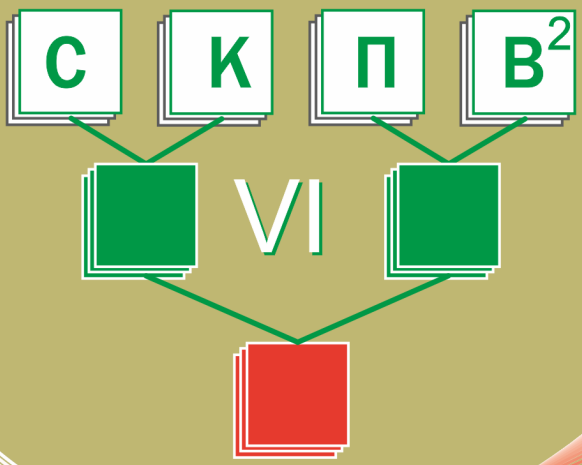
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

VI СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ



Материалы конференции

15 - 17 ноября 2011, Томск

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ШЕСТАЯ СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 15–17 ноября 2011 года

ИЗДАТЕЛЬСТВО ТОМСКОГО УНИВЕРСИТЕТА
2012

УДК 519.6
ББК 22.19
Ш 51

Ш 51 **Шестая** Сибирская конференция по параллельным и высокопроизводительным вычислениям / Под ред. проф. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2012. – 190 с.
ISBN 978 – 5 – 7511 – 2063 – 4

В сборнике содержатся материалы Шестой Сибирской конференции по параллельным и высокопроизводительным вычислениям, проходившей 15–17 ноября 2011 г. в Томском государственном университете при поддержке Министерства образования и науки РФ, Суперкомпьютерного консорциума России 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 – 2063 – 4

СОДЕРЖАНИЕ

Суперкомпьютерное образование в Сибири: проект подготовки кадров в области суперкомпьютерных технологий

В.П. Демкин
Томский государственный университет

Рассматривается межрегиональный проект подготовки кадров в области суперкомпьютерных технологий.

В течение двух лет на территории Сибирского федерального округа реализуется крупный межрегиональный проект по созданию системы подготовки высококвалифицированных кадров и образовательных услуг в области суперкомпьютерных технологий (СКТ). Он является одним из важнейших проектов в рамках приоритетных направлений развития экономики России «Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения», инициированного в 2010 г. Московским государственным университетом им. М.В. Ломоносова в соответствии с решением Комиссии при Президенте России по модернизации и технологическому развитию экономики России [1]. Проект разработан в 2010 г. ведущими в области суперкомпьютерных технологий университетами – членами Суперкомпьютерного консорциума университетов России: Московским государственным университетом им. М.В. Ломоносова, Национальным исследовательским Нижегородским государственным университетом им. Н.И. Лобачевского, Национальным исследовательским Томским государственным университетом, Национальным исследовательским Южно-Уральским государственным университетом, Национальным исследовательским Санкт-Петербургским государственным университетом информационных технологий, механики и оптики – и рассчитан на три года [2].

Основными задачами проекта являются:

− создание инфраструктуры научно-образовательных центров суперкомпьютерных технологий (НОЦ СКТ) на базе вузов, имеющих ресурсы и значительный опыт в развитии суперкомпьютерных технологий;

− разработка учебно-методического обеспечения системы подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий;

− реализация образовательных программ подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий;

− развитие интеграции фундаментальных и прикладных исследований и образования в области суперкомпьютерных технологий. Обеспечение взаимодействия с РАН, промышленностью, бизнесом;

− расширение международного сотрудничества в создании системы суперкомпьютерного образования;

− разработка и реализация системы информирования общества о достижениях в области суперкомпьютерных технологий.

Таким образом, проект представляет собой систему комплексных мероприятий для достижения стратегической цели – создания национальной системы подготовки кадров в области суперкомпьютерных технологий [3].

Научно-методическое руководство и организацию совместной работы участников проекта в СФО осуществляет Научно-образовательный центр (НОЦ) «СКТ-Сибирь», созданный в рамках проекта на базе Томского государственного университета. НОЦ «СКТ-Сибирь» – один из восьми научно-образовательных центров, являющихся основными инфраструктурными элементами проекта. Сегодня в состав НОЦ «СКТ-Сибирь» входят 15 организаций образования, науки, промышленности и бизнеса, в том числе университеты Сибирского федерального округа, имеющие высокопроизводительные вычислительные ресурсы и осуществляющие подготовку кадров в области СКТ.

Участниками проекта в 2011 г. являлись научные и образовательные учреждения высшего профессионального образования Сибирского федерального округа: Сибирский федеральный университет, Национальный исследовательский Новосибирский государственный университет, Новосибирский государственный технический университет, Национальный исследовательский Томский государственный университет, Кемеровский государственный университет, Алтайский государственный университет, Омский государственный университет им. Ф.М. Достоевского, Красноярский государственный педагогический университет им. В.П. Астафьева, Институт вычислительного моделирования СО РАН.

В результате выполнения проекта 240 студентов вузов Сибири освоили начальную подготовку по параллельным вычислениям, 63 студента физико-математических и естественнонаучных специальностей – целевую интенсивную подготовку по применению суперкомпьютеров в решении научно-технических задач, 38 преподавателей вузов

прошли повышение квалификации в области суперкомпьютерных технологий. Разработаны свод знаний и предложения в федеральные образовательные стандарты по физико-математическим направлениям, включающие современные знания в области СКТ. Разработаны совместные образовательные программы: «Введение в суперкомпьютерные технологии», «Параллельные вычисления на кластерах», учитывающие специфику вузов – участников проекта и достижения их научных школ. Разработаны и подготовлены к изданию учебники и учебные пособия для студентов вузов.

Анализ результатов проекта показал эффективную совместную работу команды сибирских университетов.

Сегодня перед вузами – основоположниками суперкомпьютерного образования в России стоит задача – подготовить к 2015 г. не менее 15000 специалистов в области суперкомпьютерных технологий. Актуальность этой задачи обусловлена исключительной важностью суперкомпьютерных технологий в повышении уровня технологического уклада, конкурентоспособности российской экономики и качества жизни населения страны.

Суперкомпьютеры предназначены для решения сверхсложных в вычислительном плане задач, когда недостаточно ресурсов обычных персональных компьютеров. Сферы применения суперкомпьютеров разнообразны: создание новых лекарств, мониторинг климатических изменений, инженерные расчеты, оптимизация транспортных потоков, прогнозирование в финансовых и экономических областях, управление телекоммуникационным трафиком, моделирование техногенных и природных процессов, мониторинг и прогнозирование социально-экономического развития – везде использование суперкомпьютера значительно повышает точность и эффективность работ, сокращает сроки создания сложных изделий, минимизирует стендовые испытания и соответственно существенно снижает стоимость разработок. Сегодня экономическое лидерство развитых стран во многом обусловлено использованием суперкомпьютеров в науке, промышленности и производстве, а также применением суперкомпьютеров в стратегически важных областях экономики [4]. Именно поэтому СКТ определены Президентом России в качестве приоритетных направлений развития национальной экономики и, следовательно, СКТ являются одним из ключевых факторов развития экономики Сибири.

В Стратегии социально-экономического развития Сибири до 2020 г. заложены основные принципы и приоритеты модернизации базовых отраслей экономики, основанные на инновационных моделях высокотехнологичных и наукоемких производств, требующих приме-

нения высокопроизводительных систем и технологий [5]. По оценкам экспертов, реализация крупных инвестиционных проектов по развитию высокотехнологичных отраслей экономики в регионах Сибири потребует к 2015 г. не менее 400 тыс. новых рабочих мест на предприятиях. С другой стороны, прогноз баланса потребности экономики Сибирского федерального округа в кадрах показывает, что существует огромный дефицит специалистов для высокотехнологичных отраслей, который сегодня существующая система высшего образования не может восполнить. Более того, сегодняшние выпускники вузов, пришедшие на предприятие, не готовы сразу включиться в производственный процесс, и им требуется дополнительная профессиональная подготовка.

В связи с этим кадровое обеспечение высокотехнологических отраслей экономики Сибири приобретает особое значение.

Несмотря на очевидный прогресс в развитии высокопроизводительных вычислительных систем в перспективе развития суперкомпьютерного образования в Сибири и внедрения СКТ в отрасли экономики, существует ряд проблем.

1. По оценкам экспертов, на сегодня с учетом динамики развития IT-отрасли в СФО спрос на специалистов в области информационных технологий на порядок превышает существующие возможности вузов. Кроме того, развитие высокотехнологичных отраслей экономики, определенных Стратегией социально-экономического развития Сибири до 2020 г., потребует существенного увеличения планов подготовки IT-специалистов, владеющих суперкомпьютерными технологиями.

Сегодня из 110 государственных вузов СФО подготовку IT-специалистов разного уровня в области суперкомпьютерных технологий осуществляют 12 университетов. Высокопроизводительные вычислительные ресурсы Сибири представлены суперкомпьютерами, созданными на базе университетов и академических институтов в Томске, Красноярске, Новосибирске, Иркутске, Омске, Кемерове, с общим ресурсом более 150 трлн операций в секунду.

Наиболее мощный суперкомпьютер Национального исследовательского Томского государственного университета с производительностью 62,3 трлн операций в секунду занимает 10-е место в списке наиболее быстродействующих вычислительных систем России и СНГ (так называемый список TOP-50, www.supercomputers.ru) и является самым мощным за Уралом суперкомпьютером.

Опыт совместной работы сибирских вузов в данном проекте показал, что для решения задач кадрового обеспечения экономики Сибири необходима качественно новая модель образования, основанная на

междисциплинарном подходе, преемственности разноуровневых образовательных программ, сетевых технологиях, системе прямых и обратных связей вузов и промышленных предприятий и включения предприятий в разработку и реализацию образовательных программ.

2. Второй не менее важной проблемой в развитии суперкомпьютерных технологий является отсутствие спроса на специалистов в области СКТ со стороны промышленных предприятий и бизнеса. На сегодня основными потребителями выпускников вузов являются учреждения академического, отраслевого и вузовского секторов науки. Ведущие предприятия отраслей экономики регионов Сибири не владеют информацией о достижениях в области суперкомпьютерных технологий, передовых разработках российских ученых, которые имеют прикладное значение и могут принести значительный экономический эффект. Следовательно, они не могут сформулировать требования к формированию новой модели специалиста, его профессиональным компетенциям. Более того, предприятия не готовы принять на производство выпускников вузов в области суперкомпьютерных технологий. На предприятиях, за малым исключением, отсутствует подготовленный инженерно-технический персонал, который мог бы заниматься внедрением СКТ в производство и ставить соответствующие задачи молодым специалистам. Следовательно, необходимо решить задачу качественной переподготовки и повышения квалификации инженерно-технического персонала и руководителей предприятий базовых отраслей промышленности как интеллектуальной основы для модернизации производственных процессов и приема молодых специалистов.

3. Создание устойчивой системы подготовки кадров связано с решением проблемы непрерывного междисциплинарного образования, начальным звеном которого является школа. Сегодня отсутствует какое-либо систематическое образование школьников в области суперкомпьютерных технологий. Необходима разработка курсов по СКТ и их учебно-методического обеспечения для школьников и педагогов, которые бы включали не только теоретические знания, но и учебный лабораторный практикум – школьные лаборатории.

В июне 2011 г. на совете ассоциации научных и образовательных учреждений «Сибирский открытый университет» обсуждался вопрос о преподавании современных знаний в педагогических вузах, отражающих достижения науки в приоритетных направлениях. Создана рабочая группа на базе Красноярского государственного педагогического университета им. В.П. Астафьева по разработке предложений совершенствования учебных планов и программ для студентов педагогических вузов. На общем собрании ассоциации в сентябре 2011 г. особое

внимание было уделено разработке совместных образовательных программ высшего профессионального образования. В декабре 2011 г. создана Международная ассоциация университетов «Совместные образовательные программы», учредителями которой стали Российский университет дружбы народов, Национальный исследовательский Томский государственный университет, Национальный исследовательский Новосибирский государственный университет, Новосибирский государственный технический университет. Необходимо также отметить, что Томский государственный университет совместно с МГУ им. Ломоносова, ЮУрГУ и ННГУ им. Н.И. Лобачевского является соучредителем Суперкомпьютерного консорциума российских университетов, в который входит ряд сибирских вузов.

Таким образом, в Сибири накоплен огромный потенциал сетевых научно-образовательных сообществ, совместная деятельность которых может стать, по словам президента Российского союза ректоров академика В.А. Садовничего, локомотивом системных инновационных изменений в научных, образовательных, технологических и гуманитарных процессах вузовской жизни, а их совместные образовательные и научно-производственные проекты могут стать эффективными механизмами социально-экономического развития региона. Следовательно, необходимо уделить серьезное внимание развитию научно-образовательных сетевых сообществ в СФО и поддержке их совместной проектной деятельности.

4. Развитие образовательной составляющей и создание системы производства и воспроизводства высококвалифицированных кадров нового поколения напрямую связано с организацией просветительской работы среди населения. Необходимо формировать положительное общественное мнение о суперкомпьютерных технологиях, обеспечивать популяризацию основных достижений в области СКТ, организовать профориентационную работу среди школьников и студентов.

Наиболее эффективным средством массовой информации является телевидение. В ассоциации «Сибирский открытый университет» накоплен богатый опыт в развитии познавательного телевидения как основы непрерывного образования детей и взрослых. На базе телепорта Томского государственного университета действует научно-образовательный телевизионный канал «ТВ-университет», в вузах ассоциации «Сибирский открытый университет» телевидение используется в реализации образовательных программ, имеются кафедры и факультеты журналистики, налажены связи с региональными телерадиокомпаниями, имеющими производственную базу для разработки и создания ТВ-программ. Познавательное телевидение является эффек-

тивным средством формирования общественного мнения и объединения интересов различных слоев общества, популяризации науки и привлечения в науку талантливой молодежи. Познавательные телевизионные программы являются важным элементом учебного процесса, необходимым условием повышения качества образовательных программ, они способствуют осуществлению просветительской миссии университетов, реализации государственных задач в воспитании молодежи.

Разработка и реализация системы информационного обеспечения общества о достижениях в области суперкомпьютерных технологий является одной из основных задач российского проекта «Суперкомпьютерное образование в России». На региональных телевизионных каналах в регионах России было представлено более 15 телевизионных программ о суперкомпьютерных технологиях, их применении в науке, производстве и бизнесе. Данный опыт показывает, что дальнейшее развитие познавательного телевидения на основе научно-образовательного канала «ТВ-университет» и электронных СМИ университетов может стать эффективным механизмом информационного обеспечения мероприятий Стратегии социально-экономического развития Сибири до 2020 г.

Таким образом, в регионе Сибирского федерального округа сегодня сосредоточены мощные высокопроизводительные ресурсы, научно-образовательный потенциал в области суперкомпьютерных технологий, которые могут служить основой дальнейшего развития приоритетного направления национальной экономики «Стратегические компьютерные технологии и программное обеспечение» в Сибири. И, следовательно, необходимым условием развития суперкомпьютерного образования и суперкомпьютерных технологий являются интеграция научно-образовательных учреждений, усиление их взаимодействия с реальным сектором экономики.

Литература

1. Суперкомпьютерное образование. http://i-russia.ru/computers/directions/459/
2. **Воеводин Вл.В., Гергель В.П., Соколинский Л.Б. и др.** Суперкомпьютерное образование в России. http://supercomputers.ru/
3. **Воеводин Вл.В., Гергель В.П., Соколинский Л.Б. и др.** Развитие системы суперкомпьютерного образования в России: текущие результаты и перспективы// Научный сервис в сети Ин-

тернет: экзафлопсное будущее: Труды Международной суперкомпьютерной конференции (19–24 сентября 2011 г., г. Новороссийск). – М.: Изд-во МГУ, 2011. – 643 с.
4. Суперкомпьютерные технологии в науке, образовании и промышленности / Под ред. В.А. Садовничего, Г.И. Савина, Вл.В. Воеводина. – М.:Изд-во МГУ, 2009. – 232 с.
5. Стратегия социально-экономического развития Сибири до 2020 года. http://www.sibfo.ru/strategia/strdoc.php

Разработка системы моделирования атмосферы с горизонтальным разрешением 10 км*

М.А. Толстых, А.В. Шляева, В.Г. Мизяк
Институт вычислительной математики РАН, Москва
Гидрометцентр России, Москва
МГТУ им. Баумана, Москва

Современная система моделирования атмосферы на временных масштабах от часа до нескольких месяцев состоит из прогностической модели атмосферы и системы усвоения данных наблюдений. Для реализации перспективной системы моделирования атмосферы на массивно-параллельных вычислительных системах решаются две задачи:

1. Разработка глобальной полулагранжевой модели атмосферы нового поколения. За основу принят динамический блок глобальной оперативной полулагранжевой модели атмосферы ПЛАВ [1].

2. Разработка схемы усвоения данных наблюдений для глобальной модели атмосферы на основе локального ансамблевого фильтра Калмана с преобразованием ансамбля (LETKF).

Глобальная полулагранжева модель атмосферы нового поколения

Основной моделью глобального среднесрочного прогноза погоды в России с конца 2009 г. является глобальная полулагранжева модель атмосферы ПЛАВ (полулагранжева, основанная на уравнении абсолютной завихренности) [1]. Модель разработана в Институте вычислительной математики РАН и Гидрометцентре России с помощью консорциума ALADIN/LACE, предоставившего параметризации процессов подсеточного масштаба. Внедрение этой модели позволило примерно в два раза сократить отставание России, по сравнению с лидирующей группой мировых прогностических центров в ошибках прогноза таких важных параметров, как давление на уровне моря, температура на уровне 850 гПа и высота поверхности 500 гПа. Краткое описание параллельной реализации модели приводится в [2].

По современным меркам разрешение оперативной версии модели является весьма грубым, поэтому коллективом разработчиков была реализована новая версия модели, имеющая горизонтальное разрешение порядка 20–25 км, а вертикальное – 51 уровень. Для повышения

* Работа выполнена при частичной поддержке гранта РФФИ 10-05-01066.

параллельной эффективности программного комплекса модели ПЛАВ было выполнено распараллеливание алгоритма постпроцессинга модели (расчета полей на стандартных изобарических поверхностях, которые и являются выходной прогностической продукцией модели). Были выполнены тестовые расчеты на вычислительной системе SGI Altix 4700, имеющей 1664 ядра процессоров Intel Itanium 9100, организованные в узлы по 128 ядер. Эта вычислительная система установлена в ГВЦ Росгидромета. Тестовые расчеты выполнялись для варианта модели с разрешением 0,225° по долготе, 0,17–0,24° по широте (примерно 20 км в средних широтах Северного полушария), 51 неравномерно расположенных вертикальных уровней в немонопольном режиме (запуск через систему очередей). Размерность сетки задачи – 1600x865x51. Значения параллельного ускорения в зависимости от числа используемых вычислительных ядер приведены на рис. 1.

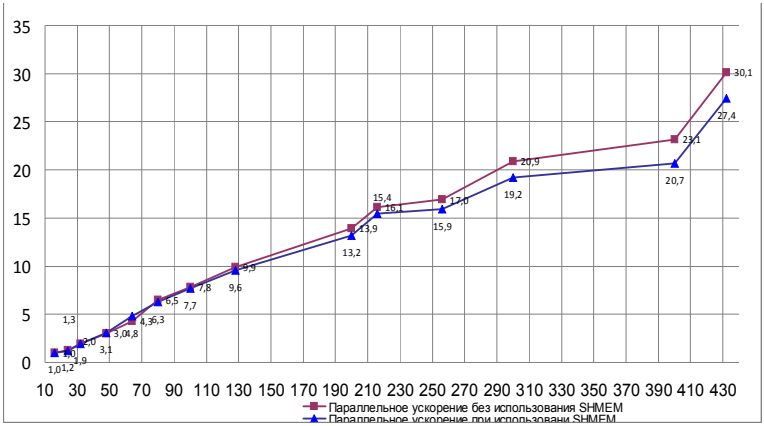


Рис. 1. Параллельное ускорение модели на вычислительной системе SGI Altix 4700 по сравнению со временем расчета на 16 процессорах как функция от количества процессоров

Мы видим, что модель позволяет эффективно использовать 432 процессора. Время расчета прогноза на сутки удовлетворяет требованиям оперативного счета. Однако дальнейшее увеличение числа процессоров не ведет к сокращению времени счета.

Часть вычислений блока решения уравнений динамики атмосферы выполняется в пространстве коэффициентов Фурье по долготе. В этом пространстве производится решение нескольких систем линейных алгебраических уравнений (СЛАУ) методом матричной прогонки. Этот метод требует наличия данных для всех широт на данном процессоре.

В то же время в основной части вычислений, осуществляемой в сеточном пространстве, данные распределены по широтам. Поэтому были выполнены модификации вычислительного алгоритма модели, которые позволили бы производить все вычисления при распределении данных по процессорам по широтам.

Для этого алгоритм векторной прогонки для решения блочно-трехдиагональных СЛАУ, получаемых в результате дискретизации уравнений диффузии вихря и дивергенции, полунеявной схемы интегрирования по времени и уравнения восстановления поля скорости по вихрю и дивергенции, был заменен на алгоритм дихотомии [3]. Двоичный логарифм параллельного ускорения этой части модели как функция двоичного логарифма числа процессоров приведен на рис. 2.

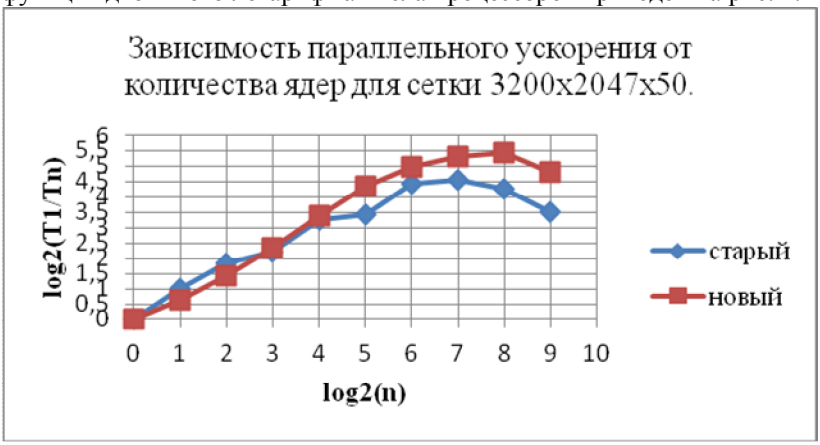


Рис. 2. Зависимость логарифма параллельного ускорения от логарифма количества ядер для сетки 3200x2047x50

Мы видим, что замена алгоритма привела к удвоению количества процессоров MPI, при котором наблюдается рост производительности.

Схема усвоения данных наблюдений для модели ПЛАВ

Система усвоения данных используется для вычисления начальных условий для старта прогноза (также называемых анализом) по краткосрочному прогнозу модели (первому приближению, размерности порядка $10^6 \times 10^8$) и данным наблюдений (размерности порядка 10^7), имеющим априори неизвестные ошибки.

В настоящее время широко распространены две группы методов усвоения данных: вариационное усвоение и усвоение с помощью ансамблевых фильтров Калмана.

Классический фильтр Калмана прогнозирует не только состояние системы, но и матрицу ковариаций. Однако его применение на практике для современных моделей атмосферы невозможно ввиду огромных размерностей вектора состояния. Идея ансамблевых фильтров Калмана [4] заключается в статистической оценке матрицы ковариаций по ансамблю первых приближений, размерность которого значительно меньше размерности модели (порядка $10^1 \times 10^2$):

Такой подход позволяет учесть реальные физические свойства потока, достаточно несложен в реализации, но предполагает дополнительные вычислительные затраты для расчёта большего числа прогнозов.

Нами разработана параллельная реализация схемы усвоения на базе локального ансамблевого фильтра Калмана с преобразованием ансамбля (Local Ensemble Transform Kalman Filter, LETKF) [5]. Одним из достоинств локализации является возможность эффективного распараллеливания алгоритма усвоения по данным, т.к. анализ можно вычислять параллельно в разных точках сетки. В нашей реализации данные (узлы регулярной широтно-долготной сетки) могут распределяться между вычислительными процессами равномерно по широте либо по косинусу широты.

Разработанная реализация была протестирована на задаче усвоения атмосферных данных для модели численного прогноза погоды ПЛАВ [1]. Для тестирования схемы усвоения данных использовалась версия модели ПЛАВ с горизонтальным разрешением 1,4°x1,125°, 28 уровней по вертикали. Радиус горизонтальной локализации – 1500 км, использовались наблюдения наземных станций, радиозондов и спутниковых наблюдений SATOB (общее число наблюдений – около 25000).

Были выполнены тестовые запуски прототипа LETKF с различным числом участников ансамбля (40 и 60) на вычислительной системе SGI Altix 4700.

Известно, что использование односторонних коммуникаций может ускорить работу параллельной программы. Были реализованы три версии коммуникаций: двухсторонние коммуникации MPI-I, односторонняя запись (MPI_PUT) и одностороннее чтение (MPI_GET) из стандарта MPI-II.

Графики параллельного ускорения расчетов при распределении обрабатываемых данных, равномерном по широте, представлены на рис. 3.

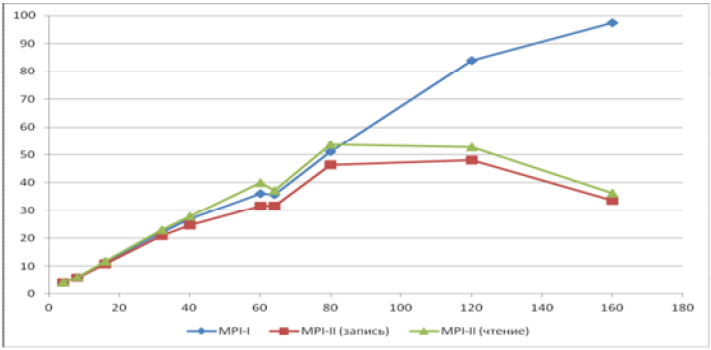


Рис. 3. Ускорение при использовании 60 участников ансамбля

Из графиков видно, что рост ускорения близок к линейному для 60 участников ансамбля. Для 40 участников ансамбля, начиная с числа процессов, равного 100, с ростом числа процессов ускорения не происходит. Это объясняется тем, что вычисления между процессами распределены неравномерно, так как вычислительная сложность задачи зависит от числа обрабатываемых локальных (расположенных на расстоянии не больше заданного) наблюдений, а наблюдения распределены неравномерно.

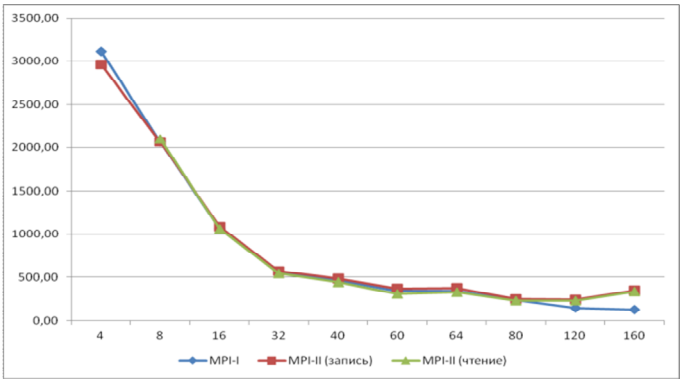


Рис. 4. Время выполнения на SGI Altix при использовании 40 участников ансамбля

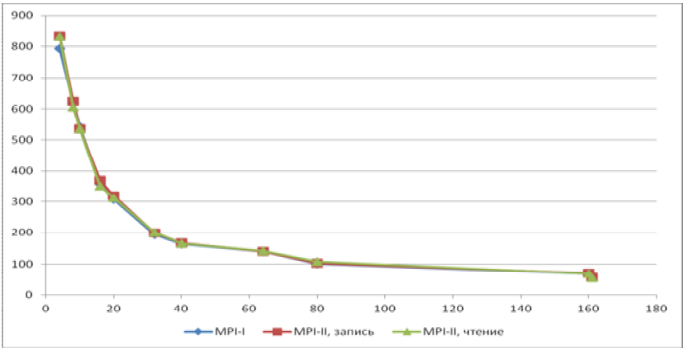


Рис. 5. Время выполнения на «РСК Торнадо» при использовании 40 участников ансамбля

Также были выполнены расчеты анализа для 40 участников ансамбля на пилотной кластерной системе РСК, установленной в октябре 2011 г. в Росгидромете. «РСК Торнадо» состоит из 96 вычислительных узлов, каждый из которых содержит по два шестиядерных процессора Intel Xeon 5680. Коммуникационная сеть построена на базе интерфейса Infiniband QDR. Время выполнения расчетов на SGI Altix и «РСК Торнадо» приведено на рис. 4 и 5 соответственно.

Как видно из графиков времени выполнения, использование односторонних коммуникаций не дало выигрыша во времени.

Выводы

Вычислительная сложность алгоритмов моделей атмосферы и усвоения данных наблюдений диктует необходимость их эффективной параллельной реализации.

Здесь представлены результаты работ по повышению параллельной эффективности программного комплекса модели атмосферы ПЛАВ. Показано, что применение метода дихотомии ведет к повышению в два раза числа процессоров, при котором наблюдается рост параллельного ускорения с ростом числа процессоров.

Нами разработан параллельный алгоритм локального ансамблевого фильтра Калмана для усвоения данных. Тестирование показало, что разработанная реализация позволяет достичь ускорения в 100 раз при распараллеливании на 160 процессах (размерность ансамбля равна 60). При меньшей размерности задачи (размерность ансамбля – 40 участников) эффективное распараллеливание ограничено 100 вычислительными процессами.

