычажкового набора используется клавишный, а ручной привод заменен электромотором. Имеются и другие усовершенствования.

Применение этих машин резко увеличивает производительность труда: можно выполнить около 1200 операций умножения пятизнач-

ных чисел на четырехзначные и приблизительно 1210 делений таких же чисел. К недостаткам работы на них следует отнести:

1) достаточно частую запись промежуточных результатов на бумаге;

2) повторный набор на клавиатуре новых данных и нажатие клавиши соответствующей арифметической операции после выполнения очередного действия;

3) значительный шум, который стоит в аудитории при выполнении арифметических операций, особенно умножения и деления.

Нынешнему поколению студентов, работающих на персональных компьютерах, трудно себе представить, какой грохот стоял в кабинете, когда несколько человек одновременно выполняли умножение или деление и особенно когда происходило деление на нуль.

В штате кабинета было всего два человека: ст. лаборант А.П. Мусатов и лаборант В.А. Пшеничников. Позднее, с 1960 г, в кабинете лаборантом работал «легендарный» Владимир Ваганян, помогавший ремонтировать вычислительную технику кабинета до 1970 г. С 1961 г. в кабинете работали М. Бухтяк (лаборант) и Н. Кардаш (ст. лаборант). После создания факультета прикладной математики и кибернетики был организован кабинет вычислительной техники, который обслуживал несколько факультетов и, в частности, ММФ. Кабинет располагался в подвальном помещении второго учебного корпуса, а позже в БИНе (третий учебный корпус ТГУ).

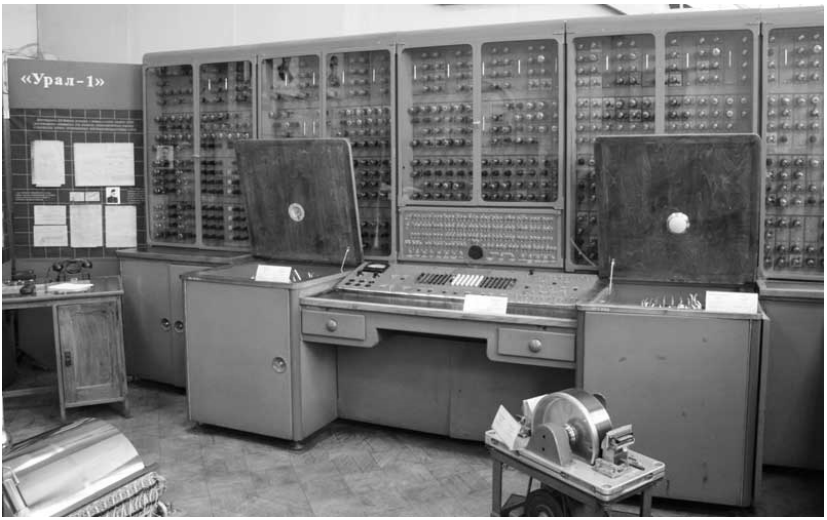


Калькулятор «Электроника 63-34»

К этому времени парк кабинета составляли калькуляторы. Последние совершенствовались. На «Электронике 63-34» можно было составлять простейшие программы, выполнять арифметические действия, хранить промежуточные

результаты в памяти калькулятора, вычислять различные тригонометрические функции и т.д.

Практические занятия по программированию, связанные с работой на ЭВМ и реализацией программ решения прикладных задач, проводились на ММФ ТГУ сначала на ЭВМ первого поколения «Урал-1». Она предназначалась для решения сравнительно несложных научно-технических задач. ЭВМ «Урал» занимала значительную территорию, отличалась большим потреблением энергии, невысокой надежностью и слабым программным обеспечением. Быстродействие машины составляло 2-3 тысячи операций в секунду, емкость оперативной памяти (ОП) – 2048 машинных слов длиной 48 двоичных знаков. Машина занимала правое крыло коридора второго этажа главного учебного корпуса, рядом с Гербарием.

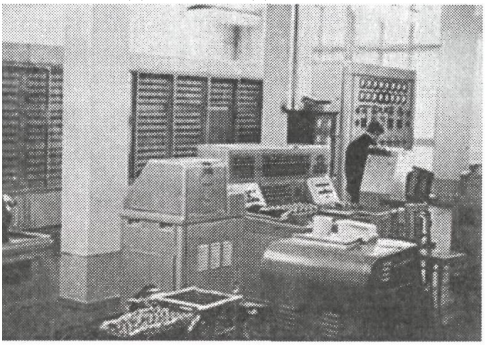


Урал-1

Когда решалась нетривиальная задача по уже отлаженной программе, программисту необходимо было приходить в 10 часов вечера на ЭВМ, устанавливать перфоленту с программой в читающее устройство, запускать программу и ждать результата, сидя рядом на стуле, откликаясь на звонок компьютера при прерываниях. И так до утра.