Литература

1. **Толстых М.А.** Глобальная полулагранжева модель численного прогноза погоды. – Москва; Обнинск: ОАО ФОП, 2011. – 111 с.
2. **Володин Е.М., Толстых М.А.** Параллельные вычисления в задачах моделирования климата и прогноза погоды // Вычислительные методы и программирование. – 2007. – Т. 8. – С. 113–122.
3. **Terekhov A.** Parallel Dichotomy Algorithm for solving tridiagonal system of linear equations with multiple right-hand sides // Parallel Computing. – 2010. – Vol. 36. – P. 423–438.
4. **Evensen G.** Sequential data assimilation with a nonlinear quasigeostrophic model using Monte-Carlo methods to forecast error statistic // J. Geophys. Res. – 1994. – Vol. 99. – P. 10143–10162.
5. **Hunt B.R., Kostelich E.J., Szunyogh I.** Efficient data assimilation for spatiotemporal chaos: a Local Ensemble Transform Kalman Filter // Physica D. – 2007. – Vol. 230. – P. 112–126.

Формосохраняющая интерполяция весовыми кубическими сплайнами

Б.И. Квасов
Институт вычислительных технологий СО РАН, Новосибирск

Даются новые алгоритмы построения весовых кубических сплайнов, весьма эффективные при интерполяции быстро меняющихся данных. Приведены оценки ошибок интерполяции весовыми сплайнами. Алгоритмы автоматического выбора параметров формы сплайна позволяют сохранять монотонность и выпуклость исходных данных.

Задача формосохраняющей интерполяции

Пусть имеются данные: (x_i, f_i), $i = 0,...,N+1,$ (1)

где $a = x_0 < x_1 < ... < x_{N+1} = b.$ Положим

$$f[x_i, x_{i+1}] = (f_{i+1} - f_i)/h_i, \quad h_i = x_{i+1} - x_i, \quad i = 0,...,N.$$

Данные (1) будем называть монотонно возрастающими, если

$$f[x_i, x_{i+1}] \geq 0, \quad i = 0,...,N,$$

и выпуклыми, если

$$f[x_i, x_{i+1}] \geq f[x_{i-1}, x_i], \quad i = 1,...,N.$$

Задача формосохраняющей интерполяции состоит в построении достаточно гладкой функции S такой, что $S(x_i) = f_i$, $i = 0,1,...,N+1,$ и S монотонна и выпукла на участках монотонности и выпуклости исходных данных.

Очевидно, что решение задачи формосохраняющей интерполяции неединственно. Будем искать его в виде весового кубического сплайна.

Определение 1. Интерполяционный кубический сплайн S со множеством весовых параметров $w_i > 0$, $i = 0,...N$ определяется как решение дифференциальной многоточечной краевой задачи (ДМКЗ)

$$\frac{d^4 S}{dx^4} = 0 \quad \text{для всех} \quad x \in (x_i, x_{i+1}),\ i = 0,...,N, \tag{2}$$

$$S \in C^1[a,b],$$

$$w_{i-1} S''(x_i^-) = w_i S''(x_i^+), \quad i = 1,...,N \tag{3}$$

с условиями интерполяции

$$S(x_i) = f_i, \quad i = 0,...,N+1. \tag{4}$$

Для однозначного определения сплайна требуется также задание краевых условий. Ограничимся использованием краевых условий двух типов:

1. Задание в граничных точках значений первой производной $S'(a) = f_0^{'}$ и $S'(b) = f_{N+1}^{'}$.
2. Задание в граничных точках значений второй производной: $S''(a) = f_0^{''}$ и $S''(b) = f_0^{''}$. (5)

Отметим, что значения производных должны быть согласованы с поведением данных. В противном случае можно получить несовместимость с условиями сохранения формы данных. Например, в случае краевых условий (5) можно использовать ограничения

$$f_0^{''} f[x_0, x_1, x_2] \geq 0, \qquad f_{N+1}^{''} f[x_{N-1}, x_N, x_{N+1}] \geq 0.$$

Определяющие соотношения для весовых кубических сплайнов

Рассмотрим алгоритм построения весовых сплайнов. Используем обозначения

$$M_i = w_{i-1} S''(x_i^-) = w_i S''(x_i^+), \quad i = 1, ..., N,$$
$$M_0 = w_0 S''(x_0^+), \qquad M_N = w_N S''(x_{N+1}^-).$$

Для $x \in [x_i, x_{i+1}]$ имеем

$$S''(x) = \frac{M_i}{w_i}(1-t) + \frac{M_{i+1}}{w_i} t, \text{ где } t = (x - x_i)/h_i, \quad h_i = x_{i+1} - x_i. \qquad (6)$$

Интегрируя (6) дважды и принимая во внимание краевые условия (4), получаем

$$S(x) = f_i(1-t) + f_{i+1} t - t(1-t)\frac{h_i^2}{6w_i}[(2-t)M_i + (1+t)M_{i+1}]. \qquad (7)$$

Дифференцируя эту формулу, имеем

$$S'(x) = f[x_i, x_{i+1}] - \frac{h_i}{6w_i}[(2 - 6t + 3t^2)M_i + (1 - 3t^2)M_{i+1}]. \qquad (8)$$

Так как $S'(x_i^-) = S'(x_i^+), \ i = 1, ..., N,$ то находим

$$\frac{h_{i-1}}{w_{i-1}} M_{i-1} + 2\left(\frac{h_{i-1}}{w_{i-1}} + \frac{h_i}{w_i}\right) M_i + \frac{h_i}{w_i} M_{i+1} = 6\delta_i f, \quad i = 1, ..., N, \qquad (9)$$

где $\delta_i f = f[x_i, x_{i+1}] - f[x_{i-1}, x_i]$.

Используя формулы (6) и (8), краевые условия можно переписать в виде

1. $2M_0 + M_1 = \dfrac{6w_0}{h_0}(f[x_0, x_1] - f_0^{'}),$

$M_N + 2M_{N+1} = \dfrac{6w_N}{h_N}(f_{N+1}^{'} - f[x_N, x_{N+1}]).$

2. $M_0 = w_0 f_0^{''}, \quad M_{N+1} = w_N f_{N+1}^{''}.$

Система (9) с краевыми условиями типов 1 и 2 имеет диагональное преобладание. Это обеспечивает существование и единственность весового кубического сплайна. Решение системы (9) может быть найдено методом обычной трехточечной прогонки.

В ряде случаев более удобным оказывается другое представление весового кубического сплайна, основанное на значениях его первой производной.

Пусть $m_i = S'(x_i), \ i = 0,..., N+1.$ На отрезке $[x_i, x_{i+1}]$ имеем

$$S(x) \equiv S_i(x) = f_i(1-t)^2(1+2t) + f_{i+1}t^2(3-2t) + + m_i h_i t(1-t)^2 - m_{i+1}h_i t^2(1-t),$$
$$x \in [x_i, x_{i+1}]. \qquad (10)$$

Дважды дифференцируя формулу (10), получаем

$$S_i^{''}(x) = \frac{2}{h_i}(3(1-2t)f[x_i, x_{i+1}] - (2-3t)m_i - (1-3t)m_{i+1}).$$

Используя это выражение на соседних подотрезках $[x_{i-1}, x_i]$ и $[x_i, x_{i+1}]$, в силу условия (3) находим

$$\lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = 3\lambda_i f[x_{i-1}, x_i] + 3\mu_i f[x_i, x_{i+1}], \ i = 1,..., N, \qquad (11)$$

где

$$\lambda_i = \frac{w_{i-1}h_i}{w_{i-1}h_i + w_i h_{i-1}}, \quad \mu_i = 1 - \lambda_i. \qquad (12)$$

Для замыкания системы (11) используем опять краевые условия:

1. $m_0 = f_0^{'}, \quad m_{N+1} = f_{N+1}^{'}.$

2. $2m_0 + m_1 = 3f[x_0, x_1] - \dfrac{h_0}{2}f_0^{''}, \quad m_N + 2m_{N+1} = 3f[x_N, x_{N+1}] - \dfrac{h_N}{2}f_{N+1}^{''}.$

Оценки ошибок приближения

Рассмотрим случай, когда исходные данные (1) получены из некоторой гладкой функции f, т.е. $f_i = f(x_i), \ i = 0,..., N+1.$ Используя стандартную методику из [2], приходим к следующим оценкам.

Теорема 1. Пусть весовой кубический сплайн $S \in C^1[a,b]$ с краевыми условиями $S'(x_0) = f_0'$ и $S'(x_{N+1}) = f_{N+1}'$ интерполирует значения $f_i = f(x_i),\ i = 0,...,N+1$ некоторой функции $f \in C^2[a,b]$. Тогда имеют место следующие оценки ошибок приближения:

$$\| S^{(r)}(x) - f^{(r)}(x) \|_C \le C_r \overline{h}^{2-r} \| f'' \|_C,\ r = 0,1, \qquad (13)$$

где $C_0 = 13/48,\ C_1 = 0,86229$ и $\overline{h} = \max_i h_i$.

Отметим, что оценки (13) имеют место и для случая краевых условий типа 2, а также при «естественных» краевых условиях, когда $S''(a) = S''(b) = 0$.

Теорема 2. Пусть весовой кубический сплайн $S \in C^1[a,b]$ с краевыми условиями $S'(x_0) = f_0'$ и $S'(x_{N+1}) = f_{N+1}'$ интерполирует значения $f_i = f(x_i),\ i = 0,...,N+1$ некоторой функции $f \in C^4[a,b]$. Тогда имеют место следующие оценки ошибок приближения:

$$\| S^{(r)}(x) - f^{(r)}(x) \|_C \le C_r \overline{h}^{2-r} \| f'' \|_C + \tilde{C}_r \overline{h}^{4-r} \| f^{IV} \|_C,\ r = 0,1, \qquad (14)$$

где $C_0 = 1/8,\ C_1 = 1/2,\ \tilde{C}_0 = 5/384$ и $\tilde{C}_1 = 1/24$.

Порядки аппроксимации в оценках ошибок приближения (14) могут быть улучшены путем специального выбора весов w_i. Пусть $w_i = (1 + \Delta_i^2)^{-1},\ \Delta_i = f[x_i, x_{i+1}]$. Тогда порядки аппроксимации в оценках (14) повышаются до $O(\overline{h}^{3-r}),\ r = 0,1$.

Адаптивный выбор параметров контроля формы

Нас интересует такой выбор весов (параметров формы) интерполяционного кубического сплайна S, который позволяет сохранять форму исходных данных $(x_i, f_i),\ i = 0,...,N+1$. Например, если функция f монотонна или выпукла на некотором отрезке $[x_j, x_k]$, то мы хотели бы, чтобы сплайн S также имел эти свойства.

Предположим для определенности, что $f[x_i, x_{i+1}] \ge 0,\ i = 0,...,N$, т.е. данные монотонно возрастают. Уравнения (11) для наклонов весового кубического сплайна по форме не отличаются от соответствующих уравнений для C^2 кубического сплайна. Это позволяет переписать достаточные условия монотонности для C^2 кубического сплайна (см. [1, с. 156]) для случая весового кубического сплайна. В частности, имеет место следующий результат.

Теорема 3. Пусть весовой кубический сплайн $S \in C^1[a,b]$ с краевыми условиями $S'(x_0) = f_0^{'}$ и $S'(x_{N+1}) = f_{N+1}^{'}$ интерполирует монотонные данные $\{f_i\}$, $i = 0,...,N+1$. Если выполняются неравенства

$$0 \le f_0^{'} \le 3f[x_0, x_1], \quad 0 \le f_{N+1}^{'} \le 3f[x_N, x_{N+1}],$$
$$\lambda_i f[x_{i-1}, x_i] \le (1+\lambda_i) f[x_i, x_{i+1}],$$
$$\mu_i f[x_i, x_{i+1}] \le (1+\mu_i) f[x_{i-1}, x_i],\ i = 1,...,N, \qquad (15)$$

то $S'(x) \ge 0$ для всех $x \in [a,b]$, т.е. сплайн S монотонен на $[a,b]$.

Принимая во внимание формулы (12), неравенства (15) можно переписать в виде

$$\frac{w_{i-1}}{w_i}\frac{h_i}{h_{i-1}} \ge \frac{f[x_i, x_{i+1}]}{f[x_{i-1}, x_i]} - 2,\ \frac{w_i}{w_{i-1}}\frac{h_{i-1}}{h_i} \ge \frac{f[x_{i-1}, x_i]}{f[x_i, x_{i+1}]} - 2,\ i = 1,...,N. \qquad (16)$$

Отсюда следует, что выбирая отношение w_{i-1}/w_i достаточно большим, можно существенно уменьшить ограничения на данные, достаточные для получения монотонности весового кубического сплайна. Для обычного кубического C^2 сплайна имеем $w_{i-1}/w_i = 1$.

Заметим, что для всякого i одно из неравенств (16) всегда выполняется. Пусть, например, имеет место неравенство $f[x_i, x_{i+1}] > f[x_{i-1}, x_i]$. Тогда второе неравенство в (16) выполнено. Первое неравенство может быть удовлетворено за счет выбора w_i непосредственно из неравенства. Можно предложить следующий алгоритм.

Пусть параметр w_{i-1} задан. Если неравенство (16) выполняется для $w_i = w_{i-1}$, то полагаем $w_i = w_{i-1}$. В противном случае находим w_i из того из неравенств (16), которое нарушается при $w_i = w_{i-1}$, заменяя в нем знак неравенства на знак равенства. Алгоритм начинаем с $w_0 = 1$ и легко находим все параметры $\{w_i\}$, обеспечивающие монотонность весового кубического сплайна для произвольных монотонных данных. Если на некотором шаге $w_i < \varepsilon$ ($w_i > 1/\varepsilon$), то полагаем $w_i = \varepsilon$ ($w_i = 1/\varepsilon$), где ε – достаточно малое положительное число, позволяющее предотвратить зануление (переполнение).

Данный алгоритм имеет ряд важных особенностей. На отрезках, где данные меняются незначительно, весовой сплайн будет иметь две непрерывные производные и наследует хорошие аппроксимативные свойства стандартного кубического сплайна. Вторая производная бу-

дет разрывна только в тех узлах сетки Δ, где требуется резкое изменение наклонов сплайна.

Заметим, что при задании «естественных» краевых условий, когда $S''(a) = S''(b) = 0,$ можно также воспользоваться неравенствами (16) как достаточными условиями монотонности (см. [1, с. 157]).

Рассмотрим теперь алгоритм выбора весовых параметров сплайна для сохранения выпуклости данных. Будем предполагать, что данные выпуклы, т.е. $f[x_{i-1}, x_i, x_{i+1}] \geq 0, \quad i = 1, ..., N.$

Уравнения (9) вместе с краевыми условиями для второй производной могут быть переписаны в виде

$$\begin{aligned} &M_0 = w_0 f_0'', \\ &\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_{i,} \quad i = 1, ..., N, \\ &M_{N+1} = w_N f_{N+1}'', \end{aligned} \tag{17}$$

где μ_i и λ_i определены в (12) и $d_i = \dfrac{6 w_{i-1} w_i}{w_i h_{i-1} + w_{i-1} h_i} \delta_i f.$ (18)

Уравнения (17) формально отличаются от соответствующих уравнений стандартного C^2 кубического сплайна только их правыми частями. Это позволяет записать достаточные условия выпуклости весового кубического сплайна (см. [1, с. 155]).

Теорема 4. Пусть весовой кубический сплайн $S \in C^1[a,b]$ с краевыми условиями $S''(x_0) = f_0''$ и $S''(x_{N+1}) = f_{N+1}''$ интерполирует выпуклые данные $\{f_i\}, \quad i = 0, ..., N+1.$ Если выполняются неравенства

$$f_0'' \geq 0, \quad f_{N+1}'' \geq 0, \quad 2d_i - \mu_i d_{i-1} - \lambda_i d_{i+1} \geq 0, \quad i = 1, ..., N, \tag{19}$$

где $d_0 = w_0 f_0''$ и $d_{N+1} = w_N f_{N+1}''$, то $S''(x) \geq 0$ для всех $x \in [a,b],$ т.е. сплайн S является выпуклым на $[a,b]$.

Предположим, что $\delta_i f > 0, \quad i = 1, ..., N.$ Неравенства (19) можно усилить, расщепив их на две части. Имеем

$$2d_i - \mu_i d_{i-1} - \lambda_i d_{i+1} = \mu_i (2d_i - d_{i-1}) + \lambda_i (2d_i - d_{i+1}), \quad i = 1, ..., N.$$

Следовательно, неравенства (19) будут выполнены, если выполняются условия

$$0 \leq w_0 f_0'' \leq 2d_1, \; 0 \leq w_N f_{N+1}'' \leq 2d_N, \; d_{i-1}/2 \leq d_i \leq 2d_{i-1}, \; i = 2, ..., N. \tag{20}$$

Принимая во внимание формулу (18), нетрудно показать, что неравенства (20) будут выполнены, если будут удовлетворены следующие ограничения на краевые условия:

$0 < f_0'' < 12\delta_1 f / h_0, \quad 0 < f_{N+1}'' < 12\delta_N f / h_N$

и на весовые параметры:

$$\frac{w_0}{w_1}\frac{h_1}{h_0} \le \frac{12\delta_1 f}{h_0 f_0^{"}} - 1,$$
$$\frac{1}{2}\frac{\delta_i f}{\delta_{i-1} f} - 1 \le \frac{w_{i-1}}{w_i}\frac{h_i}{h_{i-1}} \le 2\frac{\delta_i f}{\delta_{i-1} f} - 1, \quad i = 2, ..., N. \qquad (21)$$
$$\frac{w_N}{w_{N-1}}\frac{h_{N-1}}{h_N} \le \frac{12\delta_N f}{h_N f_{N+1}^{"}} - 1.$$

Используя эти неравенства, можно предложить алгоритм автоматического выбора параметров контроля формы, обеспечивающих сохранение выпуклости данных, который фактически является повторением описанного выше алгоритма для случая монотонных данных (исключение составляет случай отрицательных значений весовых параметров, которые мы заменяем на $\varepsilon > 0$). Легко видеть, однако, что если $\delta_{i-1} f \delta_i f < 0$, то неравенства (21) не могут быть удовлетворены ни для каких значений $w_i > 0$. Это означает, что следует рассматривать отдельно участки выпуклости и вогнутости данных.

Числовые примеры

Проиллюстрируем работу предложенных в статье алгоритмов монотонной и выпуклой интерполяции на популярных в литературе примерах. Для сравнения результатов использовалась также явная формула задания весов (параметров формы):

$$w_i = [1 + C_i (f[x_i, x_{i+1}])^2]^{-\alpha_i}, \quad C_i \ge 0, \quad \alpha_i \ge 0, \quad i = 0, ..., N.$$

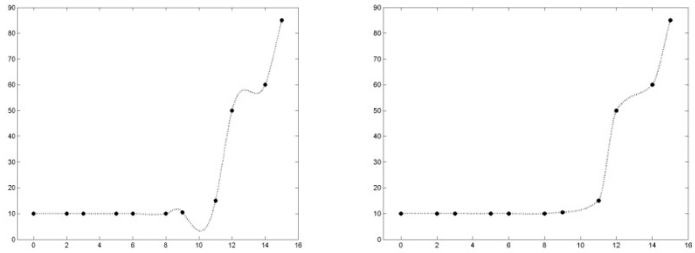
В качестве первого примера были взяты данные Акимы [1], приведенные в таблице, которые интерполировались весовым кубическим сплайном с «естественными» краевыми условиями $M_0 = M_{N+1} = 0$. На рис. 1, *а* показан график стандартного кубического сплайна класса C^2 ($w_i = 1$ для всех i). На рис. 1, *б* приведен график весового сплайна, когда веса выбирались по нашему алгоритму монотонной интерполяции.

x_i	0	2	3	5	6	8	9	11	12	14	15
f_i	10	10	10	10	10	10	10.5	15	50	60	85

В качестве второго примера была рассмотрена интерполяция функции $f(x) = 2 - \sqrt{x(2-x)},\ 0 \le x \le 2,$ которая дает полуокружность. Эта функция интерполировалась на сетке, равномерной по x (рис. 2, *а*) и

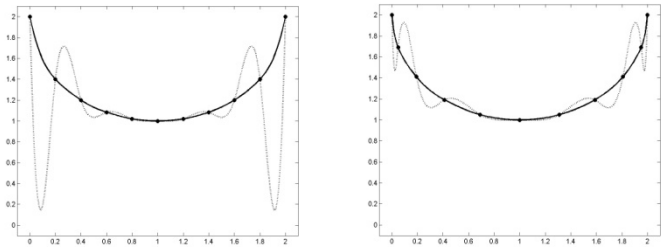
по длине хорд (рис. 2, *б*). В обоих случаях использовалось 11 точек интерполяции и краевые условия $S'(x_0) = -50,\ S'(x_{10}) = 50$ (C^2 кубический сплайн) и $m_0 = 2f[x_0, x_1],\ m_{10} = 2f[x_9, x_{10}]$ (весовой C^1 сплайн). Штриховой и сплошной линиями показаны графики интерполяционных кубического C^2 сплайна и весового кубического сплайна с весами, выбранными по нашему алгоритму. Переход к сетке с постоянным шагом по длине хорд позволяет уменьшить осцилляции C^2 сплайна, но не устраняет их. Весовой C^1 сплайн сохраняет свойство монотонности исходных данных.

Алгоритм автоматического выбора весовых параметров для сохранения выпуклости был протестирован также на этом примере. Он дает те же самые результаты, что и показанные на рис. 2. Использовались краевые условия $M_0 = 3w_0\delta_1 f / h_0$ и $M_{N+1} = 3w_N\delta_N f / h_N$.



а) б)

Рис. 1. Данные Акимы (см. таблицу) с краевыми условиями $M_0 = M_{N+1} = 0$: *а* – интерполяционный C^2 сплайн; *б* – весовой сплайн с параметрами формы по нашему алгоритму



а) б)

Рис. 2. Интерполяция полуокружности по данным на сетке: *а* – равномерной по x координате и *б* – равномерной по длине хорд. Штриховая и сплошная линии дают графики C^2 кубического ($w_i = 1$) и весового C^1 (наш алгоритм) сплайнов

Литература

1. **Квасов Б.И.** Методы изогеометрической аппроксимации сплайнами. – М.: Физматлит, 2006.
2. **Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л.** Методы сплайн-функций. – М.: Наука, 1980.

Современная система параллельного программирования. Лаконизм. Конструктивность. Расширяемость

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

Статья посвящена проблеме разработки систем параллельного программирования и подходов к реализации языков параллельного программирования. В центре внимания вопросы выбора лаконичных форм представления программ, обеспечения конструктивных построений, гарантирующих сохранение свойств программ при их реорганизации, и поддержки расширяемости языка программирования по мере развития средств и методов параллельного программирования.

Введение

«Ничем не скроешь фундаментальную трудность параллелизма» – так прозвучало заявление Мартина Одерски (Martin Odersky) в его приглашенном докладе на 20-й Международной конференции «Конструирование компиляторов», состоявшейся в марте 2011 г. в Германии под председательством Дженса Кнопа (Jens Knoop) [1]. Несмотря на многочисленные призывы и конкурсы пока не появилось общих идей по новому поколению языков и систем программирования (ЯСП), в которых бы параллелизм имел самостоятельное значение, а не рассматривался бы (по понятным причинам) как пристройка к традиционному программированию.

Современное развитие ЯСП фактически ориентировано на решение задач параллельного программирования. Как правило, новые ЯСП включают в себя библиотечные модули, обеспечивающие организацию процессов, или подъязыки, допускающие многопоточное программирование [2]. Это, увы, не исключает реальную практику ручного распараллеливания ранее отлаженных обычных программ, приведения их к виду, удобному для применения производственных систем поддержки параллельных вычислений. Значительная часть таких работ носит технический характер и заключается в систематической реорганизации структур данных, изменении статуса переменных и включении в программу аннотаций, сообщающих компилятору об информационно-логических взаимосвязях. Существенным ограничением результата ручного распараллеливания является не только опасность повторной

отладки алгоритма, но и его жесткая зависимость от характеристик целевой архитектуры.

Разнообразие моделей параллельных вычислений и расширение спектра доступных архитектур следует рассматривать как вызов разработчикам ЯСП, как проблему создания методов компиляции многопоточных программ для многопроцессорных конфигураций. Язык должен допускать представление любых моделей параллелизма, проявляемого на уровне решаемой задачи или реализуемого с помощью реальной архитектуры. Причем такое представление требует лаконичных форм и конструктивных построений, гарантирующих нужные свойства программ при их создании и реорганизации. Не менее важна расширяемость языка по мере развития средств и методов параллельных вычислений.

Рассматривая задачу формализации языков параллельного программирования как путь к решению проблемы адаптации программ к различным особенностям используемых многопроцессорных комплексов и многоядерных процессоров, приходится видеть, что решение этой проблемы требует разработки новых методов компиляции программ и развития средств и методов описания семантики языков программирования.

Лаконизм

История языков программирования накопила целый ряд примеров лаконичных форм представления программ, начиная с умолчаний и неявных циклов в языке Fortran. С точностью до реализационной семантики при разработке языков параллельного программирования можно унаследовать языковые конструкции и механизмы из привычных парадигм программирования и зарекомендовавших себя языков параллельного программирования.

Прежде всего, это алгебраические механизмы просачивания функций и операций относительно структур данных, предложенные в первом языке параллельного программирования APL. Дальнейшее упрощение изобразительных средств управления параллелизмом дает предложенный в языке Sisal подход к неявному распараллеливанию циклов на основе выстраивания пространства итераций по пространству обрабатываемых данных.

Функциональное программирование вносит свой вклад техникой рекурсивных определений и отображений, параллелизмом обработки аргументов функций, ленивых вычислений, а также сведением понятия «условия» в ветвлениях к понятиям «страж» или «образец». Объектно-

ориентированное программирование активизировало перегрузку операций и функций. Более специфичные и эффективные формы выражений возникают при обработке составных значений с помощью фильтров и проекций в новых мультипарадигматических языках. Как правило, лаконизм достигается расширением используемых понятий или варьированием их реализации.

Конструктивность

Результаты анализа сложившейся практики системной поддержки конструктивных построений, гарантирующих правильность программ при их реорганизации, показывают, что, как правило, все сводится к комбинаторике многократно используемых компонент, таких как структуры данных, функции, модули, классы объектов. Снижение сложности отладки программ достигается факторизацией программ исходя из определения структур данных и проявления подобия программ обработки данных форматам структур данных. Языки и системы функционального программирования допускают факторизацию более общего вида. Они позволяют выделять такие компоненты, как схемы управления, что дает возможность типизации управления, действий, моделей вычисления, реорганизации памяти и т.д. Эффективность параллельного программирования потребует факторизации относительно схем управления и дисциплины доступа к памяти.

Появление нового поколения языков программирования, таких как C# и F#, показывает общую тенденцию включения средств обработки кода программ, что позволяет при реализации параллельных алгоритмов решать задачи верификации и оптимизации программ, включая их распараллеливание. Следует отметить, что кроме собственно правильности, понимаемой как соответствие аргументов результатам, параллельные программы должны отвечать достаточно сложным критериям, таким как корректность синхронизации процессов, надежность, живучесть, справедливость и др., не отслеживаемые обычной схемой компиляции программ.

Тем не менее в докладах на конференциях по методам компиляции пока представляют отдельные, ранее сложившиеся, средства реализации ЯСП, лишь адаптированные к переносу в мир параллельного программирования. В их числе – «компиляция на лету», выделение чисто функциональных подмножеств и форм с однократным присваиванием, обратимая компиляция и интерпретация, транзакционная память, анализ достижимости действий, преобразования циклов над большими массивами и т.п. [1]. Конструктивные методы параллельного програм-

мирования желательно нацеливать на обеспечение нужных свойств по построению.

Расширяемость

Следует отметить проблему расширяемости ЯСП по мере развития средств и методов параллельных вычислений, обусловленную высоким темпом прогресса в области элементной базы и информационных технологий в целом. Многие понятия языка программирования – схемы управления программой, образующие программу действия, вычисления и данные – при переходе к параллельному программированию претерпевают изменения, и возникает необходимость в дополнительных понятиях.

Основные методы представления вычислений связаны с использованием неявных циклов, позволяющих избежать выписывания однотипных схем над стандартными структурами данных типа многомерных векторов. Так, например, результат операции над скалярами в языках параллельного программирования распространен на произвольные однородные структуры данных. Список операций, допускающих распространение, определен реализацией. Обычно это арифметические операции (+, −, ×, /). Этот метод может использоваться более широко распространением на технику применения функций, что повышает лаконизм выражений.

Из бинарных операций можно конструировать фильтры. Результат фильтрации исчезает из аргумента – он переносится в другую структуру данных или сохраняется как значение. Структура из фильтров дает структуру из результатов их применения к одному и тому же аргументу. Результативность вычислений можно повышать с помощью специально устроенных данных – так называемых «рецептов». Это отложенное исполнение фрагмента программы, его замыкание, т.е. пара из действия и контекста. Общие решения по обеспечению лаконизма, конструктивности и расширяемости приводят к трансформационной семантике языка программирования.

Трансформации

Трансформационная семантика обеспечивает сведение конструкций языка программирования к его базовым средствам, что позволяет упростить операционную семантику, а также выбрать реализационное ядро системы программирования при его экспериментальной раскрутке. Похожая техника применяется при сведении грамматик языка к форме, удобной для автоматизации построения анализатора, и при оптимизации программ.

Направление преобразований программ обычно связано с определенными критериями применимости и оптимальности, учитывающими результаты анализа логических и информационных связей. При организации параллельных процессов такие критерии обладают спецификой, отражающей особенности эксплуатации многоядерных архитектур. В частности, возрастает роль учета времени доступа к разнородной памяти и статического планирования загрузки процессоров наряду с обеспечением обратимости обработки данных и динамического управления производительностью вычислений. Поддержка такой семантики вычислений выходит за границы традиционных решений по реализации языков высокого уровня.

Задача трансформационной семантики – сведение программы к нормализованной форме, удобной для интерпретации программ или генерации исполнимого кода. В случае многопоточных программ преобразование сети потоков нацелено на сведение к однородной системе потоков, однозначно отображаемых на заданный комплекс процессоров – размещение потоков по процессам или назначение процессоров для выполнения потоков. Такое требование можно выразить формулой

[(b0: ; b1: ; … bK:), p1!(b0: d0i; …), … pN!(b0: d0j; ...)],

где **i** и **j** обозначают принадлежность потоку,

, – параллельное или одновременное исполнение,

; – последовательное исполнение,

! – назначение процессора,

b0:, b1:, … – барьеры,

p1, pN, … – процессоры,

d0i, d0j, … – действия,

(b0:; b1:; … bK:) – шкала событий/барьеров,

pM!(b0: d0i; …) – программа (последовательность) действий для процессора **pM**.

В такой как бы «причесанной» форме все потоки начинаются с барьеров и общая шкала событий упорядочена так, что последовательность событий потока ей не противоречит. Можно считать, что процессоры включаются сами. Шкала событий содержит списки ожидающих потоков. Действия, выполняемые процессорами, соотнесены с их исходными потоками.

Достаточно простые преобразования сети потоков позволяют варьировать схемы потоков и многие конструкции языка программирования сводить к взаимодействию простых потоков:

(A;B) ↔ (a:A; b:B) # – расстановка – стирание барьеров.

(a:A; b:B) ↔ [[a:A, b:B], (a: ; b:)] – вынесение последовательного управления в отдельный поток – восстановление последовательности действий.

(a: A; b: B) ↔ [(a: A; b:), (b: B)] – разрез последовательности – перенос «хвоста», если нет локальной информационной зависимости между **A** и **B**.

(a: A; b: B) ↔ [a: A, b: B] – разбиение последовательности на потоки – сплющивание линии в слой, если **A** и **B** информационно не связаны.

[a: A, b: B] ↔ { (a: A; b: B) | (b: B; a: A)} – слияние потоков в одну последовательность – вытягивание слоя в линию. Выбор варианта требует учета последовательности барьеров в других потоках и общей шкале событий. Критерий оптимальности – объем выполнимых вычислений.

((a: A; b: B) ; c: C) ↔ (a: A; (b: B ; c: C)) ↔ (a: A; b: B ; c: C).

[[a: A, b: B] , c: C] ↔ [a: A, [b: B , c: C]] ↔ [a: A, b: B, c: C].

([A,B];C) ↔ [(A;C), (B;C)] – исключение слияний – слияние совпадающих продолжений.

[(A;B),C]↔ ([A,C]; [B,C]) – распределение параллельного потока – слияние совпадающих потоков.

[[a: A], [a: B]]↔ [a: [A, B]] – варьирование числа одновременных потоков.

Обратимость преобразований и чувствительность их результата к информационным связям между фрагментами программы требуют формализации критериев применимости трансформаций и выбора подходящего варианта. Можно констатировать, что выяснение информационной связанности действий **B** и **A** сводится к проверке существования контекста, в котором различны результаты программы **C** при изменении порядка вычислений **B** и **A**.

[(A; B), C] ≠ [(B; A), C].

Из числа базовых средств можно вывести ветвления, циклы и вызовы функций, реализуя их средствами синхронизации потоков:

(if A then B) ↔ [(A;B:);(B:B)] – сведение обхода к синхронизации и обратно.

При отладке формируется ряд контекстов, на которых демонстрируются конкретные свойства фрагментов, из которых собирается полная программа. Это контексты для отдельных потоков, для пар синхронизованных потоков, для интегрированной из потоков программы, а кроме того, контексты для удостоверения наличия-отсутствия информационных связей между фрагментами. Возможны пользовательские преобразования схем управления процессами, что позволит не только минимизировать «ручную» аранжировку распараллеливаемых программ, но и даст основу для формирования библиотек преобразования схем программ.

Абстрактная машина

Обычно операционная семантика языка программирования базируется на определении абстрактной машины, которая в случае многопроцессорных конфигураций становится сложной конструкцией из абстрактных процессоров. Выделение в схеме компиляции параллельных программ уровня трансформационной семантики позволяет пересмотреть формат абстрактной машины. Вместо базового исполнителя команд, определяемых как переходы от одного состояния к другому, становится естественным использовать абстрактный макроассемблер, допускающий настройку на конкретную конфигурацию при исполнении программы. (Похожий механизм был реализован в форме символьного макроассемблера в качестве выходного языка будущей системы БЕТА, получивший название языка СИГМА [3]).

При подготовке примеров для демонстрации учебного языка параллельного программирования были приняты следующее ограничения:

− взаимодействия потоков выполняются только через синхронизацию;
− одновременно исполняемые фрагменты могут обмениваться данными через общую память;
− при взаимоисключении каждый вариант работает в своей копии контекста;

– поддерживается список восстановления многократно выполняемых фрагментов.

Заключение

Данной статье предшествует работа по определению языка начального обучения параллельному программированию Кубик, в которой представлена базовая модель организации взаимодействующих процессов, позволяющая показать природу трудностей параллельного программирования и типичные пути решения известных проблем. Программа факультативного курса для школьников по параллельному программированию на базе этого языка изложена в материалах секции «Информатика образования» Ершовской конференции по информатике в июне 2011 г. [4].

Следующий шаг – разработка языка высокого уровня параллельного программирования, приспособленного к ознакомлению студентов с более сложными моделями вычислений и с методами верификации программ, без которых надежность параллельного программирования весьма проблематична. Для нужд этого шага потребуется более строгая формализация семантики в трансформационно-операционном стиле. Представленный в статье подход к определению языка параллельного программирования предназначен для поддержки экспериментов по разработке новых языков, ориентированных на учебно-исследовательские проекты в области создания распределенных информационных систем.

Литература

1. LNCS 6601. Jens Knoop Compiler Construction. 20th International Conference, CC 2011. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrьcken, Germany, March 26 – April 3, 2011. – Berlin: Springer, 2011. – 330 p.
2. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.
3. **Степанов Г.Г**. [http://www.iis.nsk.su/files/artcles/mozaika_stepanov_sigma.pdf].
4. **Городняя Л.В.** О курсе «Начала параллелизма» // Ершовская конференция по информатике. Сек. Информатика образования. 27 июня – 1 июля 2011 г. – Новосибирск, 2011. – С. 51–54.

Универсальные параллельные алгоритмы с многоточечной высокоточной аппроксимацией граничных условий*

В.И. Паасонен
Институт вычислительных технологий СО РАН, Новосибирск

Описывается универсальная технология расчета краевых задач в кусочно-однородных областях декартовых или произвольных ортогональных криволинейных систем координат с расположением границ подобластей на координатных линиях. Метод базируется на применении компактных схем внутри однородных подобластей и односторонних многоточечных аппроксимаций потоков в граничных условиях.

Введение

Для численного решения краевых задач в неоднородных областях, составленных из однородных включений, одинаково ориентированных относительно системы координат, предлагается класс разностных методов, допускающих параллельную реализацию [1, 2]. Метод применим для любой криволинейной ортогональной системы координат, а также распространяется на задачи с декомпозицией сложных областей на подобласти [3] с постановкой в качестве «мягких» граничных условий на разрезах условий равенства левых и правых односторонних производных. Предлагаемый метод сформулирован также для решения основанных на разностном подходе задач интерполяции сплайнами [4], когда множество узлов интерполяции составляют подмножество более детальной сеточной области, а условия гладкого сопряжения элементов интерполирующей функции записываются как равенство многоточечных разностных аналогов производных слева и справа от узла интерполяции.

Многие дифференциальные уравнения в частных производных второго порядка в прямоугольных и криволинейных ортогональных координатных системах могут быть аппроксимированы с четвертым, а иногда и с шестым порядком относительно шагов пространственной сетки на компактном множестве узлов, не выступающем за пределы 3×3×…×3-точечного шаблона. При этом возникает проблема аппроксимации с адекватной точностью также и краевых условий. Это можно

* Работа поддержана грантом РФФИ № 11-01-00294-а и междисциплинарным интеграционным грантом № 103.

сделать аппроксимируя закон сохранения в балансной ячейке на основе интегральной постановки либо аппроксимируя разностное граничное условие «как есть» по аналогии с дифференциальной постановкой.

Для достижения универсальности и хорошей структурированности алгоритма наиболее удобно использовать односторонние аналоги первых производных в граничных условиях, взяв достаточное число узлов сетки в односторонних аппроксимациях потока. Этот способ одинаково подходит и для условий Неймана, и для условий третьего рода, и для условий равенства потоков на границах раздела различных сред. Такой симбиоз компактных схем и многоточечных аппроксимаций граничных условий, допускающий как обычную реализацию, так и применение параллельных технологий, впервые предложен в [1].

Суть метода. Предположим, что решается система разностных уравнений вида

$$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = F_i,\ 0 < i < N,\ i \neq i_k,\ k = 1,...,r-1$$

с внешними граничными условиями при $i = i_0 = 0$ и $i = i_r = N$

$$\mu_0 u_0 - \frac{\nu_0}{h_1}\sum_0^s \alpha_j u_j = \phi_0, \quad \mu_r u_N - \frac{\nu_r}{h_r}\sum_0^s \alpha_j u_{N-j} = \phi_r$$

и с внутренними граничными условиями

$$\frac{\lambda_k}{h_k}\sum_0^s (-\alpha_j) u_{i-j} = \frac{\lambda_{k+1}}{h_{k+1}}\sum_0^s \alpha_j u_{i+j}, \quad i = i_k, \quad k = 1,...,r-1.$$

Эта система представляет собой разностную аппроксимацию одномерной краевой задачи в слоистом пакете или одномерный фрагмент многомерной системы, полученный в результате ее расщепления на одномерные задачи. Будем предполагать, что коэффициенты многоточечных разностных аналогов потоков выбраны, исходя из максимально возможного порядка аппроксимации граничных условий.

Описываемый ниже алгоритм является обобщением метода распараллеливания прогонки [2] и отличается от него неоднородностью области и наличием «длинных» граничных условий s-го порядка точности. Обозначим пока неизвестные значения решения в узлах разбиения и на внешних границах через $w_k, (k = 0,...,r)$ и введем в каждом слое $\omega_k, = \{x_{k-1} \le x \le x_k\}$ локальную нумерацию узлов $m = 0,...,M,\ M = m_k$. Вектор решения y^k в слое ω_k будем отыскивать в виде специальной линейной комбинации

$$y^k = p^k w_{k-1} + q^k w_k + z^k$$

решений трех задач, где z^k – решение неоднородного уравнения с нулевыми граничными условиями

$$a_{i_k+m} z^k_{m-1} + b_{i_k+m} z^k_m + c_{i_k+m} z^k_{m+1} = F_{i_k+m},$$

а p^k, q^k – два независимых решения соответствующего однородного уравнения с различным образом нормированными (единица слева, а нуль справа, и наоборот) граничными условиями. Процесс их решения составляет содержание первого этапа алгоритма, причем его можно реализовать независимо или параллельно для каждого из r слоев пакета. Легко заметить, что наша линейная комбинация y^k трех вспомогательных решений удовлетворяет исходному неоднородному уравнению с «правильными» граничными условиями $y^k_0 = w_{k-1}$, $y^k_M = w_k$, в чем можно убедиться непосредственной подстановкой. Уравнения для вычисления значений решения w_k получаем, используя граничные соотношения на внешних и внутренних границах. Так, подставляя выражение решения y^1, соответствующее первому слою, в соотношение на левой границе, получим

$$\mu_0 w_0 - \frac{\nu_0}{h_1} \sum_0^s \alpha_j \, (p^1_j \, w_0 + q^1_j \, w_1 + z^1_j) = \phi_0.$$

Очевидно, это равенство представляет собой линейное уравнение для двух неизвестных значений w_0 и w_1 решения на границах первого слоя, коэффициенты которого легко устанавливаются приведением в нем подобных слагаемых. Из правого граничного условия аналогично устанавливается связь для двух неизвестных значений w_{r-1} и w_r на границах последнего слоя, а из соотношения на k-й границе раздела сред следует трехточечная связь для значений искомого решения на трех последовательно расположенных границах раздела сред.

Таким образом, на втором этапе вектор значений решения на границах слоев может быть определен из системы уравнений с трехдиагональной матрицей. Размерность этой системы, очевидно, постоянна и равна общему числу границ $r+1$ в слоистом пакете. Эта задача существенно отличается от задач первого этапа, так как ее размерность не зависит от степени детальности сетки. Тем не менее ее необходимо исследовать на предмет устойчивости, в чем нас убеждают известные примеры плохо обусловленных систем малой размерности. Такое доказательство проведено в [1] в случае кусочно-постоянных теплофизических характеристик. Показано, что при условии диагонального преобладания в трехточечной задаче Дирихле для каждого отдельного слоя трехточечная система, связывающая значения решения на границах слоев, также имеет диагональное преобладание, а этого достаточно для устойчивости.

Заметим, что в исходной разностной постановке устойчивость не является очевидной, так как размерность задачи растет неограниченно при стремлении шага сетки к нулю, а матрица не относится к хорошо исследованному классу. В изложенном же параллельном методе из общей задачи выделены потенциально бесконечномерные части в виде ряда независимых классических задач Дирихле с хорошими свойствами, в то время как вектор значений решения на границах раздела сред определяется устойчивым образом из линейной системы постоянной размерности, не зависящей от шага сетки.

Многомерные задачи. При решении многомерных задач рассматриваемого класса изложенный алгоритм представляет практический интерес, поскольку нет смысла решать параллельно каждую одномерную систему. Распараллеливать многомерную задачу целесообразно по подобластям той размерности, какую имеет исходная задача. Например, в плоском случае архитектура такого разбиения может представлять собой набор полос или клеток. При этом вдоль полос на плоскости (или брусков в пространстве), составленных из смежных подобластей, возникают задачи рассмотренного выше типа.

Для упрощения логики решения представляется целесообразным продолжить все участки границ подобластей до пересечений с внешними габаритами, преобразуя довольно произвольную область в область клетчатой структуры. В таком случае смежные клетки могут оказаться занятыми как различными, так и одинаковыми материалами. Тем не менее фиктивную границу между двумя подобластями с одинаковыми материалами удобнее воспринимать как полноценную границу раздела сред и не записывать в ней разностную схему, а формулировать приближенно равенство потоков, как описано выше. При одинаковых теплофизических характеристиках это условие по существу будет означать условие гладкости решения – равенство разностных аппроксимаций левой и правой производной, которое можно рассматривать как «мягкое» внутреннее граничное условие. Это обстоятельство позволяет расширить границы применимости описанного метода, включив в них задачи, решаемые с помощью декомпозиции области на подобласти [3]. Возможен также и смешанный тип задач – с неоднородностью и с декомпозицией. В этом случае область имеет и фиктивные и реальные границы, однако условия на них ставятся единообразно в форме приближенного равенства потоков через границу.

Алгоритм решения многомерной задачи на одном временном шаге осуществляется в четыре этапа. Первые два – это прогонки в двух направлениях по всем линиям сетки, кроме границ клеток. На третьем этапе из высокоточных одномерных граничных условий явно вычис-

ляется решение на сторонах клеток, исключая углы. На заключительном этапе вычисляется решение в углах всех клеток явно по специальным формулам замыкания [5, 6], которые записываются на шаблоне типа «большой крест» с s узлами в каждом направлении от центра креста. В настоящее время формулы замыкания построены в общем виде для любого порядка точности и для любой ортогональной криволинейной системы координат.

Заключительные замечания. Конкретизируем вид односторонних многоточечных разностных аппроксимаций первых производных, которые используются при записи разностных граничных условий. Аппроксимируем первую производную на равномерной сетке с шагом h, применяя одностороннюю разделенную разность с произвольным числом узлов

$$\frac{\Delta_s}{h}u(x)=\frac{1}{h}\sum_0^s \alpha_k u(x+kh).$$

Если коэффициенты α_k удовлетворяют линейной алгебраической системе уравнений

$$\sum_0^s \alpha_k\, k^l=\delta_{1l},\quad l=0,\dots,s,$$

(δ – символ Кронекера), то погрешность аппроксимации первой производной на достаточно гладких функциях составляет величину $O(h^s)$, причем ввиду линейной независимости строк матрицы системы это есть максимально возможный порядок аппроксимации. Коэффициенты многоточечной разности «вперед» вычисляются по правилу Крамера в явном виде

$$\alpha_k=\frac{(-1)^{k+1}}{k}C_s^k,\ k\neq 0,\qquad \alpha_0=-\sum_1^s \alpha_k,$$

где C_s^k – число сочетаний из s по k. Аналогично определяется разность «назад» порядка s.

Для практических целей особый интерес представляет случай $s=4$, когда внутри однородных подобластей применяются компактные схемы и ожидается четвертый порядок точности в целом. Случай $s=2$ также интересен как основа для применения любых схем второго порядка точности как альтернативы структурно более сложным балансным схемам сквозного счета.

Важнейшими положительными сторонами предлагаемой методики являются применимость параллельных технологий, универсальность и хорошая структурированность алгоритма на основе единого подхода

для различных типов граничных условий, для любых порядков аппроксимации граничных условий и для широкого класса задач.

Литература

1. **Паасонен В.И.** Сходимость параллельного алгоритма для компактных схем в неоднородных областях // Вычислительные технологии. – 2005. – Т. 10, № 3. – С. 81–89.
2. **Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В.** Об организации параллельных вычислений и распараллеливании прогонки // Численные методы механики сплошной среды. 1978. – Т. 9, № 7. – С. 136–139.
3. **Паасонен В.И.** Параллельные алгоритмы на основе мягких внутренних граничных условий // Вычислительные технологии. 2006. – Т. 11, ч. 2. – Специальный выпуск. – С. 21–27.
4. **Паасонен В.И.** Высокоточные методы построения гиперболических сплайнов // Вычислительные технологии. – 2007. – Т. 12, № 2. – С. 115–121.
5. **Паасонен В.И.** Формулы замыкания для компактных схем в неоднородных областях // Вычислительные технологии. – 2009. – Т. 14, № 4. – С. 93–99.
6. **Ичетовкин Д.А., Паасонен В.И.** Численное исследование высокоточных схем в областях клетчатой структуры // Вычислительные технологии. – 2010. – Т. 15, № 6. – С. 81–87.

Преимущества параллельных вычислений при моделировании основного состояния наноразмерных ферримагнетиков*

В.А. Родионов
Томский государственный университет

Проведён анализ эффективности двух модификаций алгоритма Метрополиса с применением распараллеливания для моделирования спиновой конфигурации наноразмерных ферримагнитных частиц различного диаметра. Показано, что максимальный выигрыш в скорости расчётов даёт распараллеливание вычисления энергии.

За прошедшее десятилетие был достигнут существенный прогресс в производительности вычислительной техники, а также и некоторый предел в скорости вычислений, которую можно получить при помощи одного ядра, поэтому в настоящее время основным направлением развития является увеличение числа ядер процессоров. Возросшее число ядер позволяет выполнять несколько наборов операций одновременно, что значительно ускоряет вычислительные процессы, однако требует от разработчиков программного обеспечения разбиения задачи на допускающие независимое выполнение подзадачи и контроля синхронизации потоков внутри процесса вычисления [1]. Некоторые вычислительные алгоритмы допускают возможность распараллеливания разными способами, при этом результирующее время и ресурсы, затраченные на вычисление, будут также отличаться. Таким образом, в настоящее время для обеспечения максимального быстродействия при расчетах необходимо использовать алгоритмы, которые могут быть разбиты на подзадачи оптимальным способом. Важно добавить, что невозможно точно предсказать, насколько будет эффективно то или иное разбиение алгоритма на части для данной конфигурации оборудования.

В данной работе было произведено исследование быстродействия алгоритма Метрополиса для расчета энергии сферической наноразмерной ферримагнитной частицы со структурой шпинели, элементарная ячейка которой представлена на рис. 1.

* Работа выполнена при поддержке программы АВЦП, грант №7142.

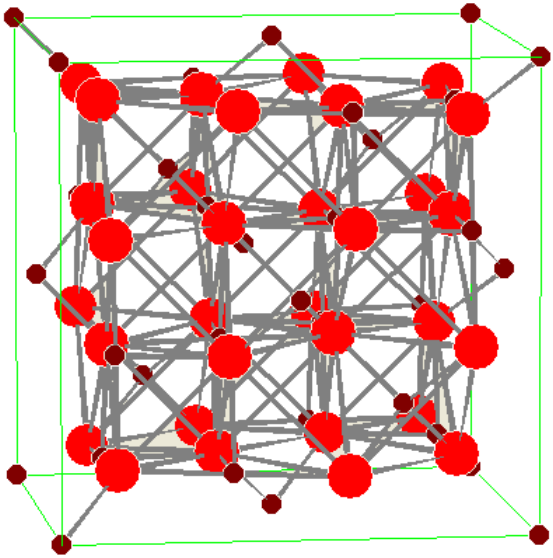


Рис. 1. Элементарная ячейка

В элементарной ячейке содержится 8 магнитоактивных ионов в тетраэдрической подрешётке и 16 в октаэдрической подрешётке, ориентированных антипараллельно. Расчёты проводились для частиц диаметром 3, 4 и 6 элементарных ячеек, содержащих 365, 838 и 2728 магнитоактивных атомов соответственно. Во всех случаях толщина возмущённого поверхностного слоя частицы бралась равной половине элементарной ячейки.

В расчётах используется модельный гамильтониан вида [2]:

$$H = -\sum_{i\neq j\in c} J^{c}_{ab}\vec{S}_i\vec{S}_j - \sum_{i\neq j\in c} J^{c}_{bb}\vec{S}_i\vec{S}_j - \sum_{i\neq j\in s} J^{s}_{ab}\vec{S}_i\vec{S}_j - \sum_{i\neq j\in s} J^{s}_{bb}\vec{S}_i\vec{S}_j -$$

$$-K_c\sum_i (S^2_{ix}S^2_{iy} + S^2_{iy}S^2_{iz} + S^2_{ix}S^2_{iz}) - K_s\sum_k (\vec{S}_k\vec{e}_k)^2 - g\mu_B\vec{H}_0\sum_i \vec{S}_i \ ,$$

где индекс ***c*** относится к атомам внутреннего ядра частицы, ***s*** – атомам её поверхностного слоя. Во всех суммах за исключением последней суммирование проводится по первой координационной сфере. Единичный вектор e определяет взаимное расположение спинов в рамках первой координационной сферы рассматриваемого атома в поверхностном слое, где векторы $p_{i,k}$ – радиус-векторы рассматриваемых ближайших соседей:

$$\vec{e}_k = \sum_i (\vec{p}_k - \vec{p}_i) \Big/ \left| \sum_i (\vec{p}_k - \vec{p}_i) \right|.$$

Множители в суммах J – значения соответствующих интегралов обмена; K – значения констант анизотропии; H_0 – нормированная величина магнитного поля; S_i и S_j – спиновые магнитные моменты атомов подрешёток a и b соответственно.

Суть алгоритма Метрополиса [3] заключается в выборе случайного вектора магнитного момента и повороте его на некоторый случайный угол по каждой из трёх осей, после чего вычисляется энергия системы и в случае её уменьшения изменение принимается, алгоритм прекращает работу по совершении заданного количества нерезультативных шагов. Стоит отметить, что при увеличении количества частиц в модели для сохранения заданной точности моделирования необходимо увеличивать количество нерезультативных шагов до остановки на величину, как минимум пропорциональную увеличению числа частиц в модели.

Для сравнения было выбрано три основных варианта распараллеливания: без распараллеливания вычислительной части, с распараллеливанием только функции вычисления энергии текущей спиновой конфигурации частицы и с распараллеливанием вычисления текущей конфигурации и созданием двух конкурирующих копий с разными конфигурациями для каждого шага вычислений, где энергия каждой копии вычисляется в отдельном потоке. Для второго случая были исследованы варианты с разбиением вычислительного алгоритма на 2, 4 и 8 потоков, а для третьего – на 1, 2 и 4 потока для каждой копии, что также даёт 2, 4 и 8 потоков на вычислительный алгоритм.

Для проверки эффективности данных вариантов были выбраны 2 конфигурации ПК на базе процессоров Intel Core i7 860 и Intel Core i5 650, остальные характеристики данных ПК практически идентичны и не являются существенными, поскольку данный алгоритм не требователен к объёму оперативной памяти и не использует во время работы видеоадаптеры, жёсткие диски или периферийные устройства. Представленные процессоры отличаются по количеству ядер и тактовой частоте: 4 ядра 2,8 ГГц каждое для Core i7 860 и 2 ядра 3,2 ГГц каждое для Core i5 650, важно заметить, что данные процессоры поддерживают технологию Hyper-threading, что позволяет запускать по 2 логических процесса на каждом ядре и поднимает производительность на 10 – 30%.

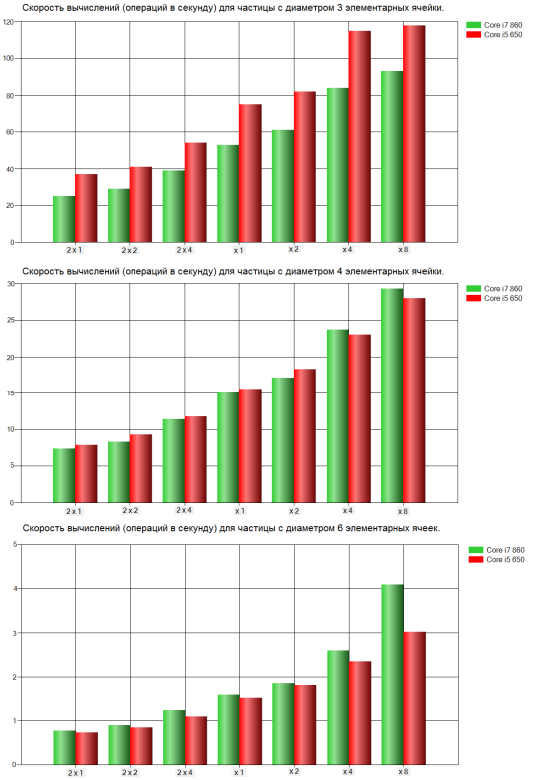


Рис. 2. Влияние размера задачи на эффективность вычислений

В ходе экспериментов было обнаружено, что процессор Core i5 на 20–30% более эффективен при обсчёте модели частицы с диаметром, равным трём элементарным ячейкам. Прирост производительности различных алгоритмов практически не отличается для обоих процессоров в тех случаях, когда число потоков для вычислительной части равно 4 или менее. При использовании 8 вычислительных потоков прирост производительности для Core i5 составляет 3–5%, а для Core i7 – около 10%, относительно варианта с 4 вычислительными потоками, что говорит о низкой целесообразности использования числа вычислительных потоков, большего, чем число потоков, одновременно обрабатываемых процессором. На моделях большего размера разница в производительности между процессорами уменьшается; так, для частицы с диаметром 4 элементарных ячейки разница уже не превышает 10%, а для частицы с диаметром 6 элементарных ячеек разница в сред-

нем 5%, но уже в пользу Core i7. Наиболее существенна разница при использовании 8 потоков для алгоритма вычисления энергии, где большее число ядер позволяет эффективно обрабатывать все потоки, что приводит к опережению по скорости вычислений уже для частицы с диаметром 4 элемент арных ячейки. Также было установлено, что шаги алгоритма с конкурирующими моделями частицы выполняются медленнее, так как содержат больше вычислений, но результирующие значения энергии при сопоставимом времени отличаются несущественно, несмотря на меньшее число выполненных шагов. Результаты экспериментов представлены на рис. 2.

Особенно важным является повышение скорости вычислений при увеличении большего числа ядер с ростом количества частиц в модели, что может быть обусловлено более эффективной загрузкой ядер процессора. Данная закономерность доказывает эффективность и целесообразность применения параллельных вычислений для моделей большого размера, что показано на рис. 3, где a – обозначение размера элементарной ячейки.

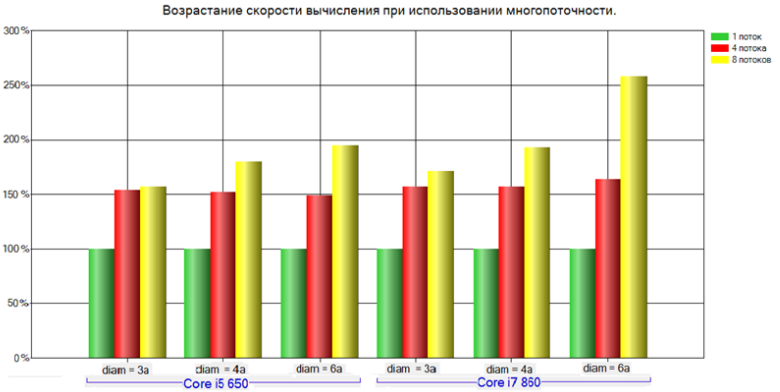


Рис. 3. Возрастание скорости вычислений при использовании многопоточности

Показано, что наиболее эффективным для моделирования на стандартном ПК является распараллеливание только вычисления энергии, так как нет дополнительных затрат на сравнение и изменение конкурирующих моделей частицы. Кроме того, с возрастанием размера используемой частицы время на создание потоков уменьшается в сравнении со временем, затраченным на вычисление энергии для текущей конфигурации, что показывает целесообразность использования параллельных вычислений для задач с большим объёмом расчетов или с

расчетами на очень больших массивах. Проведённые расчёты показывают, что для моделирования магнитных свойств реальной наночастицы диаметром 8–10 нм, что соответствует наночастицам, содержащим 12000–15000 магнитоактивнх атомов, необходимо использование суперкомпьютера, а оптимальным является алгоритм с несколькими конкурирующими моделями частицы и максимально возможным распараллеливанием алгоритма вычисления энергии.

Литература

1. **Albahari B., Albahari J.** C# 4.0 in a Nutshell. – O'Reilly Media, 2010. – 652 p.
2. **Mazo-Zuliaga J., Restrepo J., Munoz F., Mejia-Lopez J.** // J. of Applied Physics. – 2009. – Vol. 105. – P.123907.
3. **Биндер К.** Методы Монте-Карло в статистической физике. – М.:Мир, 1982. – 389 с.