Первые поколения студентов-вычислителей проходили производственную практику по программированию в ВЦ МГУ, а с 1965 г. – на ЗММ г. Томска, где была установлена ЭВМ первого поколения М-20.

Ее быстродействие составляло 20 тысяч операций в секунду, а емкость ОП – 4096 машинных слов длиной 48 двоичных знаков.



М-20 (1958 г.)

Приказом ректора от 30 марта 1965 г. здесь был открыт вычислительный центр (ВЦ) ТГУ. Исполняющим обязанности начальника ВЦ стал Ю.П. Кунченко, а материально ответственным – М.Ф. Максимов.

Ограниченность возможностей ЭВМ первого поколения заставляла первых программистов относиться к составлению программ, как к искусству. Этому принципу учил вычислителей первый преподаватель КПиВМ по программированию, автор уникального учебного пособия «Программирование для трехадресной вычислительной машины» Моисей Аронович Тынкевич. Позднее появился учебник В.Ф. Ляшенко «Программирование на ЦВМ с системой команд типа М-20».

В 1970 г. при ТГУ организовали ВЦ университета, подчиненный ректору ТГУ. ЭВМ М-20 послужила исходной моделью для ряда машин второго поколения – БЭСМ-3М; БЭСМ-4 и М-220 и др., которые пополнили парк ЭВМ ВЦ Томского университета.

В период с 1974 по 1992 г. в ТГУ активно используются ЭВМ четвертого поколения на базе больших интегральных схем (БИС). Это машины класса ЕС-ЭВМ.

Для более активного использования современной вычислительной техники ВЦ ТГУ в учебной и научной деятельности, а также по управлению работой вуза приказом ректора №463 от 31 октября 1983 г. в ТГУ был организован соответствующий совет, куда вошли деканы факультетов. На самом вычислительном центре в этот период времени был организован дисплейный класс виртуальных машин, где студенты ММФ в течение семестра выполняли задания по «Методам вычислений», а летом проходили производственную практику по программированию.

С 1989 г. на механико-математическом факультете наступило время персональных ЭВМ. Сегодня в арсенале факультета шесть дисплейных классов и, в частности, 314, 316, 319 второго учебного корпуса и 138 НИИПММ, входящие в УЛ КВМиКМ. В каждом классе (314 и 316) находится 12 пользовательских ПЭВМ, каждая из которых комплектуется: LCD монитором Belinea 17”, процессором Intel Celeron D, тактовая частота 3 ГГц, оперативной памятью 2 Гб, жестким диском (винчестером) 160 Гб, видеокартой ATI Radeon 9200; и одна преподавательская ПЭВМ с LCD монитором Belinea 17”, процессором Intel Pentium 4, тактовая частота 3 ГГц, оперативной памятью 2 Гб, жестким диском (винчестером) 160 Гб, видеокартой NVidia GeForce 5500.

Аудитория 319 укомплектована четырьмя устаревшими ПЭВМ с ЭЛТ монитором Samsung 17”, процессором AMD Sempron 2500+ с тактовой частотой 1,4 ГГц, оперативной памятью 256 Мб, жестким диском (винчестером) 40 Гб, видеокартой NVidia GeForce FX 5200.



Аудитория 314

Шесть компьютеров, пополнивших парк ДК 319 в ноябре 2009 г., можно отнести к условно современным. В комплект каждой ПЭВМ входит: LCD Монитор Acer 17”, процессор Intel Celeron E1400 (дву-

ядерный) 2 ГГц, оперативная память 1 Гб, жесткий диск 320 Гб, видеокарта NVidia GeForce 8400.

На механико-математическом факультете функционирует локальная сеть, которая объединяет более 50 рабочих компьютеров, расположенных в дисплейных классах учебной лаборатории КВМиКМ, кафедрах и деканате факультета. При помощи виртуальных ethernet адаптеров (TAP) и программных мостов к сети факультета также присоединены: дисплейный класс №138, расположенный в НИИПММ, и дисплейный класс, расположенный в общежитии №6. В распоряжении факультета имеется беспроводная Wi-Fi сеть, работающая по стандартам b/g/n, подключение к которой возможно из любого места левого крыла третьего и четвертого этажей второго корпуса.

Локальная сеть имеет доменную структуру со строгим разделением прав. Каждому студенту и сотруднику выдается свой уникальный логин для работы за компьютером, подключенным к сети. Пользователь получает возможность пользоваться сетевыми ресурсами факультета, а именно сетевым диском по протоколу SMB из локальной сети факультета и по протоколу FTP из любого места томской сети.