О параллельном решении СЛАУ задач моделирования трехмерных гармонических электромагнитных полей в частотной области*

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

Рассматривается задача моделирования гармонических электромагнитных полей с использованием метода конечных элементов (МКЭ). Построена параллельная версия итерационного алгоритма COCG, а также FGMRES и спектрально-эквивалентного алгебраического мультисеточного предобуславливателя (AMG). В качестве предобуславливателя для COCG и оператора сглаживания для предобуславливателя AMG используется разработанный параллельный SSOR. Представлены результаты экспериментов для набора модельных и практических задач, демонстрирующие эффективность и масштабируемость представленных 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]:

$$\nabla \times \mu_r^{-1} \nabla \times \vec{\mathbf{E}} - k_0^2 \dot{\varepsilon}_r \vec{\mathbf{E}} = -ik_0 Z_0 \vec{J}, \qquad (1)$$

при $\rho = 0$, 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 волнового входа либо металлической стенки на $S_0 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = \vec{\mathbf{n}} \times \vec{\mathbf{E}}_0$ и условия идеального магнитного проводника на $S_1 : \vec{\mathbf{n}} \times \vec{\mathbf{H}} = 0$. Здесь и далее под $\vec{\mathbf{n}}$ понимается вектор внешней нормали относительно грани. Условия согласования на внутренних границах описываются следующими уравнениями:

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

* Работа выполнена при поддержке РФФИ (грант 08-01-00526).

Аппроксимация

В случае аппроксимации МКЭ комплексное уравнение Гельмгольца (1) может быть переписано в вариационной постановке [3]: найти $\vec{E} \in H_{\Gamma}^{rot}$, для которого выполнено

$$\int_{\Omega}\left[\mu_r^{-1}\left(\nabla\times\vec{\mathbf{E}}\right)\cdot\left(\nabla\times\vec{\psi}\right)-k_0^2\dot{\varepsilon}_r\left(\vec{\mathbf{E}}\cdot\vec{\psi}\right)\right]d\Omega=-ik_0Z_0\int_{\Omega}\vec{J}\cdot\vec{\psi},\quad\forall\vec{\psi}\in H_0^{rot},$$

дискретный аналог которой после введения матрицы жесткости и масс

$$S_{i,j}=\int_{\Omega}\mu_r^{-1}\left(\nabla\times\vec{\psi}_j\right)\cdot\left(\nabla\times\vec{\psi}_i\right)d\Omega,\qquad M_{i,j}=\int_{\Omega}\dot{\varepsilon}_r\vec{\psi}_j\cdot\vec{\psi}_i d\Omega$$

выглядит следующим образом:

$$\left[S-k_0^2M\right]u=f,$$

где ψ_j – реберные базисные функции 3-го порядка, предложенные в [4]. Данные уравнения записываются для функций ψ_j, равных 0 на S_0, для остальных коэффициенты вычисляются из условия минимизации отклонения поля на границе S_0. При этом матрица системы уравнений может быть собрана на основе поэлементной технологии [5, 6].

Численные методы

Полученная система уравнений решалась при помощи итерационных методов в подпространствах Крылова COCG с предобуславливателем SSOR и FGMRES с алгебраическим мультисеточным предобуславливателем (AMG) [7].

Алгебраический мультисеточный предобуславливатель основан на использовании иерархических базисных функций [4]. В этом случае матрица системы для порядка функций *k* имеет вид

$$A_k=\begin{bmatrix}A_{k-1} & \cdots\\ \vdots & \ddots\end{bmatrix}.$$

Тогда переход к системе пониженного порядка осуществляется тривиальным образом. В качестве оператора сглаживания [7] для AMG выбрана также итерация SSOR (с учетом $A=L+D+L^T$):

$$u\leftarrow(D+\omega L)^{-1}\left[(2-\omega)D-(D+\omega L^T)\right]u+\omega f,$$

$$u\leftarrow(D+\omega L^T)^{-1}\left[(2-\omega)D-(D+\omega L)\right]u+\omega f.$$

Можно отметить, что данный предобуславливатель оказывается спектрально-эквивалентным [8].

На первом уровне используется прямой решатель PARDISO из библиотеки Intel® MKL [9]. Данный решатель является параллельным, включая этап решения СЛАУ. Параллелизация итерационных решателей для машин с общей памятью выполнена следующим образом. Решатели COCG и FGMRES используют векторно-векторные операции BLAS уровня 1 – dot, axpy и scale. Данные операции параллелизуются путем распределения элементов $1,\ldots,N$ между потоками, где N – порядок системы. Аналогично распараллеливается операция умножения матрицы на вектор, однако между потоками распределяются строки матрицы. Стоит отметить, что распределение переменных и строк матрицы лучше осуществлять статически для повышения эффективности работы решателя на NUMA-архитектурах.

Наибольшую сложность представляет распараллеливание итерации SSOR, поскольку SSOR требует решения треугольных систем вида. В этом случае имеются зависимости по данным. Для снятия этой проблемы для каждого неизвестного $i \in [1,\ldots,N]$ при решении верхнетреугольной системы вычисляется уровень

$$l_i = 1 + \max_{j>i:\ U_{i,j}\neq 0} l_j.$$

После этого вычисление всех неизвестных, принадлежащих одному уровню, можно производить независимо и распределить эти неизвестные между потоками. Распараллеливание решения нижнетреугольных систем выполняется аналогично.

Тестовые задачи

В качестве одного из тестов рассматривалась модельная задача с расчетной областью в виде волновода с линейными размерами a = 72, b = 34, c = 200 мм. 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 $S_1 : \vec{\mathbf{n}} \times \vec{\mathbf{E}} = 0$. Были заданы следующие параметры:

$\mu_r = 1$, $\varepsilon_r = 2 - 0.1$, $\omega = 2\pi \cdot 3 \cdot 10^9$, $\vec{\mathbf{E}}_0 = \vec{\mathbf{e}}_y \sin(\pi x/a)$.

Аналитическое решение: $\vec{\mathbf{E}}_0 = \vec{\mathbf{e}}_y \sin(\pi x/a) \dfrac{\sin(\gamma z)}{\sin(\gamma c)}$.

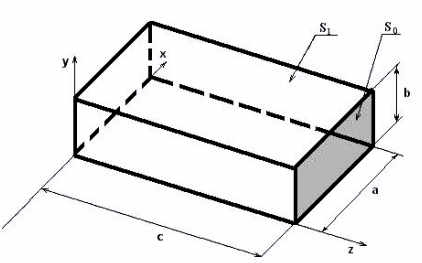


Рис. 1. Расчетная область для тестовых задач

Расчетная область делилась на параллелепипеды, каждый из которых, в свою очередь, разбивался на 6 тетраэдров. Использовались следующие разбиения: 4x2x10, 8x4x20, 15x7x40, 29x14x80, 48x23x134 по осям x, y и z соответственно. Аппроксимация проводилась базисными функциями 3-го порядка.

Таблица 1. Сравнение решателей на модельной задаче, 1 поток

Сетка	*N*	δE	COCG		FGMRES+AMG			PARDISO
			n	*t*, с	n_3	n_2	*t*, с	*t*, с
1	7500	2.1E-3	114	1.35e+0	12	37	7.89e–1	3.45e–1
2	65424	2.8E-4	174	3.66E+1	11	26	1.04e+1	1.09e+1
3	445701	4.3E-5	291	4.68E+2	11	23	9.02e+1	3.38e+2
4	3524883	5.6E-6	604	8.52E+3	11	23	8.54e+2	---
5	16198533	1.2E-6	---	---	11	23	7.21e+3	---

Таблица 2. Масштабируемость решателей на модельной задаче, сетка 15x7x40

Решатель	Количество потоков					
	1	2	4	8	16	32
COCG	1.00	1.50	2.71	4.37	6.71	8.48
FGMRES	1.00	1.37	2.22	3.46	4.70	5.34
PARDISO	1.00	1.86	3.57	6.26	10.2	13.5

Критерием останова итерационного процесса служило условие $\left\|\vec{\mathbf{f}} - \mathbf{A}\vec{\mathbf{u}}_n\right\| < \varepsilon \left\|\vec{\mathbf{f}}\right\|$, $\varepsilon = 10^{-7}$. Алгебраический мультисеточный предобуславливатель обращался приближенно с $\varepsilon = 0.2$. В табл. 1 приведена относительная ошибка δE полученных численных решений с аналитическими решениями, количество итераций *n* и время работы *t* в секундах для соответствующих методов. Также приведены количество

внешних n_3 и суммарное количество внутренних итераций в предобуславливателе n_2 для метода FGMRES. В табл. 2 показана масштабируемость решателей с ростом числа потоков. Тестирование производилось в системе Intel Xeon X7560, 2.27 ГГц, 4 сокета по 8 ядер, объем оперативной памяти – 64 Гб.

Второй рассматриваемый тест – задача электромагнитного каротажа, предоставленная Ю.Г. Соловейчиком. Имеется среда с горизонтальным слоем и вертикальная скважина, в которую вставлен измерительный прибор. Прибор представляет собой трубку с 3 петлями – 1 генераторная при z = 0.5 и 2 приемных при z = 0.1 и z=0.0. В скважине имеется цилиндрическая каверна, заполненная веществом. Смещение от центра скважины до центра прибора – 0.064, радиус скважины – 0.108, радиус прибора – 0.043, толщина каверны – 0.01 (радиус 0.108 – 0.118), координаты каверны по z: (– 0.0725,0.0725), координаты слоя по z: (– 0.425, – 0.275). Вся расчетная область представляет собой куб со стороной 10. Все размеры указаны в метрах. Проводимости сред: прибор – 0 См, скважина и каверна – 5 См, среда – 0.1 См, слой – 0.05 См. Во всей области $\mu_r = 1$, $\varepsilon_r = 1$.

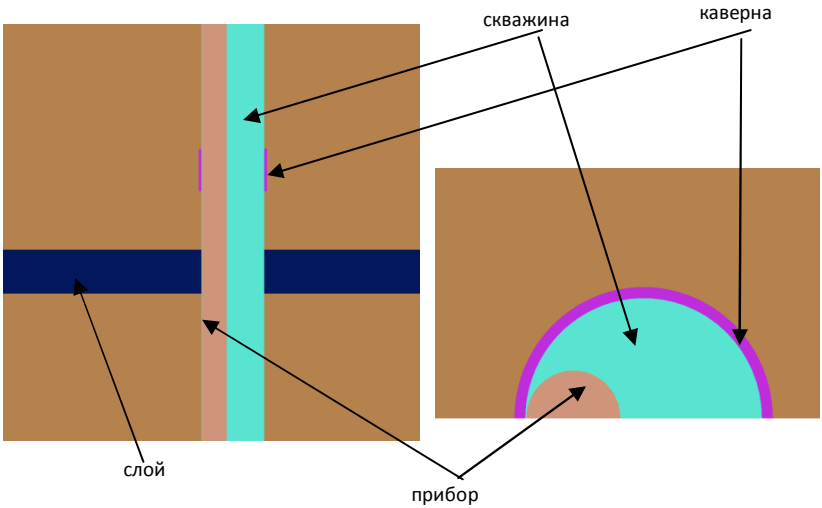


Рис. 2. Расчетная область для второй тестовой задачи

Генерация неравномерной сетки проводилась при помощи утилиты NETGEN [10]. Число тетраэдров в результирующей сетке было равно 388836, порядок системы после аппроксимации – 7058679. Число внешних n_3 и внутренних n_2 итераций в решателе FGMRES было равно

17 и 41 соответственно. Как видно, число итераций увеличилось несущественно по сравнению с модельной задачей. В то же время решатель COCG не показал сходимости в пределах 2000 итераций. Полученная разность фаз ЭДС в петлях приемника совпала с результатом группы НГТУ, полученным методом выделения поля источника, с точностью 1.8%. В табл. 3 представлены результаты тестирования решателя FGMRES на различном числе потоков.

Таблица 3. Время работы и масштабируемость FGMRES для задачи электромагнитного каротажа

Количество потоков	1	2	4	8	16	32
Время работы	5.41e3	2.94e3	1.58e3	9.73e2	6.16e2	4.85e2
Масштабируемость	1.00	1.84	3.42	5.56	8.78	11.2

Выводы

В работе получены следующие результаты. Представлен параллельный SMP решатель FGMRES с алгебраическим мультисеточным предобуславливателем, использующим итерации SSOR в качестве оператора сглаживания. Достигнута масштабируемость 5–10 раз на 32 потоках на методических и практических задачах. С помощью решателя решена задача электромагнитного каротажа.

Литература

1. **Григорьев А.Д., Янкевич В.Б**. Резонаторы и резонаторные замедляющие системы СВЧ. – М.: Радио и связь, 1984.
2. **Bossavit A.** Computational Electromagnetism. – N.Y.: Academ Press, 1987.
3. **Monk P.** Finite element methods for Maxwell's equations. – Oxford: University Press, 2003.
4. **Ingelström P**. A new set of H(curl)-conforming hierarchical basis functions for tetrahedral meshes // IEEE Transactions on Microwave Theory and Techniques. – 2006. – Vol. 54, № 1. – P. 106–114.
5. **Ильин В.П.** Методы и технологии конечных элементов.– Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.
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. **Saad Y.** Iterative Method for Sparse Linear Systems, Second Edition, SIAM, 2003.
8. **Hu J.J., Tuminaro R.S., Bochev P.B. et al.** Toward an h-independent algebraic multigrid method for Maxwell's equations //

SIAM Journal on Scientific Computing. – 2005. – Vol. 27, № 5. – P. 1669–1688.
9. http://software.intel.com/en-us/articles/intel-mkl/ – Math Kernel Library from Intel.
10. **Schöberl J. NETGEN** – An advancing front 2D/3D-mesh generator based on abstract rules // Computing and Visualization in Science. – 1997. – Vol. 1. – P. 41–52.

Параллельная реализация глобальной математической модели переноса трейсера в атмосфере (NIES TM) с использованием двухмерной декомпозиции расчетной области

Е.А. Данилкин, Д.А. Беликов, Ш.Ш. Максютов
Томский государственный университет
National Institute for Environmental Studies, Tsukuba, Japan

Разработан эффективный метод распараллеливания глобальной транспортной модели NIES TM путем двухмерной декомпозиции расчетной области с использованием интерфейса MPI. Полученный параллельный алгоритм удовлетворяет следующим требованиям:

1. *Существенное ускорение при проведении прямого и обратного моделирования переноса трейсеров в атмосфере.*
2. *Достижение приемлемой масштабируемости задачи на системах с распределенной памятью различной конфигурации.*

Модель NIES TM. Основные характеристики модели

NIES TM (the National Institute for Environmental Studies Transport Model) – глобальная трехмерная модель переноса трейсера в атмосфере разработана в Национальном Институте Экологических Исследований (Япония) [1, 2].

Как и в предыдущих версиях [1, 2], для описания переноса трейсеров (инертных газообразных примесей) в атмосфере NIES TM использует транспортное уравнение, записанное в эйлеровой форме:

$$\frac{dq^k}{dt} = \frac{\partial q^k}{\partial t} + \mathbf{V}\cdot\nabla_\sigma q^k + \dot{\sigma}\cdot\frac{\partial q^k}{\partial \sigma} = \frac{\partial}{\partial \sigma}F^k + S^k,$$
$$\nabla_\sigma = \frac{\partial}{R\cos(\varphi)\partial\lambda} + \frac{\partial}{R\partial\varphi}. \qquad (1)$$

NIES TM – офлайн-модель, в которой в качестве входных метеорологических данных используются результаты глобального реанализа (JRA-25/JCDAS) с разрешением 1.25°×1.25°×40 и шагом 6 ч [3]. Модель консервативна, для этого входные горизонтальные массовые потоки балансируются по схеме, описанной в [4], для того чтобы вертикально интегрированные изменения массы воздуха находились в балансе с тенденцией изменения давления на поверхностности.

Аппроксимация адвективных членов выполнена в рамках эйлерова потокового подхода на основе схемы Ван Лира второго порядка. При численной реализации модели используется широтно-долготная разностная сетка, покрывающая весь земной шар. Как следствие, в области полюсов плотность ячеек расчетной сетки в меридиональном направлении существенно выше, чем в области экватора. Чтобы избежать сингулярности вблизи полюсов, вызванной небольшим размером ячеек, предложенная модель использует разреженную долготную сетку, в которой размер ячеек по долготе несколько раз удваивается при приближении к полюсам. Такой подход позволяет преодолеть жесткое ограничение по числу Куранта в приполярных регионах, сохранить достаточно высокий шаг интегрирования по времени и гарантирует приемлемую производительность во время проведения численного моделирования, однако вызывает определенные трудности при организации пересылок для параллельных вычислений. Разрешение горизонтальной сетки вблизи экватора – 2.5°×2.5°. В текущей версии модели применяется гибкая гибридная «сигма-изоэнтропическая» (σ–θ) вертикальная координата с 32 уровнями. Параметризация глубокой конвекции разработана по схеме [5], модель турбулентности основана на методе, описанном в [6], с использованием высоты пограничного слоя из ECMWF ERA-Interim реанализа. Более подробная информация о модели представлена в работах [1, 2].

Препроцессинг

При проектировании параллельных алгоритмов принято выделять фрагменты вычислительного кода модели, которые могут быть подготовлены до начала параллельных расчетов и/или слабо поддаются распараллеливанию. В таком случае эти фрагменты следует выполнять на стадии препроцессинга. В случае распараллеливания NIES TM в режиме препроцессинга рассчитываются и записываются в файлы прямого доступа необходимые метеорологические параметры, список которых включает компоненты скорости ветра, массовые потоки на гранях контрольных объемов, геопотенциальную высоту, коэффициенты для параметризации турбулентности и глубокой конвекции. Размер файлов препроцессинга (обозначены как mfdump-файлы) составляет ~10 Гб для моделирования в течение месяца при том, что объем исходных полей реанализа – ~5 Гб в месяц. Наибольший интерес для наших исследований представляет период около 23 лет (~1988–2011), в течение которого активно моделируется перенос различных трейсеров. Таким образом, использование препроцессинга исключает необ-

ходимость многократного пересчета метеорологических параметров, тем не менее требует дополнительно 2,8 Тб дискового пространства для хранения mfdump-файлов.

Сторонние данные для расчета по модели переноса примеси

Дополнительным преимуществом использования режима препроцессинга является возможность применения входных метеорологических данных, подготовленных по модели глобальной циркуляции атмосферы. Весьма перспективным выглядит использование глобальной версии мезомасштабной модели WRF, которая наряду с широким выбором параметризаций, отлаженными методами инициализации, расчета и постобработки данных обладает важным для задач моделирования переноса свойством сохранения массы.

Метод распараллеливания

Для распараллеливания модели был выбран метод двумерной геометрической декомпозиции расчетной области. В соответствии с этим методом каждому процессорному элементу (ПЭ) выделяется сеточная подобласть (рис. 1), в которой он будет производить вычисления значений сеточной функции и выполнять все необходимые для этого вспомогательные процедуры.

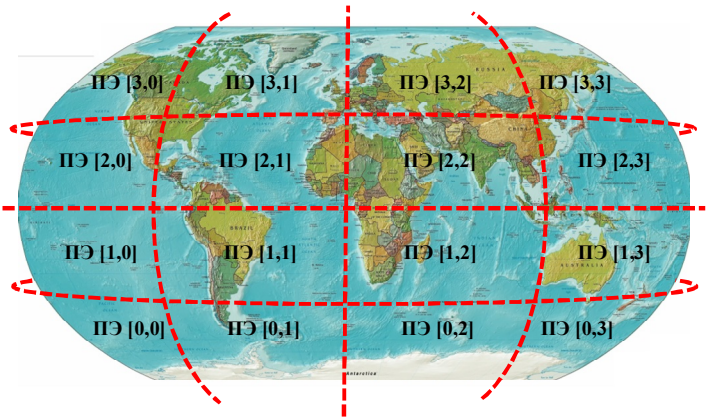


Рис. 1. Геометрическая декомпозиция расчетной области

Следующим после распределения данных по процессорным элементам этапом построения параллельной программы является пла-

нирование коммуникаций, когда устанавливаются связи между блоками, расчеты в которых выполняются параллельно. Вследствие используемого шаблона разностной схемы для вычисления очередного приближения в приграничных узлах требуются значения сеточной функции из области памяти соседнего процессорного элемента. Для этого на каждом вычислительном узле создаются фиктивные ячейки, хранящие данные с соседних вычислительных узлов, а также организованы пересылки этих граничных значений, что необходимо для обеспечения однородности вычислений [7].

Затруднением при планировании коммуникаций является разреженная в направлении от экватора к полюсам горизонтальная сетка модели. Поэтому данные, необходимые соседнему процессорному элементу, предварительно выбираются соответствующим образом из общего массива решения и упаковываются в двумерный массив, пересылаемый соседнему процессорному элементу. Пересылка данных осуществляется с использованием неблокирующей функции передачи сообщений MPI_ISEND, а прием, соответственно, функцией MPI_RECV.

Вторым затруднением при распараллеливании поставленной задачи является необходимость достаточно часто (каждый час модельного времени) выдавать большие объемы результатов моделирования. Это может быть реализовано двумя способами: а) все данные собираются на одном процессорном элементе, с которого затем осуществляется запись результатов расчета в бинарные файлы, б) каждый ПЭ записывает свою часть результатов в общий файл.

Результаты

После завершения разработки и отладки параллельной версии программы проведена оценка эффективности расчетов на нескольких процессорах по сравнению с однопроцессорным вариантом. В первую очередь выполнен ряд вычислительных экспериментов для оценки ускорения работы различных частей программы, осуществляющих схожие функции, в зависимости от количества используемых процессорных элементов. Результаты замеров, а именно время и ускорение работы, представлены в табл. 1 и 2 соответственно. Тестирование проводилось при следующих условиях: расчет переноса одной компоненты примеси (радон ^{222}Rn) в течение одного модельного дня.

Все выполняемые моделью процедуры можно разделить на три типа. Тип первый – процедуры этого типа выполняются быстрее при увеличении количества используемых процессорных элементов. Время

выполнения процедур второго типа не зависит от количества используемых процессорных элементов, например, сюда можно отнести процедуру чтения метеорологических параметров. У процедур третьего типа время работы увеличивается с увеличением количества используемых процессорных элементов.

Таблица 1. Время выполнения отдельных процедур, с

	1	3	9	18	27	36
Чтение метеорологических параметров	0,702	0,832	1,021	1,105	1,069	1,024
Подготовка расчетных полей к новому шагу по времени	6,501	4,009	1,161	0,577	0,377	0,252
Обмен приграничными значениями	0,0	0,271	0,273	0,272	0,277	0,272
Расчет полей концентрации на новом шаге по времени	16,16	7,832	2,408	1,199	0,839	0,616
Общее время без записи результатов	23,364	12,943	4,861	3,151	2,555	2,162
Общее время без записи результатов и считывания метеорологических параметров	22,662	12,111	3,839	2,046	1,486	1,138

Таблица 2. Ускорение выполнения отдельных процедур

	1	3	9	18	27	36
Чтение метеорологических параметров	1	0,84	0,69	0,64	0,66	0,69
Подготовка расчетных полей к новому шагу по времени	1	1,62	5,59	11,27	17,24	25,79
Обмен приграничными значениями		1	1	1	1	1
Расчет полей концентрации на новом шаге по времени	1	2,06	6,71	13,48	19,26	26,23
Общее ускорение без записи результатов	1	1,80	4,81	7,41	9,14	10,80
Общее ускорение без записи результатов и считывания метеорологических параметров	1	1,87	5,90	11,07	15,25	19,91

Из табл. 2 следует, что эффективность распараллеливания процедур подготовки расчетных полей и процедур расчета полей концентрации близка к линейной. Вопросы по эффективности работы вызывает процедура чтения метеорологических данных, эффективность работы ко-

торой замедляется при увеличении числа процессоров (доля от общего времени работы программы значительна – 41,5 %);

Использовать сбор решения на одном процессорном элементе оказалось неэффективно, поэтому в дальнейших расчетах применялся подход, при котором каждый процессорный элемент записывает свой файл результатов. Однако этот подход вызывает определенные сложности при сборе решения, а именно приходится после проведения расчетов собирать все данные в единое целое. Однако такой способ записи дает возможность проводить вычисления с большей эффективностью, а сбор данных можно выполнить, используя одно вычислительное ядро.

Рис. 2. Ускорение работы параллельной программы

Общее ускорение работы полученной параллельной программы представлено на рис. 2 (каждый процессорный элемент осуществляет вывод своей части результатов в отдельный файл). Из рис. 2 видно, что при использовании 9 процессорных элементов параллельная программа работает в 6 раз быстрее, а максимальное ускорение в 15,4 раза получено при использовании 36 процессорных элементов. Также нужно отметить, что эффективность работы параллельной программы уменьшается с увеличением числа используемых процессорных элементов. Так, если при использовании 6 ПЭ эффективность составляет 69 %, то для 36 ПЭ эффективность составляет 42 %.

Заключение

Разработана параллельная версия глобальной транспортной модели NIES TM путем двухмерной декомпозиции расчетной области с использованием интерфейса MPI.

Проведена оценка эффективности работы программного комплекса целиком и отдельных его процедур. Выявлены и устранены слабые места полученной параллельной реализации.

Дальнейшая работа предполагает повышение эффективности работы параллельной программы за счет перехода к использованию параллельных процедур ввода/вывода данных и за счет использования технологии опережающей рассылки.

Литература

1. **Maksyutov S., Patra P. K., Onishi R. et al.** NIES/FRCGC global atmospheric tracer transport model: description, validation, and surface sources and sinks inversion // J. Earth Simulator. – 2008. – Vol. 9. – P. 3–18.
2. **Belikov D., Maksyutov S., Miyasaka T. et al.** Mass-conserving tracer transport modeling onareducedlatitude-longitude grid with NIES-TM // Geosci. Model Dev. – 2011. – Vol. 4. – P. 207–222.
3. **Onogi K., Tsutsui J., Koide H. et al.** The JRA-25 Reanalysis // J. Met. Soc. Jap. – 2007. – Vol. 85(3). – P. 369–432.
4. **Heimann M**., **Keeling C.** A three-dimensional model of atmospheric CO2 transport based on observed winds: 2: Model description and simulated tracer experiments // Geophys. Mon. – 1989. – Vol. 55. – P. 237–275.
5. **Tiedtke M.** A comprehensive mass flux scheme for cumulus parameterization in large scale models // Mon. Weather Rev. – 1989. – Vol. 117. – P. 1779–1800.
6. **Hack J.J., Boville B.A., Briegleb B.P. et al.** Description of the NCAR community climate model (CCM2) // NCAR/TN-382. – 1993. – 108 p.
7. **Данилкин Е.А.** К вопросу об эффективности 3D-декомпозиции при численном решении уравнения переноса с использованием МВС с распределенной памятью // Вестник ТГУ. Механика и математика. – 2008. – № 2 (3). – С. 39–46.

Решение задач механики деформируемого твердого тела с использованием технологии Nvidia CUDA*

Е.П. Евтушенко, Н.И. Карпенко
Институт физики прочности и материаловедения СО РАН, Томск

Разработана программа GeoDA для решения задач механики деформируемого твердого тела в двухмерной постановке с моделями пластичности Мизеса и Друккера–Прагера–Николаевского. В качестве основы для реализации параллельного счета в программе использована технология Nvidia CUDA. Проведены результаты расчетов поведения геосреды при ведении горных работ в угольных шахтах.

Решение актуальных задач геомеханики и моделирования разрушения геоматериалов и горных массивов требует применения высокопроизводительных компьютерных систем вычислений. Такие требования обусловлены как большими временами моделируемых процессов, так и необходимостью применения явных численных схем для реализации специфических моделей деструкции геоматериалов. Поэтому решение подобных задач не обходится без применения вычислительных кластеров [1].

В настоящей работе предпринята попытка использовать для решения задач механики деформируемого твердого тела и геомеханики технологию Nvidia CUDA, показывающую высокие результаты производительности на различного рода модельных физических и математических задачах. С этой целью была написана программа «GeoDA» для решения конечно-разностных уравнений механики сплошных сред с использованием моделей математической теории эволюции, где основные операции дифференцирования запрограммированы с использованием библиотеки Nvidia CUDA [2] и выполняются на графическом процессоре. Программа является почти полным аналогом ранее реализованной программы для кластера СКИФ Cyberia [1]. Использованная в программе модель учитывает внутреннее трение, дилатансию, накопление повреждений и деградацию прочностных характеристик геосреды. Задача решена в двухмерной динамической постановке, использованы треугольные расчетные ячейки в отличие от использованных ранее четырёхугольных [1], что предоставляет большие возможности для сложных геометрических моделей.

* Работа выполнена при поддержке гранта РФФИ № 10-05-00509.

Приведем коротко систему уравнений лагранжевого подхода к описанию движения сплошной среды, использованную как в GeoDA, так и в программе [1]:

– уравнения, выражающие законы сохранения:

$$\rho V = \rho_0 V_0, \quad \rho \dot{v}_i = \frac{\partial \sigma_{ij}}{\partial x^j} + \rho F_i$$

– эволюционные, определяющие уравнения:

$$\dot{\sigma}_{ij}^J = \lambda(\dot{\theta}^T - \dot{\theta}^P)\delta_{ij} + 2\mu(\dot{\varepsilon}_{ij}^T - \dot{\varepsilon}_{ij}^P), \dot{\varepsilon}_{ij}^{\mathrm{p}} = \left(s_{ij} + \frac{2}{3}\Lambda\left(Y - \frac{\alpha}{3}J_1\right)\delta_{ij}\right)\dot{\lambda},$$

$$\dot{\theta}^P = 3\Lambda\dot{\varepsilon}_{eff}^P.$$

Здесь ρ_0, ρ, V_0, V – начальное и текущее значения плотности и объёма частицы материала; x^i – декартовы координаты; v_i, F_i – компоненты векторов скорости перемещений и массовых сил; σ_{ij}, $\dot{\varepsilon}_{ij}^T = \frac{1}{2}\left(\frac{\partial v_i}{\partial x^j} + \frac{\partial v_j}{\partial x^i}\right)$ – компоненты тензоров напряжений и скорости деформации; δ_{ij} – символ Кронекера; λ и μ – коэффициенты Лямэ, Λ и α – коэффициенты дилатансии и внутреннего трения; J_1 и J_2 – первый и второй инварианты тензора напряжений, точка над символом означает материальную производную по времени.

$$\dot{\varepsilon}_{ij}^T = \dot{\varepsilon}_{ij}^E + \dot{\varepsilon}_{ij}^P, \quad \dot{\theta}^T = \dot{\varepsilon}_{ii}^T, \quad \dot{\theta}^P = \dot{\varepsilon}_{ii}^P, \quad \dot{\varepsilon}_{eff}^P = \sqrt{\frac{2}{3}\dot{e}_{ij}^P\dot{e}_{ij}^P}, \quad \dot{e}_{ij}^P = \dot{\varepsilon}_{ij}^P - \frac{1}{3}\dot{\theta}^P\delta_{ij},$$

$$s_{ij} = \sigma_{ij} + P\delta_{ij}, \quad -P = \frac{1}{3}\sigma_{ii}.$$

Предельная поверхность имеет следующий вид [3]:

$$-\alpha P + J_2^{1/2} = Y, \quad Y = Y_0(1 - D(\sigma)),$$

$$D(\sigma) = \int \frac{(\sigma - \sigma_o)^2}{(\sigma^*)^2 t^*} dt \quad \text{для} \quad \sigma > \sigma_0.$$

σ – эффективное напряжение, а σ_0, σ^*, t^* – параметры модели, определяющие пороговое напряжение, с которого начинают накапливаться повреждения, предельное напряжение и характерное время накопления повреждений соответственно; $D(\sigma)$ – интеграл поврежденности среды, при достижении значения $D=1$ материал считается разрушенным.

В соответствии с требованиями вычислительных методов [4] проводится дискретизации задачи по пространству и времени, и задача представляется набором операций над дискретными массивами дан-

ных (сеточная функция). Для численного решения уравнений в частных производных необходимо записать их разностные аналоги над сеточными функциями. В нашем случае производная вычисляется единообразно для каждого элемента сеточной функции, и эта процедура хранится и исполняется на графическом процессоре для каждого элемента сеточной функции. Мы используем явную схему, что позволяет на отдельном временном слое обрабатывать эти массивы параллельно, преобразовывая отдельные элементы каждого массива. Получается, что графические процессоры изначально проектировались так, чтобы эффективно реализовывать подобные алгоритмы.

Таким образом, программа GeoDA имеет слой для операций над сеточными функциями, представляющими разностные аналоги уравнений механики сплошных сред, который работает на CUDA-устройстве, и слой управления, который выполняется на хосте. В свою очередь, слой управления имеет еще слои, которые предоставляют доступ и обработку данных, обеспечивая общую функциональность программного обеспечения.

С использованием GeoDA были выполнены расчеты поведения геосреды при ведении горных работ и расчеты первой и последующих посадок кровли в зависимости от скорости движения забоя в угольной шахте.

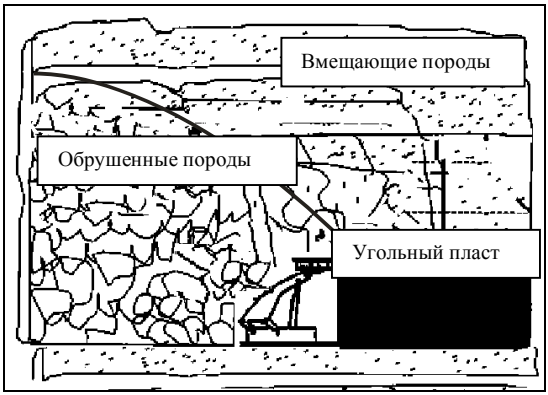
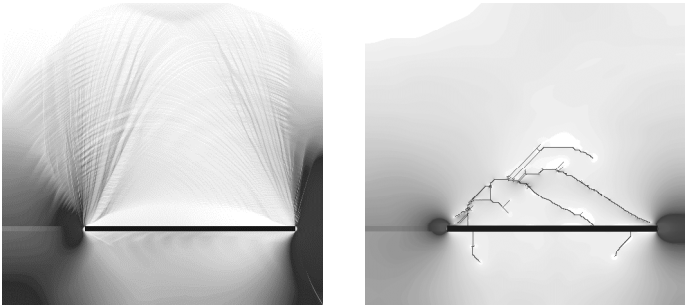


Рис. 1. Геометрия задачи об обрушении кровли в угольной шахте

На рис. 1 схематично представлена постановка задачи о посадке кровли при ведении горных работ. Выработка угольного пласта со временем здесь идет слева направо и образует полость в горном массиве. Под действием сил тяжести и давления вышележащих слоев в

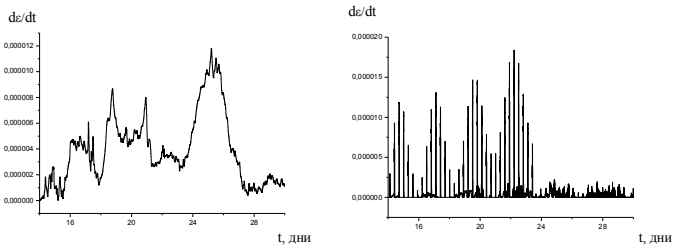
образующейся нависающей кровле идут деформационные процессы, приводящие к её обрушению.

На рис. 2 представлены результаты расчета деформаций пород кровли к моменту достижения выработкой заранее определенной длины (50 м). Причем для расчетов были выбраны различные значения параметров накопления повреждений, что дало возможность описывать геосреду либо как вязкую, либо как хрупкую, в зависимости от требований модели.



а) б)

Рис. 2. Характер развития повреждений в горном массиве над выработкой (чёрная горизонтальная линия). Оттенками серого показано среднее напряжение и полосы неупругого скольжения *а* и магистральные трещины *б*



а) б)

Рис. 3. График мониторинга скорости неупругих деформаций в кровле

Полученные графики мониторинга скорости неупругих деформаций в кровле представлены на рис. 3. Видно, что для вязкой модели (см. рис. 3, *а*) неупругое течение в кровле идет волнами по мере того, как продвигается выработка, и создает все большее напряжение и деформацию в консольно зависающей кровле. Посадка кровли в хрупкой геосреде (см. рис. 3, *б*) идет последовательностью периодических катастроф (подрастаний магистральной трещины), по мере того как про-

двигается выработка, период здесь связан с характером движения выработки. В зависимости от конкуренции отрицательных обратных связей, стабилизирующих деформационный процесс, и положительных обратных связей, обусловленных деградацией нагружаемой среды, сценарий эволюции геосреды может меняться от типичного вязкопластического течения до хрупкого поведения. Таким образом, в модели отражен обязательный этап эволюции геосреды – катастрофа на соответствующем масштабе. Физически этот режим означает прорыв разрушения с меньших масштабов на большие, и увеличение масштаба деструкции всегда развивается как катастрофа в режиме с обострением [5].

Программа GeoDA показала хорошую производительность на задачах геомеханики, обеспечив на видеокарте NVIDIA GTX275 скорость, сопоставимую с двумя четырёхъядерными процессорами Q8200, и будет развиваться далее.

Литература

1. **Макаров П.В., Смолин И.Ю., Евтушенко Е.П. и др.** Моделирование обрушения кровли над выработанным пространством // Физ. мезомех. – 2008. – Т. 11, № 1. – С. 44–50.
2. NVIDIA Programming Guide 1.1. http://developer.nvidia.com/object/cuda.html
3. **Стефанов Ю.П.** Некоторые особенности численного моделирования поведения упруго-хрупкопластичных материалов // Физ. мезомех. – 2005. – Т. 8, № 3.– С. 129–142.
4. **Уилкинс М.Л.** Расчёт упруго-пластических течений // Вычислительные методы в гидродинамике / Под ред. Б. Олдера, С. Фернбаха, М. Ротенберга. – М.: Мир, 1967. – С. 212–263.
5. **Макаров П.В., Смолин И.Ю., Евтушенко Е.П. и др.** Сценарии эволюции горного массива над выработкой // Физ. мезомех. – 2009. – Т. 12, № 1. – С. 75–82.

Численная реализация алгорита SIMPLE на компьютерах с параллельной архитектурой

Д.В. Деги, А.В. Старченко
Томский государственный университет

Рассматривается численное решение уравнений Навье–Стокса с помощью алгоритма SIMPLE на следующих многопроцессорных ЭВМ: системы с общей памятью (используется технология OpenMP), системы с распределенной памятью (MPI), графических процессорах (CUDA).

Введение

К настоящему времени создано большое количество вычислительных алгоритмов, ориентированных на последовательную модель программирования. Однако далеко не всегда удается разработать эффективную параллельную реализацию для многих из них. Поэтому актуальной в настоящее время становится проблема создания новых параллельных алгоритмов решения задач науки и техники на суперкомпьютерах, в частности для имеющих большое теоретическое и прикладное значение задач гидродинамики.

Физическая постановка

Рассматривается стационарное, изотермическое, ламинарное течение несжимаемой вязкой жидкости в прямоугольной полости (каверне) с верхней стенкой, движущейся в своей плоскости со скоростью U [1]. Жидкость, целиком заполняющая каверну, вовлекается в движение силами вязкости. Действие других сил пренебрегается. Значения плотности ρ и коэффициента вязкости μ постоянны. Кроме того, требуется выполнение условия прилипания частиц жидкости к твердой стенке.

Математическая постановка

Рассмотренный физический процесс можно описать с помощью следующих уравнений:

– уравнения движения в проекции на ось Ox:

$$\frac{\partial v_x v_x}{\partial x}+\frac{\partial v_y v_x}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\mu(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial^2 v_x}{\partial y^2});$$

– уравнения движения в проекции на ось Oy:

$$\frac{\partial v_x v_y}{\partial x}+\frac{\partial v_y v_y}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\mu(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial^2 v_y}{\partial y^2});$$

– уравнения неразрывности:

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0.$$

Здесь $\bar{v}=(v_x,v_y)$ – скорость движения жидкости, p – давление.

Граничные условия будут следующими:

– из условия непротекания следует:

$$(v_x)_{x=0}=(v_x)_{x=L_x}=(v_y)_{y=0}=(v_y)_{y=L_y}=0;$$

– из условия прилипания следует:

$$(v_y)_{x=0}=(v_y)_{x=L_x}=(v_x)_{y=0}=0,\ (v_x)_{y=L_y}=U,$$

где L_x и L_y – соответственно длина и ширина каверны; U – скорость движения верхней крышки.

Численный метод решения

Приближенное решение ищется на равномерной, шахматной сетке с помощью метода конечных объемов. Конвективные члены аппроксимируются с помощью схемы «против потока», диффузионные и производные от давления – центрально-разностной схемой. В итоге получается, что разностная схема, аппроксимирующая уравнение неразрывности, имеет вид

$$[(v_x)_{i+1\mathrm{j}}-(v_x)_{ij}]h_y+[(v_y)_{i\,j+1}-(v_y)_{ij}]h_x=0,\ i=\overline{0,Nx}\ ;\ j=\overline{0,Ny};$$

разностная схема для проекции уравнения движения по оси Ox:

$$aP_{i+1\,\mathrm{j}}(v_x)_{i+1\,\mathrm{j}}=aE_{i+1\,j}\ (v_x)_{i+2\,j}+aW_{i+1\,j}(v_x)_{ij}+aN_{i+1\,j}(v_x)_{i+1\,j+1}+$$

$$+aS_{i+1\,j}(v_x)_{i+1\,j-1}+d_{i+1\,j}(p_{ij}-p_{i+1\,j}),\ i=\overline{0,Nx-1},j=\overline{0,Ny};$$

разностная схема для проекции уравнения движения по оси Oy:

$$bP_{i\,j+1}(v_y)_{i\,j+1}=bE_{i\,j+1}(v_y)_{i+1\,j+1}+bW_{i\,j+1}(v_y)_{i-1\,j+1}+bN_{i\,j+1}(v_y)_{i\,j+2}+$$

$$+bS_{i\,j+1}(v_y)_{ij}+l_{i\,j+1}(p_{ij}-p_{i\,j+1}),\ i=\overline{0,Nx},j=\overline{0,Ny-1}\ .$$

Здесь h_x и h_y – шаги сетки по горизонтальной и вертикальной оси соответственно.

Коэффициенты $aP_{ij}, aE_{ij}, aW_{ij}, aN_{ij}, aS_{ij}, bP_{ij}, bE_{ij}, bW_{ij}, bN_{ij}, bS_{ij}$ – известные сеточные функции, зависящие от поля скорости, полученного на предыдущей итерации; $l_{i\,j+1} = h_x / \rho$, $d_{i+1\,j} = h_y / \rho$; Nx и Ny – размер сетки по горизонтали и вертикали соответственно.

Погрешность аппроксимации разностной схемы для уравнения неразрывности имеет второй порядок относительно каждого координатного направления, а для уравнений движения – первый. В силу нелинейности уравнений Навье–Стокса становится невозможным прямое решение системы сеточных уравнений и рассматриваемый процесс приобретает итерационный характер. Для совместного решения системы из трех систем разностных уравнений был применен алгоритм *SIMPLE* [2]. Основные вычислительные трудности данного алгоритма – это решение уравнений для скоростей, а также для поправки давления на каждой итерации, которые выполнялись с помощью метода нижней и верхней релаксации соответственно. В качестве критерия завершения итерационного процесса выбрано выполнение условий: норма поправки давления меньше либо равно заданного ε (0.001), а также выполнения дискретного аналога уравнения неразрывности (с точностью ε). Для организации эффективного параллельного расчета неизвестных методом релаксации был применен способ «*красно-черного*» упорядочивания узлов сетки [3].

Последовательный алгоритм решения задачи

В табл. 1 показано время работы последовательной программы в зависимости от размера используемой сетки. Последовательная версия программы ориентирована на скорость вычислений, так, в ней произведены следующие преобразования:

- организована последовательная выборка элементов из оперативной памяти («кэш-механизмы») [4];
- проведена минимизация количества арифметических операций, в том числе и операций, требующих больше тактов работы процессора (например, деления).

Таблица 1. Время работы программы при использовании различных сеток (точность – ε =0.001)

Сетка	Итераций	Время, с
128×128	1004	21
256×256	3176	294
512×512	12160	5037
1024×1024	53713	107655

Из таблицы видно, что при уменьшении каждого размера ячейки сетки в 2 раза число итераций увеличивается, возрастает объем вычислений, и время работы программы увеличивается более чем в 10 раз. Так, на сетке размером 1024×1024 элементов время расчета составляет около 30 ч.

Параллельный алгоритм решения задачи в среде OpenMP

В качестве системы с общей памятью использовался один вычислительный узел кластера **СКИФ Cyberia** [5], имеющий следующие характеристики: 8 Гб оперативной памяти, два двухъядерных процессора Intel Xeon 5150 Woodcrest с тактовой частотой 2,66 ГГц. В системе OpenMP выполнялось распараллеливание циклов. Исходная последовательная программа состоит в основном из следующих друг за другом циклов, тем самым упрощается сам процесс распараллеливания. Для трудоёмкого в плане вычислений цикла (занимающего на каждой итерации более 20% процессорного времени выполнения всей итерации) создается отдельная параллельная область с помощью конструкции #pragma omp parallel for [6], в которой цикл распараллеливается. Для оставшихся менее трудоемких циклов (коррекция скорости, давления, проверка критерия завершения итерационного процесса) создается единая параллельная область, и в ней уже идет распараллеливание циклов. Итерации цикла распределялись в статическом режиме. В табл. 2 показано получающееся ускорение.

Таблица 2. Ускорение OpenMP программы

Количество вычислительных ядер	Ускорение при использовании различных сеток			
	128×128	256×256	512×512	1024×1024
2	1.6	1.7	1.8	1.5
4	2.9	3.2	3.4	2.2

Из табл. 2 видно, что при увеличении количества ядер ускорение увеличивается. Лучшее ускорение получается на сетке 512×512.

Параллельный алгоритм решения задачи в среде CUDA

Расчеты проводились на видеокарте GTX 260, состоящей из 192 потоковых ядер с тактовой частотой 1,24 ГГц и объемом памяти 2 Гб. Распараллеливание рассматривалось по принципу геометрической декомпозиции. Использовалась двумерная декомпозиция данных на сеточной области. При организации вычислений на видеокарте важной является работа с глобальной памятью [7], также как и для современных процессоров являются важными «кэш-механизмы». Оптимизация работы с глобальной памятью включает в себя следующее: использование общей памяти, объединение запросов на запись и считывание из глобальной памяти и т.д. При построении параллельного алгоритма применялись следующие действия по оптимизации работы с памятью:

- для уменьшения повторных запросов из мультипроцессоров в глобальную память данные сначала копировались в их общую память, а уже затем использовались в вычислениях (с учетом конфликтов банков);
- с целью минимизации обменов данных между ОЗУ и глобальной памятью GPU все вычисления происходили на видеокарте. После завершения каждой итерации из GPU в ОЗУ копировалось всего два значения для проверки критерия завершения итерационного процесса центральным процессором;
- использование блоков, размер которых кратен числу 16, для объединения запросов в глобальную память.

В табл. 3 показано время работы и ускорение CUDA-программы по сравнению с последовательной версией программы.

Таблица 3. Время работы и ускорение CUDA программы

Сетка	Время, с	Ускорение
128×128	1.7	12.4
256×256	13	22.6
512×512	136	37
1024×1024	2240	48

Как видно из табл. 3, с увеличением плотности узлов сетки ускорение возрастает. На сетке 1024×1024 ускорение уже достигает 48 и время вычислений сокращается до ~ 37 мин.

Параллельный алгоритм решения задачи в среде MPI

Вычисления проводились на кластере СКИФ Cyberia. Библиотека MPI представляет собой достаточно большой набор функций [4], с помощью которых выполняются различные действия, такие как обмен значений между процессами, создание декартовой топологии, объявления производных типов данных. Так, для рассматриваемой задачи применялись:

- функции, совмещающие пересылки и прием передаваемых данных (MPI_Sendrecv);
- для удобства передачи и приема набора чисел объявлялись специальные типы (MPI_Type_Vector), с помощью которых передача или прием осуществлялись как с одним значением;
- распараллеливание рассматривалось по принципу геометрической декомпозиции (использовалась двумерная декомпозиция данных), для этого вся совокупность загружаемых процессоров разбивалась с помощью функции создания декартовой топологии (MPI_Cart_create).

В процессе вычислений каждое вычислительное ядро хранило в доступной оперативной памяти лишь только обрабатываемые данные, таким образом, это позволило запускать на кластере задачи с большим объемом данных, не ограничиваясь объемом ОЗУ одного узла.

В табл. 4 показано ускорение MPI-программы.

Таблица 4. Ускорение MPI-программы

Кол-во вычислительных ядер	Ускорение при использовании различных сеток			
	128×128	256×256	512×512	1024×1024
2	1.8	1.9	1.9	1.7
4	3.3	4	2.6	2.1
8	4.8	7.1	8.2	4.3
16	6.6	11.8	16.6	11.2
32	7.5	18.4	29	32.2
64	9.1	24.5	46.6	74

При увеличении количества вычислительных ядер ускорение на всех сетках увеличивается. На сетке размером 1024×1024 элементов при использовании 64 вычислительных ядер ускорение равно 74 (наблюдается эффект сверхлинейного ускорения), тем самым время работы последовательной программы сокращается до ~ 24 мин.

Заключение

Таким образом, в данной работе разработаны эффективные параллельные реализации алгоритма вычислительной гидродинамики SIMPLE для многопроцессорных ЭВМ с общей и распределенной памятью, а также с гибридной архитектурой, объединяющей использование CPU и GPU. Для рассмотренных размеров сетки производительность параллельной программы на видеокарте GTX 260 выше, чем производительность параллельной MPI-программы, работающей на 32 ядрах (16 процессоров Intel Xeon 5150) linux-кластера СКИФ Cyberia. С другой стороны, наименьшее по объему кода изменение по сравнению с последовательной программой оказалась в OpenMP-программе (добавлено всего 12 строк). MPI-программа в 1,6 раза больше текста последовательной программы (с частичным изменением текста), параллельная CUDA-программа – в 2,4 раза с полным преобразованием кода последовательной программы.

Литература

1. **Лойцянский Л.Г.** Механика жидкости и газа: Учеб.для вузов. – 6-е изд., перераб. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 840 с.
2. **Патанкар С.** Численные методы решения задач теплообмена и динамики жидкости. – М.: Энергоатомиздат, 1984. – 124 с.
3. **Ортега Дж.** Введение в параллельные и векторные методы решения линейных систем/ Под ред. Х.Д. Икрамова. – М.: Мир, 1991. – 367с.
4. **Беликов Д.А.** Высокопроизводительные вычисления на кластерах: Учеб. пособие/ Д.А. Беликов, И.В. Говязов и др.; под ред. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2008. – 198 с.
5. http://skif.tsu.ru
6. **Левин М.П.** Параллельное программирование с использованием системы OpenMP. – М.: БИНОМ, 2008. – 118 с.
7. **Боресков А.В.** Основы работы с технологией CUDA,/ А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.

Эффективное сжатие цифровых изображений с применением высокопроизводительных вычислительных систем

С.С. Кулбаев, И.В. Бойченко, В.В. Голенков
Томский государственный университет систем управления и радиоэлектроники

При сжатии качество и размер цифровых изображений всегда являются противоречивыми критериями. В данной работе предлагается способ увеличить коэффициент сжатия изображений и при этом уменьшить потери качества. Существенное увеличение коэффициента сжатия цифровых изображений достигается путём применения параллельного алгоритма сжатия на основе фракталов с применением высокопроизводительных вычислительных систем (ВПВС).

Введение

В последние годы для достижения высокой степени сжатия изображений было предложено множество методов. Среди них очень перспективным методом сжатия является метод фрактального сжатия изображений [1]. Большую ценность представляют алгоритмы фрактального сжатия, способные работать в режиме реального времени, однако большинство из них требует много времени для обработки данных в силу трудоемкости. Более быстрые алгоритмы сокращают время работы за счёт ухудшения качества обработки данных и/или коэффициента сжатия (соотношение размер/качество). Таким образом, разработка методов и алгоритмов на основе фрактального сжатия, способных обрабатывать данные за короткий промежуток времени с высоким качеством, является актуальной задачей и представляет научный интерес. Фрактальное сжатие поддается параллельной обработке, что позволяет создавать программное обеспечение для эффективного решения данной задачи на множестве вычислительных узлов.

Организация параллельных вычислений

Параллелизм (одновременное выполнение нескольких операций обработки данных) осуществляется, в основном, введением избыточности функциональных устройств (многопроцессорности) [2]. В этом случае можно достичь ускорения процесса решения вычислительной задачи, если осуществить разделение применяемого алго-

ритма на информационно независимые части и организовать выполнение каждой части вычислений на разных процессорах. Подобный подход позволяет выполнять необходимые вычисления с меньшими затратами времени. Однако максимально возможное ускорение ограничивается числом имеющихся процессоров и количеством «независимых» частей в выполняемых вычислениях, т.е. количеством заданных рангов. Алгоритм фрактального кодирования заключается в разбиении изображения на ранговые блоки и поиске для них на этом же изображении доменных блоков более крупного размера.

Ранговые блоки покрывают все изображение и не накладываются друг на друга, доменные блоки могут накладываться друг на друга и не покрывать все изображение. Доменный блок должен максимально соответствовать ранговому блоку и обязательно быть больше него. Кодирование изображения представляет собой совокупность ранговых блоков и преобразование их в доменные [3].

В параллельном алгоритме сжатия изображений на основе фракталов изображение разбивается на несколько ранговых блоков. Каждый блок обрабатывается отдельным потоком. Обработка рангового блока подразумевает поиск соответствия данного рангового блока доменному блоку. В случае, если соответствующий доменный блок не найден, ранговый блок разбивается на несколько меньших блоков, каждый из которых, в свою очередь, обрабатывается отдельным потоком.

Многопоточная обработка (multithread processing)

Одним из преимуществ многопоточной обработки является возможность в многопроцессорных системах одновременно выполнять несколько потоков на разных процессорах, увеличивая тем самым производительность.

Многопоточная обработка изображений с помощью фрактального алгоритма происходит следующим образом. Исходное изображение разбивается на ранговые блоки и на группы доменных блоков. Затем создаются потоки, равные количеству ранговых блоков, и для каждого из ранговых блоков создаётся по потоку. Потоки работают независимо друг от друга. Каждый поток, в свою очередь, порождает по четыре потока. Порождённые потоки могут также порождать ещё по четыре потока (по мере необходимости). И так далее, до тех пор, пока не будет достигнут минимальный размер блока (рис. 1). Причём чем меньше размер блока, тем меньше потери качества изображения. При его достижении поток записывает в файл не сами данные рангового блока, а структуру данного рангового блока. В состав структуры рангового

блока входят: размер рангового блока; координаты по оси x, y; форма аффинного преобразования; три цветовые составляющие (YUV).

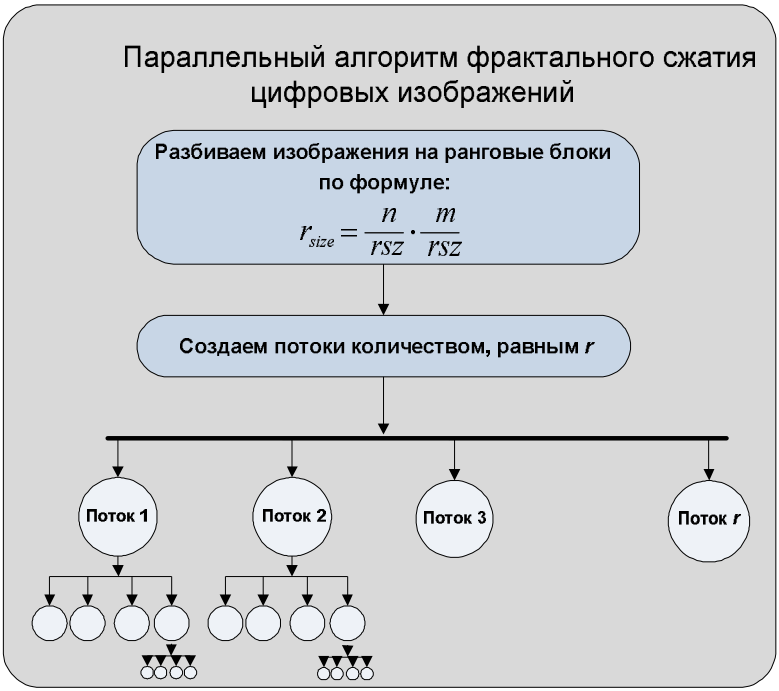


Рис. 1. Схема работы многопоточного параллельного алгоритма фрактального сжатия изображения

Результаты работы

Результаты работы приведены в таблице. В ходе исследования было использовано изображение размером 512x512 пикселей, 24 бит на пиксель. Размер изображения составляет 512x512x24 = 6 291 456 бит, или 786,4 Кбайт. В состав параметров фрактального алгоритма входят: размер рангового блока (8), шаг поиска домена (4), погрешность (среднеквадратичное отклонение 0,01), режим поиска наилучшего домена в пределах среднеквадратичного отклонения. В качестве вычислительных систем был использован четырехъядерный процессор Intel Core i7, поддерживающий до 8 вычислительных потоков на четырех физических ядрах 2,8 ГГц.

Результаты работы фрактального алгоритма и JPEG на четырехъядерном процессоре

Формат	Размер тестового изображения	PNM PSNR, дБ	Время сжатия	Размер выходного файла
BMP	786,4 Кбайт (512x512)	-	-	786,4 Кбайт
JPEG	786,4 Кбайт (512x512)	53.49	0.44 с (ImageMagik один поток)	63 Кбайт
Параллельный алгоритм фрактального сжатия	786,4 Кбайт (512x512)	42.73	0.24 с (многопоточное)	46,3 Кбайт

Также была проведена серия экспериментов на увеличение производительности вычислительных систем в зависимости от количества используемых ядер процессора (рис. 2). Частота одного ядра 2,8 ГГц. Как видно из графика (рис. 3), рост ускорения по мере увеличения числа ядер замедляется.

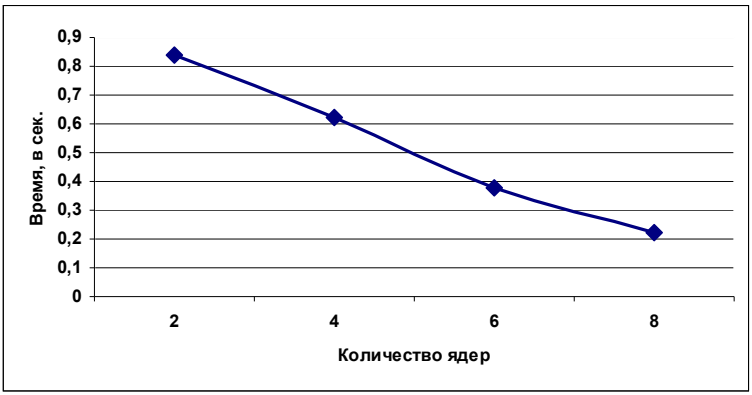


Рис. 2. Время выполнения на многоядерных вычислительных системах

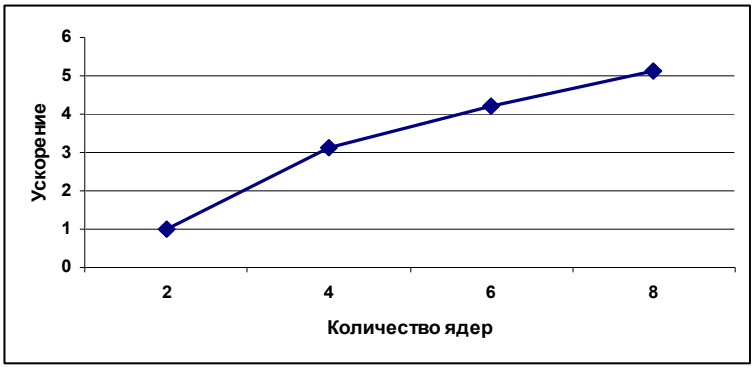


Рис. 3. Коэффициент ускорения от числа ядер

Оптимизация с помощью развёртки циклов

Одним из методов оптимизации программ является метод развертки циклов, который позволяет за счет увеличения количества исполняемого кода увеличить скорость выполнения программ. Техника разворачивания циклов в общем случае сводится к уменьшению количества итераций за счет дублирования тела цикла соответствующее число раз.