Локальная сеть факультета управляется серверами, работающими под операционной системой FreeBSD. Центральный маршрутизатор, обеспечивающий выход в сеть Internet, имеет следующую конфигурацию: 2xIntel Pentium III 780MHz, 1Gb RAM, SCSI raid. Пользовательские данные хранятся на сервере со следующей конфигурацией: 4xIntel(R) Xeon(R) CPU E5335 2.00GHz, 4Gb RAM, SAS Raid. Для резервного копирования данных используется сервер.

Центральный маршрутизатор Wi-Fi сети – Cisco Linksys wrt320n.

Все серверы оснащены источниками бесперебойного питания APC на 1,5 кВт на случай внезапного отключения электричества. Автономная работа возможна на срок порядка 30 минут.

Дисплейные классы имеют ограниченный выход в сеть Internet (доступ возможен только к разрешенным и научно-образовательным ресурсам), однако данное ограничение не распространяется на компьютеры кафедр и деканат.

Из дисплейных классов учебной лаборатории КВМиКМ организован доступ преподавателей и студентов факультета к мощному вычислительному кластеру ТГУ СКИФ «Cyberia», который используется для проведения занятий по параллельным вычислениям.

Количество вычислительных узлов на кластере 283. Они соединяются высокопроизводительной сетью – Infiniband (Qlogic Infinipath). Каждый узел оснащен процессором 2xIntel Xeon 5150 (двуядерным)

2,66 ГГц, оперативной памятью 4 Гб, жестким диском (винчестером) 80 Гб.

За 20 лет работы учебной лаборатории ее услугами воспользовалось около 1300 выпускников – математиков и механиков механико-математического факультета.

В учебной лаборатории КВМиКМ в настоящее время работает 5 лаборантов (2,5 ставки). Это, как правило, студенты старших курсов ММФ, 3 старших лаборанта (1,5 ставки) из числа аспирантов КВМиКМ, 1 программист (1 ставка), 1 ведущий электроник (1 ставка), учебный мастер (1 ставка) и заведующий учебной лабораторией (1 ставка). Коллектив учебной лаборатории молод, но объем качественно выполняемой им работы достаточно велик (только учебной нагрузки на каждый класс приходится в среднем 58–60 часов).



Кластер СКИФ Cyberia

В основе деятельности всех работников лаборатории лежит стремление помочь факультету готовить специалистов высокого класса, умеющих работать на самой современной вычислительной технике.

С вводом на ММФ стандартов третьего поколения роль УЛ КВМиКМ в научно-исследовательском и учебном процессах будет только возрастать.

Литература

1. **Березин И.С., Жидков Н.П.** Методы вычислений. М.:Изд-во физ.-мат. лит-ры, 1959. Т. 1. 464 с.
2. **Берцун В.Н., Михайлов М.Д., Тынкевич М.А.** Кафедре вычислительной математики и компьютерного моделирования 45 лет // Вестник ТГУ: бюл. оперативной научной информации. 2003. № 10. С. 7-16.
3. **Круликовский Н.Н.** Из истории развития математики в Томске. Томск: Изд-во ТГУ, 2006. 174 с.
4. **Томский университет.** 1880—1980. Томск: Изд-во Том. ун-та. 430 с.
5. **Фоминых С.Ф., Некрылов С.А., Берцун Л.Л., Литвинов А. В.** Профессора Томского университета : Библиографический словарь. Томск: Изд-во Том. ун-та, 1998. Т. 2. 544 с.
6. **Фоминых С.Ф., Некрылов С.А., Берцун Л.Л. и др.** Профессора Томского университета: Библиографический словарь. Томск: Изд-во Том. ун-та, 2001. Т. 3. 532 с.
7. **Фоминых С.Ф., Некрылов С.А., Берцун Л.Л. и др.** Профессора Томского университета: библиографический словарь. Томск: Изд-во Том. ун-та, 2003. Т. 4. Ч. 1. 534 с.
8. **Развитие математики, механики и кибернетики в Томском университете:** Сб. ст. Томск: Изд-во Том. ун-та, 1981. 124 с.
9. **Гладких Б.А.** Информатика от Абака до Интернета. Введение в специальность: Учеб. пособие. Томск: Изд-во НТЛ, 2005. 484 с.

Научное издание

Пятая Сибирская конференция
по параллельным и высокопроизводительным вычислениям

Томск, 1−3 декабря 2009 года

Редактор **В.Г. Лихачёва**

Компьютерная верстка **Н.Н. Богословский**

Лицензия ИД 04617 от 24.04.01 г. Подписано в печать 11.06.06 г.

Формат 60x84 1/16. Бумага офсетная №1.

Печать офсетная. Гарнитура «Times».

Печ. л. 11,6; уч.-изд. л. 10,8; усл. печ. л. 10,6.

Заказ Тираж 100 экз.

ОАО «Издательство ТГУ», 634029, г. Томск, ул. Никитина, 4

Типография «Иван Федоров», 634003, г. Томск, Октябрьский взвоз, 3