При параллельной обработке изображений на основе фракталов одним из основных элементов является итератор. В процессе итерации указывается опорное множество отсчетов изображения, по которым осуществляется итерация, а также задаются тип и параметры преобразования (т.е. угол поворота, зеркальное отражение и т.д.), выполняемого на каждой итерации, и возможные связи с другими параллельно выполняемыми итераторами. То есть параллельно вычисляются яркость, оттенок и контрастность доменного блока, сопоставляемого с ранговым блоком. Процесс, описывающий типовую операцию обработки изображения, представляет собой параллельную композицию итераторов различного типа.

Компилятор gcc (GNU Compiler Collection) поддерживает несколько уровней оптимизации (O1 – минимальная оптимизация, O2 – умеренная оптимизация, O3 – агрессивная оптимизация). Компилятор gcc разворачивает циклы на уровне оптимизации – O2 с ключом **-funroll-loops**, поддерживающим только цикл типа for. Ключ **-funroll-all-loops** поддерживает все виды циклов (например, do/while) [4, 5].

Запуск программы производился в различных вариантах – с установленным флагом оптимизации при компиляции и без него. При

сравнении результатов запуска вариантов программы с установленным флагом оптимизации при компиляции и без включения данного флага можно сделать следующий вывод: программа, скомпилированная с флагом – O2, выполняется в среднем в 1,5 раза дольше (24 мс), нежели программа, скомпилированная с этим же флагом и с развёртыванием циклов (18 мс).

Заключение

В работе реализован многопоточный параллельный алгоритм сжатия цифровых изображений на основе фракталов с применением ВПВС. Так как каждый ранговый блок изображения в алгоритме обрабатывается независимо друг от друга, то можно достичь коэффициента ускорения, близкого к числу ранговых блоков (разумеется, если число вычислительных процессов приблизительно равно числу ранговых блоков, учитывая погрешность среднеквадратического отклонения). Если число процессов превышает количество ранговых блоков, то коэффициент ускорения не будет расти. При проведении оптимизации необходимо в первую очередь учитывать возможности компилятора и аппаратуры (тип процессора и его архитектуру) для получения наилучших показателей производительности.

Литература

1. **Ватолин Д.С.** Фрактальное сжатие изображений // ComputerWorld-Россия. – 1996. – № 6 (23) .
2. **Гергель В.П.** Основы параллельных вычислений для многопроцессорных вычислительных систем / В.П. Гергель, Р.Г. Стронгин. – Н. Новгород: ННГУ, 2003. – 2 изд.
3. **Уэлстид С.Т.** Фракталы и вейвлеты для сжатия изображений в действии: Учеб. пособие / С.Т. Уэлстид. – М.: Триумф, 2003. – 320 с.
4. **Kahle J.A., Day M.N. et al.** Introduction to the Cell multiprocessor // IBM Journal of Research and Development. – 2005. – Vol. 49, № 4/5.
5. Software Development Kit for Multicore Acceleration. Version 3.0. Programming Tutorial. DRAFT // IBM. – 2007.

Параллельная реализация алгоритма K-MEANS ++ с использованием технологии OpenCL

Г.А. Максимов, Н.Н. Богословский
Томский государственный университет

K-MEANS является широко используемым методом кластеризации, в ходе которого предпринимается попытка свести к минимуму среднеквадратичное расстояние между точками кластеров и их центрами. Хотя гарантии точности отсутствуют, простота и скорость работы алгоритма становятся приоритетными факторами применения его на практике. Выбирая начальные центры для K-MEANS очень простым вероятностным алгоритмом, можно получить прирост как скорости работы, так и точности [1]. При больших объемах входных данных время, затрачиваемое на выбор центров, становится велико для использования алгоритма в режиме реального времени, что обязывает его распараллеливать. В качестве технологии параллельного программирования используется язык-надстройка для C++ – OpenCL (Open Computing Language).

Введение

Пусть $X \subset R^q$ – конечное множество и k – целое число. Задача кластеризации заключается в разбиении исходного множества X на k подмножеств, состоящих из элементов, схожих по какому-либо признаку. Такие подмножества называются кластерами, а в качестве признака может служить ближайший центр кластера.

Наиболее популярным методом кластеризации является алгоритм K-MEANS (иногда называемый k-средних), предложенный 1950-х годах почти одновременно Г. Штейнгаузом и С. Ллойдом [5]. Суть алгоритма заключается в стремлении минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$\Phi = \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| x_j - c_i \right\|^2 ,$$

где k – число кластеров; C_i, $i = \overline{1,k}$ – кластеры; c_i – центры масс точек кластеров.

Результат работы алгортма K-MEANS зависит от начальных центров, выбираемых случайным образом, и часто бывает неудовлетворительным. Существует эффективный способ выбора начальных центров

(центройдов) для описанной выше техники кластеризации. Он был предложен Д. Артуром и С. Вассиливицким в 2009 г. и получил название K-MEANS++ [1]. Алгоритм состоит из трех основных этапов:

1. Первый центр выбирается случайным образом из числа входных данных X.
2. Точка $x_i \in X$ становится следующим центром с вероятностью $D(x_i)^2 \Big/ \sum_{x \in X} D(x)^2$, где $D(x) = \min_{c \in C} \|x - c\|$ – расстояние до ближайшего имеющегося центра $c \in C$.
3. Второй шаг повторяется до тех пор, пока не выбрано нужное количество центров.

Выбор центров подобным образом улучшает как скорость работы, так и качество результата K-MEANS [1], но при больших объемах входных данных, которые возникают при обработке изображений, время работы алгоритма K-MEANS++ становится велико, и его сложно использовать в программах, работающих в режиме реального времени. Поэтому необходимо распараллеливать данный алгоритм.

При реализации было решено уйти от вероятностного выбора точки в качестве центра, отдавая предпочтение поиску максимума среди величин $D(x)$. Таким образом, поиск центров представляет собой полный перебор точек среди входных данных, и схема работы алгоритма выглядит следующим образом:

1. Первый центр выбирается случайным образом из числа входных данных X.
2. Точка $x_i \in X$ становится следующим центром, если $D(x_i) = \max_{x \in X} D(x)$.
3. Второй шаг повторяется до тех пор, пока не выбрано нужное количество центров.

Параллельная реализация

Так как параллельную программу планируется запускать на графических процессорах, необходимо учитывать ограничения, возникающие при вычислениях: малый объем оперативной памяти (по сравнению с многопроцессорными вычислительными системами) и значительные временные затраты на копирование данных (на устройство и обратно).

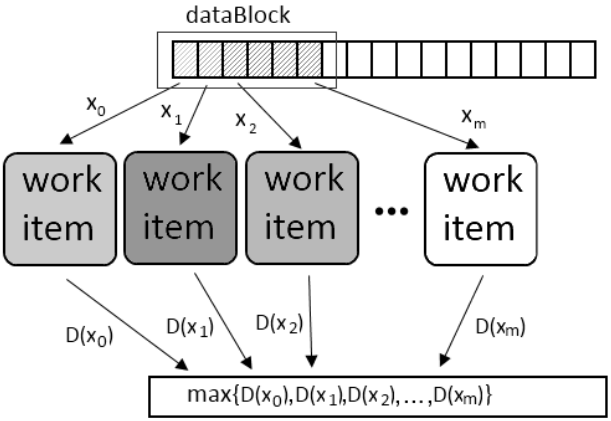


Рис. 1. Схема вычислений параллельной программы

Основная идея параллельной реализации заключается в возможности независимо вычислять величину $D(x)$ для каждой точки $x \in X$ и необходимости делить исходное множество данных X на N непересекающихся подмножеств (блоков) B_n, $n = \overline{1, N}$, размер которых допускает их целостное копирование. Определим подзадачу как отыскание наиболее удаленной от имеющихся центров точки, т.е. выбор одного центра. Для ее решения необходимо последовательно скопировать N блоков данных на устройство. Работу параллельной программы для выбора одного центра можно разделить на следующие основные этапы:

1. Копирование блока B_n в память видеокарты.
2. Параллельное вычисление дистанций $D(x)$ для всех $x \in B_n$.
3. Копирование массива дистанций $D(x)$ в ОЗУ компьютера.
4. Нахождение $D_n(x) = \max\limits_{x \in B_n} D(x)$.
5. Определение максимума среди уже найденных $D_n(x)$.
6. Пункты 1–4 повторяются N раз.

Для решения указанной подзадачи каждая нить (*work-item*) будет обрабатывать одну точку блока входных данных, скопированных на устройство. Схема обработки одного блока данных изображена на рис. 1.

В целях снижения временной стоимости операций копирования задействована закрепленная память (*pinned/non-pageable memory*).

Результаты

Тестирование проводилось с переменным количеством требуемых центров на различных наборах данных на видеокарте Nvidia GTX 460. Среднее время работы (в секундах) последовательного (CPU) и параллельного (GPU) алгоритмов представлено в таблице.

Среднее время работы алгоритма, с, и ускорение

Объем входных данных, количество центров	GPU	CPU	Ускорение
1184, 100	0.4	2.6	6.5
1184, 300	4.4	21.1	4.8
15000, 100	5.2	52.9	10.2
15000, 200	19.3	202.6	10.5
15000, 500	119.7	1352.3	11.3
15000, 1500	1010.7	12213.4	12.1

Модификация K-MEANS++

Была реализована идея еще одного способа поиска центров, ориентированного на графические процессоры и базирующегося на K-MEANS++.

В целях снижения затрат на копирование данных было решено в каждом подмножестве B_n входных данных X, используя K-MEANS++, выбирать $k-1$ точек в качестве центров (первый центр для всех подмножеств общий – c_0). Затем из полученной совокупности снова выбирать $k-1$ центров тем же алгоритмом (рис. 2).

В настоящий момент проводится тестирование эффективности использования центров, выбранных подобным образом как начальных в алгоритме K-MEANS. Однако важно заметить, что получить ускорение работы программы, реализующей данный алгоритм, удастся при достаточно большом наборе входных данных (сотни мегабайт или гигабайты). В ином случае суммарное время выполнения всех операций копирования много меньше времени, необходимого для обработки хотя бы одного блока данных.

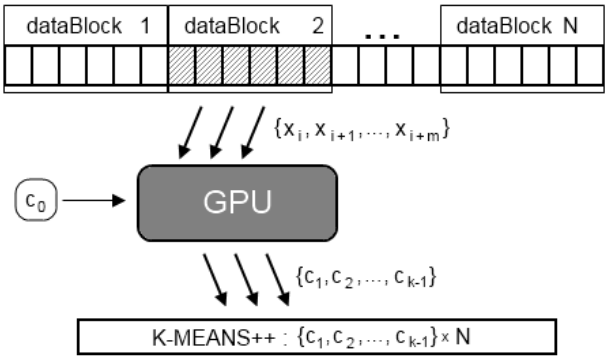


Рис. 2. Модифицированный алгоритм выбора центров

Заключение

Распараллеливание алгоритма K-MEANS++ проводилось с помощью технологии OpenCL, что делает его пригодным для запуска на различных вычислительных и графических устройствах. Ввиду полученного ускорения использование его на практике значительно сократит время выбора центров для K-MEANS, а значит, и время кластеризации в целом.

Литература

1. **Arthur D., Vassilvitskii S**. K-MEANS++: The Advantages of Careful Seeding [Электронный ресурс]. – URL: http://www.stanford.edu/~darthur/kMeansPlusPlus.pdf
2. **Arthur D.** Analyzing and Improving Local Search: K-MEANS and ICP [Электронный ресурс]. – URL: http://www.stanford.edu/~darthur/thesis.pdf
3. **Munshi A., Gaster B.** OpenCL Programming Guide. – USA: Addison-Wesley, 2011. – 603 p.
4. ATI Stream Computing OpenCL Programming Guide. – USA: Advanced Micro Devices, Inc, 2010. – 142 p.
5. **Berkhin P.** Survey of clustering data mining techniques. Techical report, Accrue Software, San Jose, CA, USA, 2002.
6. **Gaster B., Howes L.** Heterogeneous Computing with OpenCL. – USA: Elsevier Inc., 2010. – 288 p.

Неявный итерационный полинейный рекуррентный метод с «плоской» экстраполяцией приращения решения

А.А.Фомин, Л.Н. Фомина
Институт угля СО РАН, Кемерово
Кемеровский государственный университет

Предлагается новый алгоритм полинейного рекуррентного метода, который обладает возможностью естественного распространения на трехмерные задачи. Анализируется его эффективность.

На современном этапе развития численных методов решения многомерных задач математической физики наиболее востребованы методы, позволяющие эффективно решать трехмерные по пространству задачи. Понятно, что трудоёмкость решения подобных задач резко возрастает по отношению к двумерным хотя бы потому, что количество уравнений СЛАУ в этом случае увеличивается как минимум на порядок. В связи с этим возникает настоятельная необходимость распространения неявного итерационного полинейного рекуррентного метода, демонстрирующего высокую эффективность при решении двумерных задач, на трехмерный случай.

Однако здесь имеет место проблема, связанная с тем, что разработанные на данный момент четыре алгоритма метода не обладают свойством естественного распространения на системы уравнений с семью диагональными матрицами. Иными словами, использование этих алгоритмов для решения пространственных задач связано с неоправданно большим увеличением доли приближенных преобразований исходной матрицы при применении принципа компенсации и, соответственно, снижением доли эквивалентных преобразований. А, как известно, высокая эффективность неявного итерационного полинейного рекуррентного метода при решении СЛАУ с пятидиагональными матрицами (двумерный случай) обусловлена тем, что всего лишь один из этапов преобразований матрицы системы является приближенным, а все остальные – эквивалентными [1].

Следовательно, требуется разработать новый способ приближенного замыкания преобразований, основные принципы которого сохранялись бы при переходе от двумерных задач к пространственным. Тем самым минимизировалась бы доля приближенных преобразований матрицы системы в пространственном случае и, соответственно, эффективность метода оставалась бы на высоком уровне. При этом по-

нятно, что прежде чем реализовывать новый алгоритм для трехмерных задач, он должен быть всесторонне протестирован на решениях двумерных.

Такой алгоритм был создан и получил наименование LRe2. Суть его приближенного этапа состоит в приравнивании значений искомой функции в центре сеточного прямоугольника, полученных путем интерполяции значений функции в диагонально противоположных углах прямоугольника (узлах сетки). Из данного соотношения, записанного в плоскости сеточного прямоугольника, получается экстраполяционная формула для значения искомой функции в требуемом узле – вершине прямоугольника.

Также был разработан ускоренный в подпространствах Крылова вариант LRe2 – алгоритм LRe2sK. Сравнительный анализ вновь созданных алгоритмов LRe2 и LRe2sK проведён на решении ряда представительных задач [2, 3]. Необходимо понимать, что LRe2 не позиционируется в качестве еще одного алгоритма решения двумерных задач. Даже если окажется, что он будет в разумных пределах уступать по производительности уже существующим алгоритмам полинейного рекуррентного метода, то с этим можно будет согласиться, памятуя о том, что данный вариант метода обладает возможностью естественного распространения на трехмерные задачи.

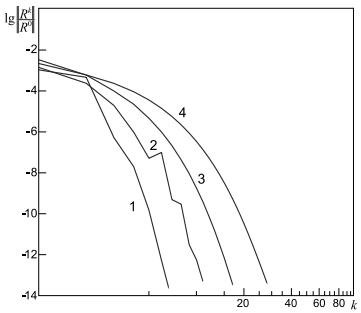


Рис. 1

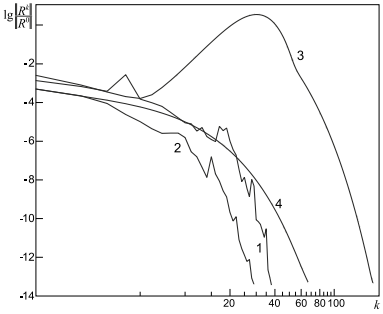


Рис. 2

Зависимость отношения норм невязок от номера итерации при решении первой (рис. 1 – сетка 101×101) и второй (рис. 2 – сетка 1001×1001) тестовых задач. Номера кривых соответствуют решению методом: *1* – LR1sK [2]; *2* – LRe2sK; *3* – LR1 [1]; *4* – LRe2

На рис. 1 приведен график решения первой тестовой задачи [2, 3], разностная дискретизация которой приводит к системе уравнений с самосопряженной матрицей. При решении данной задачи с условиями Дирихле на сетке 101×101 (число обусловленности матрицы СЛАУ $M_A = 5{,}9 \times 10^3$) четыре алгоритма полинейного рекуррентного метода

продемонстрировали высокие показатели по количеству итераций и по времени расчёта: каждый алгоритм затратил меньше 0,1 с для достижения условия $\|R^k\|/\|R^0\| < \varepsilon$. Здесь R – невязка; ε – наперёд заданная точность. В данном случае $\varepsilon = 5\times10^{-14}$. Использование более подробной сетки 1001×1001 ($M_A = 5{,}9\times10^5$) не изменило качество поведения кривых, только увеличив примерно вдвое количество итераций.

Разностная дискретизация второй тестовой задачи [2,3] приводит к системе уравнений с несамосопряженной матрицей. Результаты расчётов на грубой (101×101, $M_A = 9{,}4\times10^4$) и подробной сетках (1001×1001, $M_A = 9{,}4\times10^6$) в целом повторяют предыдущие результаты с той лишь разницей, что на подробной сетке новый алгоритм LRe2 показал несколько более быструю сходимость по отношению к контрольному LR1 как по итерациям, так и по затратам машинного времени. При этом заданная точность $\varepsilon = 5\times10^{-14}$ алгоритмом LRe2sK достигается за 29 итераций и временем 19,2 с, а алгоритмом LRe2 – за 67 итераций и временем 23,7 с.

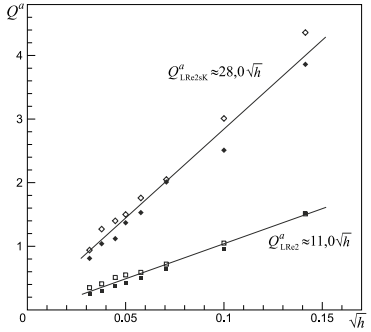


Рис. 3

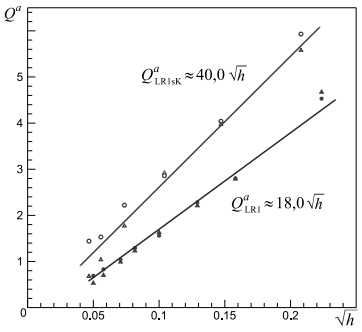


Рис. 4

Предельное значение средней скорости сходимости в зависимости от шага сеточного разбиения области. Расчеты проведены алгоритмами LR1, LR1sK, LRe2 и LRe2sK. Граничные условия первого рода, $\varepsilon = 5\times10^{-14}$, $\Phi^0 = 1$. ■ ♦ – первая тестовая задача, ▪ ◊ – вторая тестовая задача

Эффективность алгоритма метода удобно характеризовать зависимостью предельного значения средней скорости сходимости Q^a от шага сеточного разбиения. Для полинейного рекуррентного метода такие зависимости от квадратного корня из шага сетки очень хорошо аппроксимируются отрезками прямых. Понятно, что чем круче наклон такой прямой, тем эффективнее алгоритм метода. На приведенных графиках рис. 3 и 4 видно, что коэффициенты наклона алгоритма LRe2 (см. рис. 3) приблизительно в $1{,}5 \div 2$ раза меньше соответствующих наклонов контрольного алгоритма LR1 (см. рис. 4).

Для того чтобы более глубоко изучить новый алгоритм, определив его предельные возможности, необходимо попытаться решить им более сложные задачи. Такими задачами являются так называемые задачи Ван-дер-Ворста [4], которые удобно обозначить как вторая и четвертая модельные задачи.

Результаты расчетов второй модельной задачи с точностью $\varepsilon = 10^{-10}$ на оригинальной сетке 150×150 [4] (рис. 5, число обусловленности $M_A = 1{,}8\times10^{7}$) и подробной сетке 1001×1001 (рис. 6, число обусловленности $M_A = 6{,}0\times10^{8}$) говорят о том, что, с одной стороны, темпы сходимости нового алгоритма LRe2 продолжают в 1,5 ÷ 2 раза отставать от алгоритма LR1; а с другой стороны, разрешающие возможности собственно LRe2 на подробной сетке оказались выше, чем у LR1, поскольку методом LR1, в отличие от нового LRe2, не удалось построить решение с наперед заданной точностью. При этом более впечатляющим выглядит результат ускорения алгоритмов в подпространствах Крылова – не в 2 ÷ 3 раза, как для первой и второй тестовых задач, а фактически на порядок.

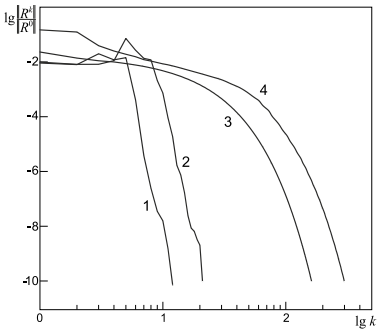


Рис. 5

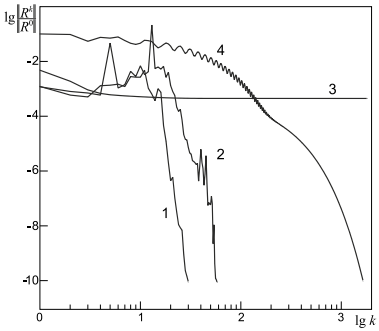


Рис. 6

Зависимость отношения норм невязок от номера итерации при решении второй модельной задачи Ван-дер-Ворста. $\varepsilon = 10^{-10}$. Сетка: рис. 5 – 150×150; рис. 6 – 1001×1001. Номера кривых соответствуют решению методом: *1* – LR1sK; *2* – LRe2sK; *3* – LR1; *4* – LRe2

При расчёте самой сложной, четвёртой модельной задачи Ван-дер-Ворста, в которой перепад коэффициентов при старшей производной доходит до 9 порядков, алгоритм LRe2 позволил получить решение с заданной точностью $\varepsilon = 10^{-10}$ всего за 108 итераций, в то время как LR1 «остановился» на $\|R^k\|/\|R^0\| \approx 2{,}0\times10^{-5}$. На рис. 7 хорошо видна динамика развития процесса сходимости в виде зависимости отношения норм невязок от номера итерации на оригинальной сетке 128×128 ($M_A = 3{,}3\times10^{8}$) [4]. Применение ускоренных алгоритмов LRe2sK и

LR1sK позволило резко сократить количество итераций, причем LRe2sK слегка «обошел» LR1sK: 14 итераций против 15.

Что касается процесса сходимости до заданной точности $\varepsilon = 10^{-10}$ на подробной сетке 1001×1001 ($M_A = 2{,}0\times10^{10}$), то можно сказать, что система оказалась неразрешимой для рассматриваемых релаксационных алгоритмов LR1 и LRe2 (рис. 8), в то время как ускоренные варианты релаксационных алгоритмов LR1sK (28 итераций) и LRe2sK (51 итерация) успешно построили решение с требуемой точностью. Как и ранее, количество необходимых итераций при использовании LR1sK оказалось приблизительно в два раза меньше, чем при использовании LRe2sK.

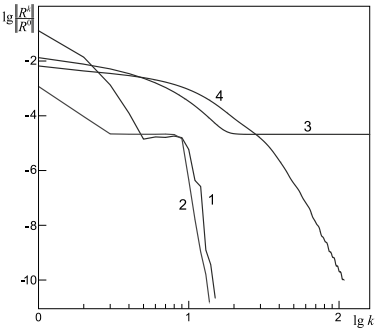


Рис. 7

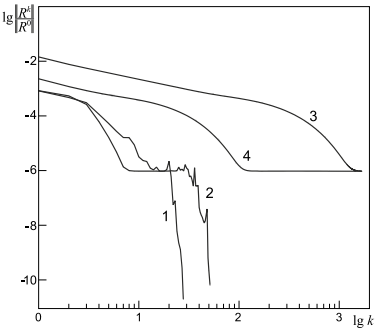


Рис. 8

Зависимость отношения норм невязок от номера итерации при решении четвёртой модельной задачи Ван-дер-Ворста. $\varepsilon = 10^{-10}$. Номера кривых соответствуют решению методом: *1* – LR1sK; *2* – LRe2sK; *3* – LR1; *4* – LRe2

Проведённый сравнительный анализ вновь созданных алгоритмов LRe2 и LRe2sK с их ближайшими аналогами из серии алгоритмов неявного итерационного полинейного рекуррентного метода LR1 и LR1sK позволяет сделать следующие выводы:

1. Алгоритмы LRe2 и его ускоренная версия LRe2sK сопоставимы по эффективности с алгоритмами LR1 и LR1sK, уступая им по итерациям (временам) сходимости в среднем в полтора-два раза. В отдельных случаях (не самосопряженная матрица системы, подробные сетки) эффективность LRe2 (LRe2sK) может превосходить эффективность LR1 (LR1sK).

2. Разрешающие возможности (минимально реализуемые значения отношений нормы текущей невязки к норме начальной невязки) алгоритмов LRe2 и LRe2sK аналогичны возможностям алгоритмов LR1 и LR1sK.

3. Предельное значение средней скорости сходимости при решении задачи с условиями Дирихле для LRe2 составляет $Q^a \approx 11{,}0\sqrt{h}$, а для LRe2sK – $Q^a \approx 28{,}0\sqrt{h}$, где h – сеточный шаг, в то время как для LR1 и LR1sK на тех же задачах они соответственно составляют $Q^a \approx 18{,}0\sqrt{h}$ и $Q^a \approx 40{,}0\sqrt{h}$.

4. С учетом полученных результатов принцип «плоской» экстраполяции, использованный в приближенном этапе алгоритма LRe2, может быть рекомендован для построения «пространственного» варианта неявного итерационного полинейного рекуррентного метода.

Литература

1. **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.
2. **Фомин А.А.** Ускорение полинейного рекуррентного метода в подпространствах Крылова / А.А. Фомин, Л.Н. Фомина // Вестник Томского государственного университета. Математика и механика. – 2011. – № 2. – С. 45–54.
3. **Фомин А.А.** Сравнение эффективности высокоскоростных методов решения разностных эллиптических СЛАУ / А.А. Фомин, Л.Н. Фомина // Вестник Томского государственного университета. Математика и механика. – 2009. – № 2. – С. 71–77.
4. **Van der Vorst H.A.** BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linerar systems // SIAM J. Sci. Stat. Comput. – 1992. – Vol. 13, № 2. – P. 631–644.

Новые типы полуортогональных мультивейвлетов с суперкомпактными носителями и их применение в численном анализе*

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

Рассматривается построение базисных вейвлетов в пространствах эрмитовых сплайнов нечетной степени. Представлены результаты численных экспериментов по их применению для решения дифференциальных уравнений.

Слово «вейвлет» означает в переводе малую, т.е. быстро затухающую волну (всплеск). В математике этим словом называется волновая функция, множество сжатий и смещений которой порождает пространство ограниченных функций на всей числовой оси [1–3]. За счет сжатия вейвлеты способны выявить с разной степенью подробности различие в характеристиках измеренного сигнала, а путем сдвига проанализировать свойства сигнала в разных точках на всем изучаемом интервале. Поэтому вейвлеты иногда называют «математическим микроскопом». При решении задач численного анализа, поскольку вейвлеты преобразуют систему базисных функций с распределенными параметрами в систему с сосредоточенными параметрами, такой базис оказывается гораздо более эффективным с точки зрения обусловленности и сходимости. И. Добеши [1] впервые построила ортонормальные относительно обычного скалярного произведения вейвлеты неполиномиального типа с компактными носителями. Коэн с соавт. построили биортогональные вейвлеты [2]. Недостатками данных вейвлетов является то, что они не имеют аналитического представления и графически похожи на фрактальные кривые. Ограничившись ортогональностью к пространству сплайн-разложений на прореженной сетке, Чуи с соавт. построили полуортогональные сплайн-вейвлеты [3]. Однако недостатком является то, что эти вейвлеты определены на достаточно широком носителе. В работах [4, 5] с помощью вейвлетов Лежандра численно решены интегральное уравнение Абеля и дифференциальное уравнение типа Лане–Эмдета.

В данной статье, используя эти идеи, мы построим базисные вейвлеты на пространствах эрмитовых сплайнов нечетной степени. В

* Работа выполнена при поддержке РФФИ по проекту №11-07-90903_моб_снг_ст.

отличие от классических вейвлетов они обеспечивают выполнение свойства полуортогональности относительно скалярного произведения с производными, причем имеют суперкомпактный носитель, равный носителю базисных сплайнов. И то и другое бывает чрезвычайно важно при их использовании для приближенного решения дифференциальных уравнений по способу Галеркина.

Введем обозначения: R, Z, N – множество действительных, целых и натуральных чисел соответственно; $L_2(R)$ – линейное пространство квадратично-интегрируемых вещественных функций на R; $\langle u,v\rangle := \int_{\mathbf{R}} u(x)v(x)\,dx,\ u(x),v(x)\in L_2(\mathbf{R})$ – 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 $L_2(R)$;

Π_k – множество многочленов степени k (k=0,1,2,…); S_m – пространство эрмитовых сплайнов ($2m$+1)-й степени, удовлетворяющих следующим условиям, ($m > 0$):

а) $s\in C[R]\cap C^1[R]\cap C^2[R]\cap\ldots\cap C^m[R]$;

б) $s(x)\in\Pi_{2m+1},\ x\in(j,j+1),\ j\in Z$.

Пусть $\phi_\alpha, \alpha=\overline{0,m}$ – базисные эрмитовы сплайны ($2m$+1)-й степени:

$$\phi_\alpha(t)=\begin{cases}\omega_\alpha(t) & \text{при}\quad -1\le t\le 0,\\ \xi_\alpha(t) & \text{при}\quad 0\le t\le 1,\end{cases}\quad\text{где}\ \omega_\alpha(t)=(1-t)^{m+1}\sum_{\beta=0}^{m-\alpha}\frac{(m+\beta)!}{\alpha!\,\beta!\,m!}t^{\alpha+\beta},$$

$$\xi_\alpha(t)=(-1)^\alpha\,\omega_\alpha(-t),\quad \alpha=\overline{0,m}.$$

Метод построения полуортогональных сплайн-вейвлетов

Основой для построения вейвлет-преобразования является некоторый набор вложенных пространств. Пусть S представляет собой инвариантное пространство сдвигов, порожденное φ_α, α=0,1,…,m (т.е. S – это пространство эрмитовых сплайнов ($2m$+1)-й степени на бесконечно продолженной в обе стороны сетке с шагом 1). Функция g принадлежит пространству S тогда и только тогда, когда существуют последовательности b_α, α=0,1,…,m на Z, для которых выполняется равенство: $g=\sum_{j\in Z}\left[\sum_{k=0}^{m}b_k(j)\varphi_k(\cdot-j)\right]$.

Пусть $S_1=\{g(2\cdot),\ g\in S\}$ – пространство сплайнов на более густой сетке с шагом ½; тогда $S\subset S_1$. Между S_1 и S остается пространство (разность), которое заполняется вейвлетами.

В этом пространстве будем искать такие функции, которые ортогональны S относительно m-х производных, т.е. мы требуем выполнение равенств:

$$\left\langle \psi_0^{(m)}, \phi_k^{(m)}(\cdot - j) \right\rangle = \left\langle \psi_1^{(m)}, \phi_k^{(m)}(\cdot - j) \right\rangle = \ldots$$
$$\ldots = \left\langle \psi_k^{(m)}, \phi_k^{(m)}(\cdot - j) \right\rangle = 0, \ \forall j \in Z, \ k = \overline{0,m}. \qquad (1)$$

Предположим, что $\psi(x) = \sum_{j \in Z} \left[\sum_{k=0}^{m} b_k(j) \phi_k(2x - j) \right], x \in R$. Введем преобразование Лапласа последовательностей b_k, $k = \overline{0,m}$ на Z, так что положительная или отрицательная степень z означает сдвиг на один шаг по шаблону узлов эрмитового сплайна вправо или влево, соответственно. Пусть

$$q_{01}(z) = \sum_{j \in Z} b_0(2j+1) z^{2j+1}, q_{02}(z) = \sum_{j \in Z} b_0(2j) z^{2j},$$
$$q_{11}(z) = \sum_{j \in Z} b_1(2j+1) z^{2j+1}, q_{12}(z) = \sum_{j \in Z} b_1(2j) z^{2j}, \ldots,$$
$$q_{m1}(z) = \sum_{j \in Z} b_m(2j+1) z^{2j+1}, q_{m2}(z) = \sum_{j \in Z} b_m(2j) z^{2j}.$$

Тогда решение однородного функционального уравнения

$$Q(z)\left(q_{01}(z), q_{02}(z), q_{11}(z), q_{12}(z), \ldots, q_{m1}(z), q_{m2}(z),\right)^T =$$
$$= 0, z \in R \setminus \{0\},, \ j \in Z,$$

где $Q(z) = \left\{ Q_{kj}(z) \right\}, k = \overline{0,m}, \ j = \overline{1, 2m+2}$ – матрица скалярных произведений (1) будет означать, что $\left\langle \psi^{(m)}, \phi_k^{(m)}(\cdot - j) \right\rangle = 0, k = \overline{0,m}.$

Если ранг полученной однородной системы равен m+1, то она имеет m+1 независимых решений. Эти m+1 независимых решений порождают m+1 материнских вейвлетов $\psi_\alpha, \alpha = \overline{0,m}$, удовлетворяющих условию (1).

По представленному выше алгоритму для каждого конкретного значения m можно построить эрмитовы сплайн-мультивейвлеты (2m+1)-й степени, полуортогональные относительно скалярного произведения $\left\langle u^{(m)}, v^{(m)} \right\rangle$. В частности, в работе [6] был рассмотрен случай m=1, и построенные мультивейвлеты имели суперкомпактный носитель [–1, 1]. При этом один из них симметричный, а другой антисимметричный.

Построенные мультивейвлеты (2m+1)-й степени можно применить для численного решения дифференциальных, интегральных, интегро-

дифференциальных уравнений, при решении которых методом Галеркина от пробной функции требуется существование непрерывной m-й производной [7].

Сплайн-мультивейвлеты седьмой степени

Пусть при m=3 ϕ_1, ϕ_2, ϕ_3 и ϕ_4 – базисные эрмитовы сплайны седьмой степени. Аналитическое представление эрмитовых мультивейвлетов седьмой степени имеет вид (рис. 1) [8]:

$\psi_1(t)=\phi_1(2t+1)+\phi_1(2t-1)-42\phi_3(2t+1)-42\phi_3(2t-1)$;

$\psi_2(t)=\phi_2(2t+1)+\phi_2(2t-1)-150\phi_4(2t+1)-150\phi_4(2t-1)$;

$\psi_3(t)=20\phi_1(2t+1)+20\phi_1(2t-1)+147\phi_2(2t+1)-147\phi_2(2t-1)-1680\phi_3(2t)$;

$\psi_4(t)=-\phi_1(2t+1)+\phi_1(2t-1)-7\phi_2(2t+1)-7\phi_2(2t-1)-1680\phi_4(2t)$.

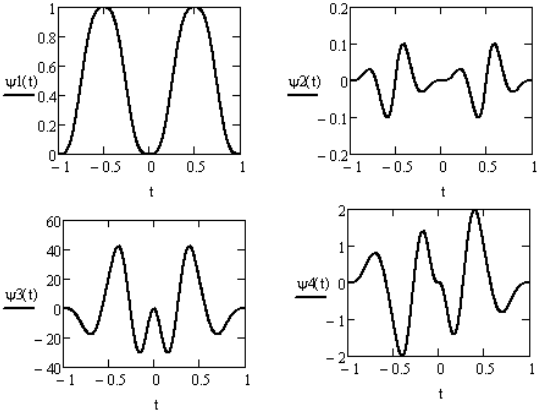


Рис. 1. Графики эрмитовых мультивейвлетов седьмой степени

Носителями построенных вейвлетов ψ_1, ψ_2, ψ_3, ψ_4 является отрезок [–1,1], они удовлетворяют равенствам

$$\langle \psi'''_1, \phi'''_m(-j)\rangle=\langle \psi'''_2, \phi'''_m(-j)\rangle=\langle \psi'''_3, \phi'''_m(-j)\rangle=\langle \psi'''_4, \phi'''_m(-j)\rangle=0,$$
$$m=1,2,3,4.\ \forall j\in\mathbf{Z}, \qquad (2)$$

и их сдвиги генерируют пространство вейвлетов W так, что S_1 является прямой суммой S и W.Кроме того, ψ_1, ψ_3 антисимметричны, а ψ_2, ψ_4 симметричны.

Построим теперь вейвлет-базис в пространстве функций с суммируемой третьей производной $H_0^3(0,1)$.

Множество

$$\Phi_n:=\Phi^{1,n}\cup\Phi^{2,n}=\cup\Phi^{3,n}\cup\Phi^{4,n}, \qquad (3)$$

где $\Phi^{1,n}=\{\phi_1(2^n-j)\colon j=1,\ldots,2^n-1\}$, $\Phi^{2,n}=\{\phi_2(2^n-j)\big|_{(0,1)}\colon j=1,\ldots,2^n\}$,
$\Phi^{3,n}=\{\phi_3(2^n-j)\big|_{(0,1)}\colon j=0,\ldots,2^n\}$, $\Phi^{4,n}=\{\phi_4(2^n-j)\big|_{(0,1)}\colon j=0,\ldots,2^n\}$,
является базисом для V_n (V_n – пространство сплайнов седьмой степени, удовлетворяющих условиям: $n>0$, $v\in C[0,1]\cap C'[0,1]\cap C''[0,1]\cap C'''[0,1]$; $v(0)=v(1)=v'(0)=v'(1)=0$). Элементы Φ_n обозначим через $\{v_1,\ldots,\ v_{2^{n+2}+1}\}$.

Пусть Ψ_n – множество вейвлетов седьмой степени:

$$\Psi_n:=\{\psi_1(2^n-j)\big|_{(0,1)}\colon j=1,\ldots,2^n-1\}\cup\{\psi_2(2^n-j)\big|_{(0,1)}\colon j=1,\ldots,2^n-1\}\cup$$
$$\cup\{\psi_3(2^n-j)\big|_{(0,1)}\colon j=0,\ldots,2^n\}\cup\{\psi_4(2^n-j)\big|_{(0,1)}\colon j=0,\ldots,2^n\}, \qquad (4)$$

и W_n – линейное пространство, натянутое на Ψ_n, очевидно, что $\dim(W_n)=2^{n+2}$. Нетрудно доказать, что $\forall n\in\mathbf{N}$: $\langle v''',w'''_n\rangle=0$, $\langle w'''_m,w'''_n\rangle=0$, $m\neq n$. Из этого следует, что $V_n\cap W_n=\{0\}$. Отсюда

$$\left\| v'''+\sum_{n=1}^{\infty} w'''_n \right\|^2_{L_2(0,1)} = \|v'''\|^2_{L_2(0,1)}+\sum_{n=1}^{\infty}\|w'''_n\|^2_{L_2(0,1)}\ . \qquad (5)$$

Ясно, что V_1 разлагается на $\phi_{1,j}$, $j=1,2,\ldots,8$. Следовательно, $H_0^3(0,1)$ разлагается на g_j, $j=1,\ldots,2^{n+2}+1$, где $g_j=\phi_{1,j}$, при $j=1,\ldots,8$ и $g_{2^{n+2}+1+j}=\psi_{n,j}$, $n\in\mathbf{N}$, $j=1,\ldots,2^{n+2}+1$: $H_0^3(0,1)=V_1\oplus W_1\oplus W_2\oplus\ldots$. Очевидно, что $\left(\psi'''_{n,j}\right)_{n\in N,1\le j\le 2^{n+2}+1}$ является базисом Рисса в $L_2(0,1)$. Аналогично доказывается, что $(g'''_{k,j})_{k,j\in\mathbf{N}}$ является базисом Рисса в $L_2(0,1)$.

Элементы Ψ_n обозначим через $\{w_{2^{n+2}+2},\ldots,w_{2^{n+3}+2}\}$, $n\in N$. Пусть $g_k:=v_k/\|v'''_k\|_2$ при $k=1,2,\ldots,8$, и $g_k:=w_k/\|w'''_k\|_2$ при $k>8$. Тогда $\|g'''_k\|_2=1$ при $n\in\mathbf{N}$.

Применение

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

$$y'''+p_2(x)y''+p_1(x)y'+p_0(x)y=f(x) \qquad (6)$$

с граничными условиями

$$u(0)=u(1)=u'(0)=u'(1)=0, \qquad (7)$$

где $p_0(x)$, $p_1(x)$, $p_2(x)$ – заданные непрерывные функции. Пусть $a(u,v)$ обозначает билинейную форму, $u,v\in H_0^3(0,1)$:

$$a(u,v)=\int_0^1\left(u'''v'''+p_0u''v'''+p_1u'v'''+p_2uv'''\right)dx\ .$$

Тогда вариационная запись (6) – (7) имеет следующий вид:

$$a(u,v)=\langle f,v'''\rangle,\ \forall v\in H_0^3(0,1)\ .$$

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

$$a(u_n,v)=<f,v'''> \quad \forall v \in V_n. \tag{8}$$

Задача (8) имеет единственное решение. Мы предлагаем использовать найденное выше множество вейвлетов $G=\{g_1,\ldots, g_{2^{n+2}+1}\}$ как базис для V_n. С этим базисом для V_n задача (8) может быть дискретизирована следующим образом:

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

Число обусловленности матрицы A_n равномерно ограничено:

$A_n=(a(g_j,g_k))_{1\le j,k\le 2^{n+2}+1}$.

Рассмотрим численный пример:

$y''' = 24t - 180t^2 + 360t^3 + 210t^4$, $y(0)=y'(0)=y(1)=y'(1)=0$.

Точное решение $u(t)=t^4(1-t)^3$. Последовательные ошибки приближенного решения равны:

$\|u(t)-u_4(t)\|_2=2{,}15\times10^{-8}$, $\|u(t)-u_8(t)\|_2=2{,}092\times10^{-14}$,

$\|u(t)-u_{16}(t)\|_2=1{,}291\times10^{-15}$.

Здесь $\left\|u(t)-u_n(t)\right\|_2 = \sqrt{\int_0^1 \left(u(t)-u_n(t)\right)^2 dt}$ – норма разностей.

Литература

1. **Cohen A. Doubeshies, Vial P.** Wavelets on the interval and fast wavelet transforms // Appl. Comput. Harmonic Anal. – 1993. – Vol.1. – P. 54–81.
2. **Dahmen W., Jia R.Q., Kunoth A.** Biorthogonal multiwavelets on the interval: Cubic Hermite splines // Constr. Approx. – 2000. – Vol. 16. – P. 221–259.
3. **Чуи Ч.** Введение в вейвлеты: Пер. с англ. – М.: Мир, 2001. – 412 с.
4. **Yousefi S.A.** Numerical solution of Abel's integral equation by using Legendre wavelets // Applied Mathematics and Computation. – 2006. – Vol. 176. – P. 574–580.
5. **Yousefi S.A.** Legendre wavelets method for solving differential equations of Lane–Emden type // Applied Mathematics and Computation. – 2006. – Vol.181. – P. 1417–1422.

6. **Jia R.-Q., Liu S.-T.** Wavelet bases of Hermite cubic splines on the interval // Advances Computational Mathematics. – 2006. – Vol. 25. – P. 23–39.
7. **Brenner S.C., Scott L.R.** The Mathematical Theory of Finite Element Methods. – New York: Springer, 1994.
8. **Турсунов Д.А., Шумилов Б.М.** Новый тип эрмитовых сплайн-вейвлетов седьмой степени // Материалы 14-го Всероссийского семинара «Моделирование неравномерных систем». – Красноярск, 2011. – С. 176–181.

Применение билинейных сплайн-вейвлетов для аппроксимации поверхностей*

А.Ж. Кудуев, Б.М. Шумилов
Ошский государственный университет
Томский государственный архитектурно-строительный университет

Для аппроксимации билинейными сплайнами функций двух переменных, определенных в прямоугольной области, предложено использовать разложение по базису неортогональных вейвлетов. Для случая прямоугольной таблицы значений функции алгоритм аппроксимации сводится к серии одномерных вейвлет-преобразований по строкам и столбцам таблицы. Для случая непрямоугольной таблицы предлагается использовать метод наименьших квадратов. Обсуждаются способы удаления незначащих коэффициентов разложения с целью сжатия таблиц и сглаживания результирующих поверхностей. Представлены результаты численных экспериментов.

Введение

Билинейное пространство вейвлетов определяется как тензорное произведение пространств сплайн-вейвлетов 1-й степени по каждой переменной. Наилучшим из таковых является пространство полуортогональных сплайн-вейвлетов, определяемых как подмножество сплайнов на густой сетке, ортогональных пространству сплайнов на прореженной сетке [1]. Неортогональные сплайн-вейвлеты [2], определяемые из условия ортогональности многочленам 1-й степени, не на много уступают по качеству сжатия полуортогональным сплайн-вейвлетам, но имеют менее заполненные матрицы вейвлет-преобразования. Преимущество перехода к вейвлет-коэффициентам состоит в том, что часто большое количество коэффициентов оказывается очень малым по величине. Удаление этих малых коэффициентов приводит лишь к незначительным погрешностям при воспроизведении таблицы заданных значений, являя форму сжатия «с потерями».

Если таблица прямоугольная, то использование стандартного подхода соответствует одномерному вейвлет-преобразованию каждой строки значений таблицы. В результате получаются коэффициенты усредняющей прямой и вейвлет-коэффициенты для каждой строки таблицы. Затем эти преобразованные строки рассматриваются так, как

* Работа выполнена при поддержке РФФИ (проект № 11-08-90902 моб_снг_ст.).

если бы они сами являлись данными, и применяется одномерное преобразование к каждому столбцу полученной таблицы. Полученные в результате значения оказываются коэффициентами для базиса, образованного всевозможными тензорными произведениями функций одномерного базиса, четыре из которых представляют усредняющую билинейную поверхность. При этом вейвлет-функции стандартно построенного базиса имеют прямоугольные носители.

Если таблица непрямоугольная и имеет пропуски данных, то коэффициенты билинейного сплайна приходится отыскивать согласно какому-либо универсальному методу аппроксимации. Переход здесь к разложению по вейвлетам принципиально улучшает обусловленность решения, поскольку вейвлеты преобразуют систему базисных сплайн-функций с распределенными параметрами в систему с сосредоточенными вейвлет-параметрами. Применение для аппроксимации метода наименьших квадратов позволяет определить статистическую меру значимости каждого коэффициента согласно критерию *t* Стьюдента для того, чтобы решить, какие вейвлет-коэффициенты должны быть удалены.

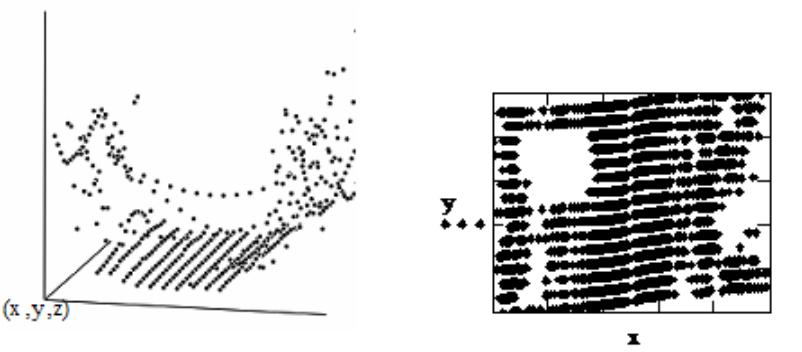
Сглаживание данных лазерного сканирования поверхностей автомобильных дорог

Зачастую расположение предписанных пользователем точек поверхности отличается от узлов прямоугольной сетки и их число значительно больше, чем число коэффициентов сплайна. Например, на рис. 1 показано множество точек (x, y, z), представляющих материалы лазерного сканирования поверхности автомобильной дороги [3], где *а* – «облако» точек сканирования, *б* – горизонтальные проекции точек. Здесь $x=\{x_i,\ i=0,1,\ldots\}$ – локальные координаты точек в поперечном сечении дороги; $y=\{y_i, i=0,1,\ldots\}$ – продольные координаты; $z=\{z_i, i=0,1,\ldots\}$ – координаты рельефа дорожной поверхности (высоты над уровнем моря).

Поскольку данные имеют два преимущественных направления: вдоль и поперек дороги, возникает естественное желание использовать для их аппроксимации параметрические сплайны двух переменных. Поэтому ставится задача трансформировать изгибы автомобильной дороги в прямоугольную область (рис. 2, 3), на которой задано множество точек сканирования *x, y, z*. Предлагаемый алгоритм [4] состоит из четырех шагов.

1. На поверхности рассмотрим семейство линий, обладающих следующими свойствами:

1) Никакие две линии не имеют между собой общих точек. 2) Начало и конец каждой линии находятся в точках, расположенных на противоположных участках границы; два других участка границы включаются в число рассматриваемых линий. Практически построение упомянутого семейства линий сводится к выделению из множества заданных точек поверхности упорядоченных подмножеств L_k.



а) б)

Рис. 1

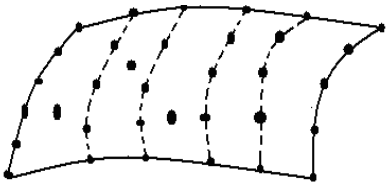


Рис. 2

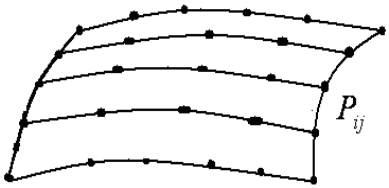


Рис. 3

2. Для каждого подмножества L_k найдем интерполирующий его параметрический сплайн $S_k(x,\bar{s})$, $S_k(y,\bar{s})$, $S_k(z,\bar{s})$ с использованием нормированной параметризации по суммарной длине хорд. На отрезке изменения параметра $\bar{s}[0,1]$ введем равномерную сетку узлов Δ_v: $v_j = j / 2^{Lv}, j = 0, 1,\ldots, 2^{Lv}, L_v \geq 0$, и вычислим координаты точек $P_{kj}^{(1)}\left\{S_k(x,v_j),\ S_k(y,v_j),\ S_k(z,v_j)\right\},\ j=0,\ldots,2^{Lv};\ k=0,\ldots,N_0$.

3. Для каждого j=0,…, 2^{Lv} построим параметрические сплайны $\bar{S}_j(x,\bar{s})$, $\bar{S}_j(y,\bar{s})$, $\bar{S}_j(z,\bar{s})$, интерполирующие точки

$P_{1j}^{(1)}, P_{2j}^{(1)}, ..., P_{N_{0,j}}^{(1)}$. Вновь используется нормированная параметризация по суммарной длине хорд. Далее введем сетку Δ_u: $u_i = i / 2^{Lu}$, $i = 0, 1, \ldots, 2^{Lu}$, $L_u \geq 0$, и определим координаты точек $P_{i,j}\left(x_{ij}, y_{ij}, z_{ij}\right)$:

$x_{ij} = \overline{S}_j\left(x; u_i\right)$, $y_{ij} = \overline{S}_j\left(y; u_i\right)$, $z_{ij} = \overline{S}_j\left(z; u_i\right)$, $i = 0, ..., 2^{Lu}$, $j = 0, ..., 2^{Lv}$.

4. В области [0,1]x[0,1] образуем сетку $\Delta = \Delta_u x \Delta_v$. В качестве декартовых координат поверхности в её узлах (u_i, v_j) принимаем значения x_{ij}, y_{ij}, z_{ij}. Интерполируя их на сетке Δ, получаем для координат искомой поверхности следующие формулы:

$$x = S\left(x; u, v\right), \; y = S\left(y; u, v\right), \; z = S\left(z; u, v\right).$$

Наличие пропусков данных на проезжей части, вызываемых экранированием проезжающими мимо автомобилями, и разным числом точек на отдельных сканах, вызываемым потерей данных на придорожном ландшафте и повторными отражениями от находящихся на объекте людей, техники, растительности и т. д., осложняет ситуацию. Мы рассматриваем вариант, когда коэффициенты построенного сплайна отыскиваются согласно методу наименьших квадратов, а именно:

$$\sum_i \left[\left(x_i - S\left(x; u_i, v_i\right)\right)^2 + \left(y_i - S\left(y; u_i, v_i\right)\right)^2 + \left(z_i - S\left(z; u_i, v_i\right)\right)^2 \right] \to \min_{x,y,z}.$$

Вейвлет-разложение двумерного билинейного сплайна имеет вид

$$\overline{S}^L\left(u, v\right) = \varphi_u^{L_1} \overline{C}^L \left[\varphi_v^{L_2}\right]^T = \varphi_u^0 \overline{C}_{[0,0]}^{[0,0]} \left[\varphi_v^0\right]^T + \sum_{j=0}^{L_v - 1} \varphi_u^0 \overline{E}_{[0,1]}^{[0,j]} \left[\psi_v^j\right]^T +$$

$$+ \sum_{i=0}^{L_u - 1} \psi_u^i \overline{F}_{[1,0]}^{[i,0]} \left[\varphi_v^0\right]^T + \sum_{i=0}^{L_u - 1} \sum_{j=0}^{L_v - 1} \psi_u^i \overline{D}_{[1,1]}^{[i,j]} \left[\psi_v^j\right]^T,$$

где нижние и верхние индексы у матриц $\overline{C}_{[0,0]}^{[0,0]}, \overline{E}_{[0,1]}^{[0,j]}, \overline{F}_{[1,0]}^{[i,0]}, \overline{D}_{[1,1]}^{[i,j]}$ указывают начальные и конечные значения нумерации строк и столбцов.

Здесь коэффициенты сплайна $\overline{C}_{i,j}^L$ собираются в матрицу вида

$$\overline{C}^L = \begin{bmatrix} \overline{C}_{0,0}^L & \overline{C}_{0,1}^L \ldots & \overline{C}_{0,2^{L_2}}^L \\ \overline{C}_{1,0}^L & \overline{C}_{1,1}^L \ldots & \overline{C}_{1,2^{L_2}}^L \\ \vdots & \ddots & \vdots \\ \overline{C}_{2^{L_1},0}^L & \overline{C}_{2^{L_1},1}^L \ldots & \overline{C}_{2^{L_1},2^{L_2}}^L \end{bmatrix}.$$

Выражение $\varphi_u^0 \overline{C}_{[0,0]}^{[0,0]} \left[\varphi_v^0\right]^T$ представляет собой запись усредняющей билинейной поверхности; базисные сплайн-функции и вейвлет-

функции на данном уровне L записываются в виде двух матриц-строк: $\varphi_u^{L_u} = \left[N_0^{L_u}, N_1^{L_u}, ..., N_{2^{L_u}}^{L_u} \right]$, $\psi_u^{L_u} = \left[M_1^{L_u}, ..., M_{2^{L_u-1}}^{L_u} \right]$ и $\varphi_v^{L_v} = \left[N_0^{L_v}, N_1^{L_v}, ..., N_{2^{L_v}}^{L_v} \right]$, $\psi_v^{L_v} = \left[M_1^{L_v}, ..., M_{2^{L_v-1}}^{L_v} \right]$, для направлений u и v, соответственно, и $N^L{}_i(v) = \varphi_1(v-i)\ \forall i$, $M^L{}_1(v) = \Psi_{21}(v-1)$, $M^L{}_i(v) = \Psi_2(v-i)$, $i = 2, 3, \ldots, 2^{L-1}-1$, $M^L_{2^{L-1}}(v) = \Psi_{21}\left(2^L - v + 1\right)$, $v = 2^L u + 1$, по каждому направлению;

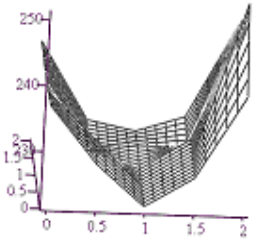
$$\varphi_1(t) = \begin{cases} t, & 0 \le t \le 1, \\ 2-t, & 1 \le t \le 2, \\ 0, & t \notin [0,2]; \end{cases}$$

$$\Psi_{21}(t) = \frac{\sqrt{3}}{9}\left(6\sqrt{2^L}\phi_1(2t+1) - 5\phi_1(2t) + 2\phi_1(2t-1)\right),$$

$$\Psi_2(t) = \frac{1}{2}\phi_1(2t+1) - \phi_1(2t) + \frac{1}{2}\phi_1(2t-1).$$

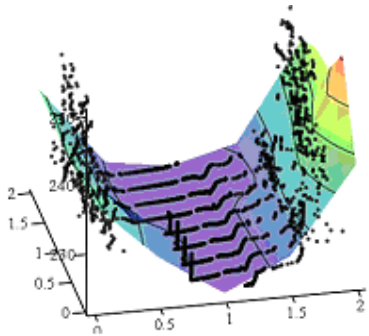
Результаты численных экспериментов

На рис. 4, 5 представлен график билинейного сплайн-восполнения полученного МНК-вейвлет-разложения на густую прямоугольную сетку, а также результат наложения исходных точек на полученную поверхность.



(a,c,zcalc)

Рис. 4



(x,y,z),(a,c,zcalc)

Рис. 5

На рис. 6 представлен соответствующий график стандартизированных ошибок аппроксимации, позволяющий указать точки скана, не принадлежащие проезжей части и рекомендуемые для фильтрации.

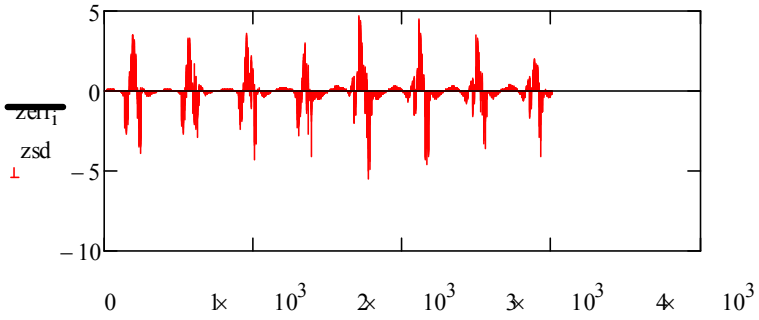


Рис. 6

Литература

1. **Столниц Э., ДеРоуз Т., Салезин Д.** Вейвлеты в компьютерной графике: Пер. с англ. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002. – 272 с.
2. **Koro K., Ade K.** Non-orthogonal spline wavelets for boundary element analysis // Engineering Analysis with Boundary Elements. – 2001. – Vol. 25. – P. 149–164.
3. **Турсунов Д.А., Шумилов Б.М., Байгулов А.Н., Колупаева С.Н.** Предварительная обработка материалов лазерного сканирования автомобильных дорог // Вестник Томского государственного архитектурно-строительного университета. – 2011. – № 3. – С. 153–163.
4. **Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л.** Методы сплайн-функций. – М.: Наука. Главная редакция физико-математической литературы, 1980. – 352 с.

Преобразование, сжатие и улучшение данных на прямоугольной сетке бикубическими мультивейвлетами

Э.А. Эшаров, Б.М. Шумилов, У.С. Ысманов, Ж. Абдыкалык кызы
Томский государственный архитектурно-строительный университет
Ошский государственный университет

Для аппроксимации функций двух переменных бикубическими сплайнами рассматривается применение метода наименьших квадратов на основе разложения по бикубическим мультивейвлетам. Приведены результаты численных экспериментов.

Введение

В конце прошлого века возникло и успешно развивается новое и важное направление в теории и технике обработки сигналов, изображений и временных рядов, получившее название вейвлет-преобразования (ВП), которое хорошо приспособлено для изучения структуры неоднородных процессов. Термин «вейвлет» (wavelet) ввели в своей статье Гроссманн (Grossmann) и Морле (Morlet) в середине 80-х годов XX в. в связи с анализом свойств сейсмических и акустических сигналов. Их работа послужила началом интенсивного исследования вейвлетов в последующее десятилетие рядом ученых таких, как Добеши (Dobechies), Мейер (Meyer), Малл (Mallat), Фарж (Farge), Чуи (Chui) и др.

В математическом смысле вейвлетом называется малая, т.е. быстро затухающая волновая функция, множество сжатий и смещений которой порождает пространство ограниченных функций на всей числовой оси [1–3]. За счет сжатия вейвлеты способны выявить с разной степенью подробности различие в характеристиках измеренного сигнала, а путем сдвига проанализировать свойства сигнала в разных точках на всем изучаемом интервале. Поэтому вейвлеты называют математическим микроскопом. При анализе нестационарных сигналов за счет свойства локальности вейвлеты обеспечивают существенное преимущество перед преобразованием Фурье, которое дает только глобальные сведения о частотах исследуемого сигнала, поскольку используемые при этом базисные функции (синусы и косинусы) определены на бесконечном носителе. И. Добеши впервые построила ортонормальные вейвлеты неполиномиального типа с компактным носителем [1].

Чуи с соавт. построили полуортогональные сплайн-вейвлеты [2]. Коэн с соавт. построили биортогональные вейвлеты [3].

Недостатками данных вейвлетов является то, что они не имеют аналитического представления и графически похожи на фрактальные кривые либо определены на достаточно большом носителе. И то и другое бывает чрезвычайно важно при их использовании для приближенного решения задач численного анализа. В ряде работ уменьшение носителя достигалось построением эрмитовых сплайн-мультивейвлетов, у которых с каждым узлом связано более одной базисной функции. В частности, в [4] были построены полуортогональные кубические мультивейвлеты на суперкомпактном носителе, равном носителю базисного сплайна.

В данной работе рассматривается вычисление бикубических эрмитовых сплайнов на основе разложения по бикубическим мультивейвлетам и приведены численные примеры использования метода наименьших квадратов для аппроксимации данных с пропусками.

Вейвлет-разложение бикубических эрмитовых сплайнов в прямоугольной области

Пусть пространство V^L, $L=[L_1,L_2]$, является тензорным пространством двумерных эрмитовых кубических сплайнов на прямоугольнике $[a_1, b_1]\times[a_2, b_2]$ с равномерной сеткой узлов $\Delta^{Lj}: u_{ii} = a_i + (b_i - a_i)\, i / 2^{Lj}$, $i = 0, 1,\ldots, 2^{Lj}$, $L_i \geq 0$, $j=1,2$ по каждому направлению и всевозможными произведениями функций $N^{Lj}_{i,k}(v) = \varphi_k(v - i)$, $k = 0, 1\ \forall i$, где $v = 2^{Lj}(u - a_j) / (b_j - a_j) + 1$, порожденных сжатиями и сдвигами двух масштабирующих функций $\varphi_0(t)$, $\varphi_1(t)$ (рис. 1) в качестве базисных функций [5]:

$$\phi_0(t) = \begin{cases} t^2(3-2t), & 0 \le t \le 1, \\ (2-t)^2(2t-1), & 1 \le t \le 2, \\ 0, & t \notin [0,2]; \end{cases}$$

$$\phi_1(t) = \begin{cases} -t^2(1-t), & 0 \le t \le 1, \\ (2-t)^2(t-1), & 1 \le t \le 2, \\ 0, & t \notin [0,2]. \end{cases}$$

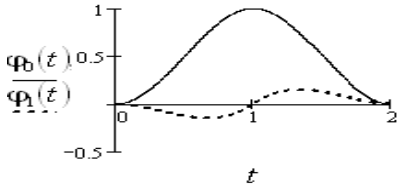


Рис. 1. Графики масштабирующих функций φ_0, φ_1

Тогда эрмитова бикубическая сплайн-поверхность может быть представлена как

$$\overline{S}^{L}(u,v)=\sum_{i=0}^{2^{L_1}}\sum_{j=0}^{2^{L_2}}\overline{C}_{i,j}^{L,0,0}N_{i,0}^{L_1}(u)N_{j,0}^{L_2}(v)+\sum_{i=0}^{2^{L_1}}\sum_{j=0}^{2^{L_2}}\overline{C}_{i,j}^{L,0,1}N_{i,0}^{L_1}(u)N_{j,1}^{L_2}(v)+$$
$$+\sum_{i=0}^{2^{L_1}}\sum_{j=0}^{2^{L_2}}\overline{C}_{i,j}^{L,1,0}N_{i,1}^{L_1}(u)N_{j,0}^{L_2}(v)+\sum_{i=0}^{2^{L_1}}\sum_{j=0}^{2^{L_2}}\overline{C}_{i,j}^{L,1,1}N_{i,1}^{L_1}(u)N_{j,1}^{L_2}(v), \quad (1)$$

где $N_{i,k}^{L_1}(u)$ и $N_{j,k}^{L_2}(v)$ являются базисными эрмитовыми сплайнами для направлений u и v соответственно. Здесь коэффициенты $\overline{C}_{i,j}^{L,k,l}=\gamma_{i,j}^{(k,l)}, k,l=0,1,$ являются соответствующими составляющими координат контрольных точек и направляющих косинусов касательной к поверхности в узлах сетки $\Delta^{L_1}\times\Delta^{L_2}$.

Вейвлет-пространство W^{L-1} определяется как ортогональное дополнение V^{L-1} до V^{L} по отношению к определенному скалярному произведению. В этом случае пространство V^{L} может быть представлено в виде прямой суммы V^{L-1} и W^{L-1}: $V^{L}=V^{L-1}\oplus W^{L-1}$. Для случая неограниченного интервала $(-\infty, \infty)$ одномерные базисные функции W^{L-1}, имеющие суперкомпактный носитель [0, 2], получены в [4] относительно скалярного произведения $\langle f,g\rangle=\int_{-\infty}^{\infty}f'(t)g'(t)dt$ (рис. 2):

$$\psi_0(t)=-2\phi_0(2t)+4\phi_0(2t-1)-2\phi_0(2t-2)-21\phi_1(2t)+21\phi_1(2t-2), \quad (2)$$

$$\psi_1(t)=\phi_0(2t)-\phi_0(2t-2)+9\phi_1(2t)+12\phi_1(2t-1)+9\phi_1(2t-2). \quad (3)$$

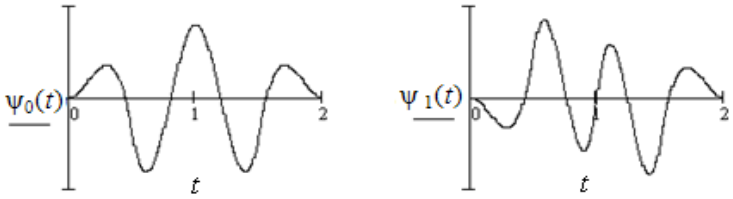
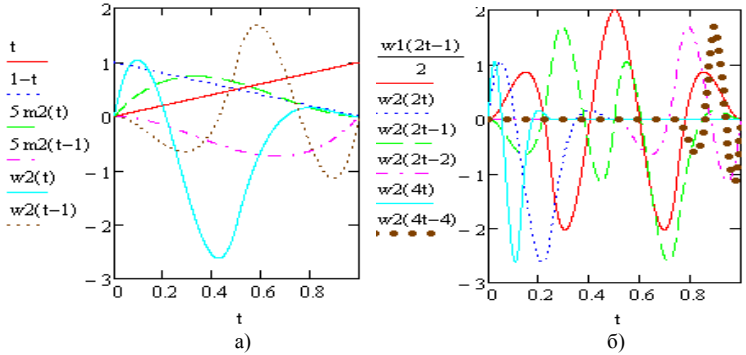


Рис. 2. Графики симметричного и несимметричного «материнских» вейвлетов $\psi_0(t), \psi_1(t)$

Часто ограничиваются рассмотрением периодического случая, когда совпадают первая и последняя контрольные точки по каждому направлению. Это соответствует аппроксимации замкнутых поверхностей. Для незамкнутых поверхностей с целью сохранения свойства полуортогональности по каждому частному направлению можно было бы вычесть из исходных координат уравнение билинейной смешанной поверхности, соединяющей частичные кубические эрмитовы сплайны, интерполирующие данные на каждой стороне. Тогда исправленные значения координат и частных производных по краям обнуляются, и после их подгонки, например, согласно методу наименьших квадратов, к полученным в результате подгонки коэффициентам бикубического эрмитового сплайна требуется добавить вычтенное ранее уравнение.

Результаты численных экспериментов

Мы рассматривали более простое решение, когда для построения тензорного произведения в одномерный базис кубических мультивейвлетов добавляются две линейные базисные функции по краям отрезка аппроксимации (рис. 3).



а) б)

Рис. 3. Графики базисных функций на первом (*а*) и втором (*б*) уровнях разложения по кубическим мультивейвлетам

На рис. 4 представлен график восполнения построенного бикубического сплайна на густую прямоугольную сетку, а также результат наложения исходных точек на полученную сплайн-поверхность. На рис. 5 представлен соответствующий график стандартизированных ошибок аппроксимации метода наименьших квадратов, позволяющий указать точки дискретно заданной двумерной поверхности, рекомендуемые для фильтрации.

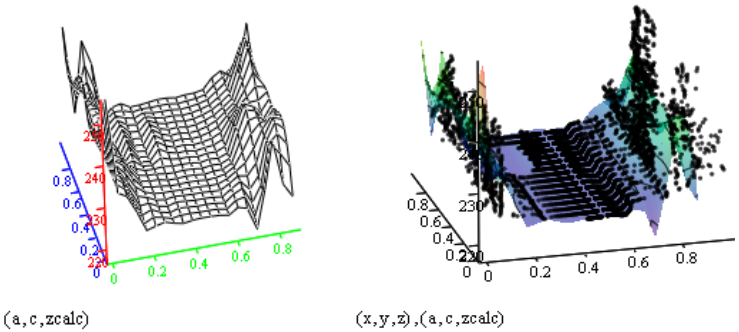


Рис. 4. График восполнения бикубического сплайна и результат наложения исходных точек на полученную поверхность

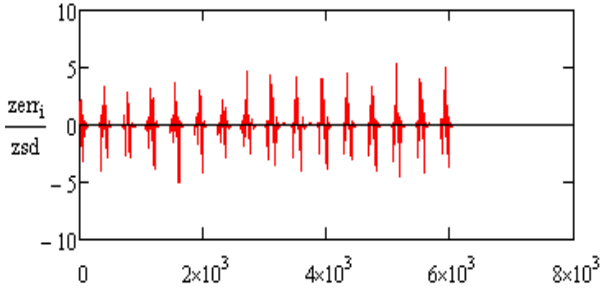


Рис. 5. График стандартизированных ошибок аппроксимации для бикубических вейвлетов

Литература

1. **Cohen A. Doubeshies, Vial P.** Wavelets on the interval and fast wavelet transforms // Appl. Comput. Harmonic Anal. – 1993. – Vol. 1. – P. 54–81.
2. **Столниц Э., ДеРоуз Т., Салезин Д.** Вейвлеты в компьютерной графике: Пер. с англ. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002. – 272 с.
3. **Dahmen W., Jia R,Q., Kunoth A.** Biorthogonal multiwavelets on the interval: Cubic Hermite splines // Constr. Approx. – 2000. – Vol. 16. – P. 221–259.
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. **Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л.** Методы сплайн-функций. – М.: Наука. Главная редакция физико-математической литературы, 1980. – 352 с.

Алгоритм с расщеплением вейвлет-преобразования эрмитовых сплайнов 5-й степени*

Б.М. Шумилов, Ш.М. Матанов
Томский государственный университет
Ошский государственный университет

Для эрмитовых сплайнов 5-й степени предложен новый тип «ленивых» вейвлетов со смещенным носителем. С использованием расщепления по четным и нечетным узлам получен алгоритм вейвлет-разложения в виде решения трех систем линейных уравнений, из которых одна система трехдиагональная со строгим диагональным преобладанием и две другие системы четырехдиагональные.

Введение

Вейвлетом называется короткая или быстро затухающая волновая функция (всплеск), множество сжатий и смещений которой порождает пространство измеримых функций на всей числовой оси [1]. С вычислительной точки зрения наиболее удобны для применения ортонормальные и биортогональные вейвлеты, для которых имеются явные локальные формулы вычисления коэффициентов разложения заданной функции. К недостаткам относится то, что их двойственные (за исключением вейвлетов Хаара) не имеют аналитического представления и графически похожи на фрактальные кривые. Полуортогональные сплайн-вейвлеты образуют иерархический базис в пространстве кусочно-полиномиальных гладких функций (сплайнов). Недостатки состоят в том, что они определены на достаточно большом носителе и для них не существует явных конечных формул, связывающих базисные функции пространства сплайнов на густой сетке, базисные функции на прореженной сетке и вейвлеты. Поэтому при вычислениях предлагалось обходиться приближенными соотношениями для главных коэффициентов разложения либо прибегать к решению системы линейных алгебраических уравнений с ленточной матрицей, для которой не гарантирована хорошая обусловленность.

В [2] уменьшение носителей достигалось построением мультивейвлетов, у которых с каждым узлом связано более одной базисной функции. В [3] для частного случая кубических эрмитовых мультивейвлетов с помощью метода неопределенных коэффициентов были

* Работа выполнена при поддержке РФФИ (проект № 11-01-90900 моб_снг_ст).

впервые получены конечные неявные соотношения разложения. После этого [4, 5] алгоритм вычисления сводится к решению независимо двух трехдиагональных систем уравнений со строгим диагональным преобладанием, что предпочтительно с точки зрения распараллеливания вычислений. В данной работе, используя эти идеи, мы докажем существование конечных неявных соотношений разложения для эрмитовых сплайнов 5-й степени и обоснуем эффективный алгоритм вейвлет-анализа на их основе.

Построение «ленивых» эрмитовых сплайн-вейвлетов 5-й степени на конечном отрезке

Пусть пространство V_L является пространством сплайнов степени 5 гладкости C^2 на отрезке $[a, b]$ с равномерной сеткой узлов $\Delta^L : u_i = a + (b-a)\, i / 2^L,\ i = 0, 1, \ldots, 2^L,\ L \geq 0$, и базисными функциями $N^L_{i,k}(v) = \varphi_k(v - i)$, $k = 0, 1, 2\ \forall i$, где $v = 2^L (u-a)/(b-a) + 1$, с центрами в целых числах, порожденными сжатиями и сдвигами трех функций вида [6] (рис. 1):

$$\begin{bmatrix} \varphi_0(t) \\ \varphi_1(t) \\ \varphi_2(t) \end{bmatrix} = \left\{ \begin{bmatrix} t^3(6t^2 - 15t + 10) \\ -t^3(3t^2 - 7t + 4) \\ \dfrac{t^3}{2}(t^2 - 2t + 1) \end{bmatrix}, 0 \leq t \leq 1; \begin{bmatrix} (2-t)^3(6t^2 - 9t + 4) \\ (2-t)^3(3t^2 - 5t + 2) \\ \dfrac{(2-t)^3}{2}(t^2 - 2t + 1) \end{bmatrix}, 1 \leq t \leq 2 \right\},$$

$$\varphi_k(t) = 0,\ k = 0, 1, 2,\ t \notin [0, 2].$$

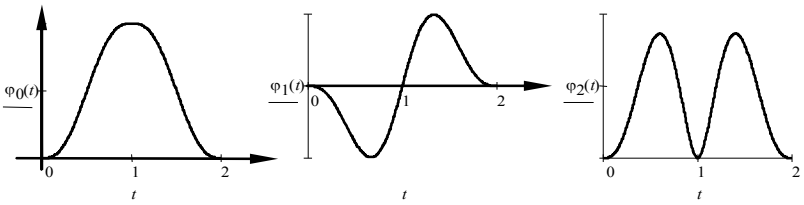


Рис. 1. Графики функций $\varphi_0(t)$, $\varphi_1(t)$, $\varphi_2(t)$

На любой сетке Δ^L, $L \geq 0$, эрмитов сплайн 5-й степени может быть представлен как

$$S^L(u) = \sum_{k=0}^{2} \sum_{i=0}^{2^L} C_i^{L,k} N_{i,k}^L(u), \quad a \leq u \leq b. \tag{1}$$

Если сетка Δ^{L-1} получена из Δ^{L} посредством удаления каждого второго узла, то соответствующее пространство V_{L-1} с базисными функциями $N_{i,k}^{L-1}$, в два раза большими по ширине с центрами в четных целых числах, вложено в V_L. Пространство вейвлетов W_{L-1} определяется как дополнение V_{L-1} до V_L, так что любая функция в V_L может быть записана в виде суммы некоторой функции в V_{L-1} и некоторой функции в W_{L-1}.

В [1] для получения базисных функций в W_{L-1} использовались функции $N^{L}_{i,k}$ в V_L с центрами в нечетных целых числах («ленивые» вейвлеты). Мы предлагаем использовать в качестве вейвлетов для W_{L-1} функции $N^{L}_{i,k}$ в V_L с центрами в четных целых числах. Поскольку W_{L-1} должно являться дополнением V_{L-1} в V_L, размерности этих пространств должны удовлетворять соотношению

$$\text{Dim}\,(V_L) = \text{Dim}\,(V_{L-1}) + \text{Dim}\,(W_{L-1}).$$

Для выполнения этого условия далее будет исследован вариант, когда из исходных координат вычитается уравнение прямой, соединяющей первую и последнюю вершины при дополнительном условии, что вторая производная в последней точке обращается в нуль.

Будем обозначать базисные сплайн-функции и коэффициенты эрмитового сплайна 5-й степени с отсутствующими элементами по концам отрезка аппроксимации как φ_0^{L} и C_0^{L}. Аналогично обозначим базисные вейвлет-функции как $M^{L}_{i,k} = \varphi_k(v - 2i)$, $k = 0, 1, 2$, $i = 0, 1, \ldots, 2^L$, и запишем их в виде матрицы-строки: $\psi_0^L = \left[M^L_{0,1}, M^L_{0,2}, M^L_{1,0}, M^L_{1,1}, M^L_{1,2}, ..., M^L_{2^L-1,0}, M^L_{2^L-1,1}, M^L_{2^L-1,2}, M^L_{2^L,1} \right]$. Соответствующие вейвлет-коэффициенты будем собирать в вектор: $D_0^L = \left[D_0^{L,1}, D_0^{L,2}, D_1^{L,0}, D_1^{L,1}, D_1^{L,2}, ..., D_{2^L-1}^{L,0}, D_{2^L-1}^{L,1}, D_{2^L-1}^{L,2}, D_{2^L}^{L,1} \right]^T$. В результате вейвлет-преобразование может быть записано как [1]

$$C_0^L = \left[P_0^L \,|\, Q_0^L \right] \left[\frac{C_0^{L-1}}{D_0^{L-1}} \right], \qquad (2)$$

где блочная матрица $[P_0^L|Q_0^L]$ составлена из коэффициентов двухмасштабных соотношений [5] и единиц.

Метод расщепления сплайн-вейвлет-разложения

Обратный процесс разбиения коэффициентов C^L на более грубую версию C^{L-1} и уточняющие коэффициенты D^{L-1} состоит в решении

системы линейных уравнений (2). При этом для распараллеливания вычислений систему (2) целесообразно расщепить по четным и нечетным узлам [4, 5]. Имеет место следующая

Теорема 1. *Пусть* значения *сплайн-коэффициентов* $C_i^{L,2}$, $C_i^{L,0}$ *и* $C_i^{L,1}$ *в нечетных узлах пересчитаны из последовательного решения трех систем линейных уравнений соответственно вида*

$$\begin{bmatrix} \frac{99}{2} & -4 & \frac{1}{2} & & \\ \frac{1}{2} & -4 & \frac{111}{2} & \ddots & \\ & & \frac{111}{2} & \ddots & \\ & & \frac{1}{2} & \ddots & -4 \\ & & & \ddots & -4 \end{bmatrix} \cdot \begin{bmatrix} C_1^{L,2} \\ C_3^{L,2} \\ \vdots \\ C_{2^L-1}^{L,2} \end{bmatrix} := \begin{bmatrix} C_1^{L,2} \\ C_3^{L,2} \\ \vdots \\ C_{2^L-1}^{L,2} \end{bmatrix},$$

$$\begin{bmatrix} \frac{53}{4} & 8 & \frac{1}{8} & & \\ \frac{1}{8} & 8 & \frac{111}{8} & \ddots & \\ & & \frac{111}{8} & \ddots & \\ & & \frac{1}{8} & \ddots & 8 \\ & & & \ddots & 8 \end{bmatrix} \cdot \begin{bmatrix} C_1^{L,0} \\ C_3^{L,0} \\ \vdots \\ C_{2^L-1}^{L,0} \end{bmatrix} := \begin{bmatrix} C_1^{L,0} - C_1^{L,2} \\ C_3^{L,0} - C_1^{L,2} \\ C_5^{L,0} \\ \vdots \\ C_{2^L-3}^{L,0} - C_{2^L-1}^{L,2} \\ C_{2^L-1}^{L,0} - C_{2^L-1}^{L,2} \end{bmatrix},$$

$$\begin{bmatrix} \frac{111}{16} & \frac{1}{16} & & \\ \frac{1}{16} & \frac{55}{8} & \ddots & \\ & \frac{1}{16} & \ddots & \frac{1}{16} \\ & & \ddots & \frac{111}{16} \end{bmatrix} \cdot \begin{bmatrix} C_1^{L,1} \\ C_3^{L,1} \\ \vdots \\ C_{2^L-1}^{L,1} \end{bmatrix} := \begin{bmatrix} C_1^{L,1} + 15C_1^{L,0} + C_1^{L,2} \\ C_3^{L,1} - 15C_1^{L,0} - C_1^{L,2} \\ C_5^{L,1} \\ \vdots \\ C_{2^L-3}^{L,1} + 15C_{2^L-1}^{L,0} + C_{2^L-1}^{L,2} \\ C_{2^L-1}^{L,1} - 15C_{2^L-1}^{L,0} - C_{2^L-1}^{L,2} \end{bmatrix}.$$

Тогда вектор сплайн-коэффициентов на прореженной сетке Δ^{L-1} *представляет собой результат умножения матрицы*

$$\begin{bmatrix}
-150 & -16 & -168 & & & & & & & & & & \\
-1260 & 0 & -1960 & & & & & & & & & & \\
4 & -1 & 2 & 16 & 1 & 0 & 4 & 0 & 2 & & & & \\
30 & -8 & 14 & 0 & -8 & 0 & -30 & 0 & -14 & \ddots & & & \\
180 & -48 & 76 & 0 & 48 & 64 & 180 & 0 & 76 & \ddots & & & \\
 & & & & -1 & 0 & 20 & 1 & 24 & \ddots & & & \\
 & & & & -8 & 0 & 0 & -8 & 0 & \ddots & & & \\
 & & & & -48 & 0 & -360 & 48 & -1048 & \ddots & & & \\
 & & & & & & 4 & -1 & 2 & \ddots & 16 & 1 & 0 \\
 & & & & & & 30 & -8 & 14 & \ddots & 0 & -8 & 0 \\
 & & & & & & 180 & -48 & 76 & \ddots & 0 & 48 & 64 \\
 & & & & & & & & & \ddots & 0 & -16 & 0
\end{bmatrix}$$

на вектор, $\left[C_1^{L,0}, C_1^{L,1}, C_1^{L,2}, ..., C_{2^L-1}^{L,2}\right]^T$, *сплайн-коэффициентов в нечетных узлах густой сетки* Δ^L, *тогда как вектор вейвлет-коэффициентов равен такому же произведению с матрицей*

$$\begin{bmatrix}
75 & 8 & 84 & & & & & & & & & & \\
315 & 0 & 490 & & & & & & & & & & \\
-4 & 1 & -2 & -16 & -1 & 0 & -4 & 0 & -2 & & & & \\
-15 & 4 & -7 & 0 & 4 & 0 & 15 & 0 & 7 & \ddots & & & \\
-45 & 12 & -19 & 0 & -12 & -16 & -45 & 0 & -19 & \ddots & & & \\
 & & & & 1 & 0 & -20 & -1 & -24 & \ddots & & & \\
 & & & & 4 & 0 & 0 & 4 & 0 & \ddots & & & \\
 & & & & 12 & 0 & 90 & -12 & 262 & \ddots & & & \\
 & & & & & & -4 & 1 & -2 & \ddots & -16 & -1 & 0 \\
 & & & & & & -15 & 4 & -7 & \ddots & 0 & 4 & 0 \\
 & & & & & & -45 & 12 & -19 & \ddots & 0 & -12 & -16 \\
 & & & & & & & & & \ddots & 0 & 8 & 0
\end{bmatrix}$$

при условии добавления к результату вектора $\left[C_0^{L,1}, C_0^{L,2}, C_2^{L,0}, C_2^{L,1}, C_2^{L,2}, ..., C_{2^L}^{L,1}\right]^T$ *сплайн-коэффициентов в четных узлах густой сетки.*

Пример. Рассмотрим в качестве тестовой функции функцию Хартена [7]:

$$f(x)=\begin{cases}\frac{1}{2}\sin(3\pi x), & x\le\frac{1}{3},\\ \left|\sin(4\pi x)\right|, & \frac{1}{3}<x\le\frac{2}{3},\\ -\frac{1}{2}\sin(3\pi x), & x>\frac{2}{3}.\end{cases}$$

Это кусочно-гладкая функция, имеющая разрывы первого рода в точках $x=1/3$ и 2/3 и угол (разрыв первой производной) в точке $x=1/2$. Ее первая и вторая производные – также кусочно-гладкие функции.

На рис. 2 представлены результаты реконструкции коэффициентов эрмитового сплайна 5-й степени $S^5(x)$ при условии обнуления незначимых вейвлет-коэффициентов $D_0{}^3(12)$, $D_0{}^3(13)$ и $D_0{}^4(18)$, $D_0{}^4(22)$, $D_0{}^4(28)$, $D_0{}^4(30)$. Здесь сплошной линией обозначаются исходная функция и ее производные. Попытка обнуления следующих по абсолютной величине вейвлет-коэффициентов $D_0{}^4(2)$, $D_0{}^4(6)$, $D_0{}^4(9)$, $D_0{}^4(39)$, $D_0{}^4(42)$ приводит к некоторому ухудшению качества воспроизведения первой производной и заметному искажению самой функции в окрестности третьего узла (последняя часть рисунка). Поведение второй производной при этом совершенно не изменяется.

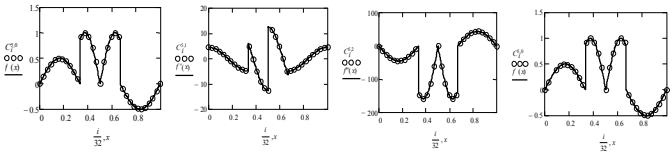


Рис. 2. Графики вейвлет-реконструкции коэффициентов эрмитового сплайна 5-й степени

Заключение

Можно предложить параллельную реализацию представленного в статье алгоритма вейвлет-преобразования, в которой три прямых хода прогонки выполняются независимо, а три обратных хода выполняются с максимальным запаздыванием на два такта. Это предоставляет возможность эффективного использования эрмитовых сплайн-вейвлетов 5-й степени на современных процессорах с многоядерной архитектурой.

Литература

1. **Столниц Э., ДеРоуз Т., Салезин Д.** Вейвлеты в компьютерной графике: Пер. с англ. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2002. – 272 с.
2. **Strela V. Multiwavelets**. Theory and Applications. – Massachusetts Institute of Technology, 1996. – 99 p. Submitted to the Department of Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mathematics.
3. **Шумилов Б.М., Эшаров Э.А.** Построение эрмитовых сплайн-вейвлетов // Вестник Томского государственного университета. Сер. Математика. Кибернетика. Информатика. Приложения. – 2006. – № 19. – С. 260–266.
4. **Шумилов Б.М.** Алгоритм с расщеплением вейвлет-преобразования эрмитовых кубических сплайнов // Вестник Томского государственного университета. Сер. Математика. Механика. – 2010. – № 4(12). – С. 45–55.
5. **Шумилов Б.М.** «Ленивые» вейвлеты эрмитовых кубических сплайнов и алгоритм с расщеплением // Вестник Томского государственного университета. Сер. Управление, вычислительная техника и информатика. – 2011. – № 1(14). – С. 64–72.
6. **Стечкин С.Б., Субботин Ю.Н.** Сплайны в вычислительной математике. – М.: Наука, 1976. – 248 с.
7. **Aràndiga F., Baeza A., Donat R.** Discrete multiresolution based on hermite interpolation: computing derivatives // Communications in Nonlinear Science and Numerical Simulation. – 2004. – Vol. 9. – P. 263–273.

Оптимизация исполнения фрагментированных программ на основе профилирования*

В.А. Перепелкин
Институт вычислительной математики
и математической геофизики СО РАН, Новосибирск

Рассматривается задача оптимизации исполнения параллельной программы на основе профилирования в системе фрагментированного программирования LuNA. Предлагается алгоритм анализа профиля и конструирования более эффективного распределения ресурсов. Приводятся результаты тестирования.

При реализации больших численных моделей на суперкомпьютерах необходимо обеспечивать высокую эффективность прикладной параллельной программы (тут и далее эффективность понимается в смысле уменьшения времени выполнения параллельной программы на вычислителе заданной конфигурации). Достижение высокой эффективности прикладной параллельной программы является сложной задачей системного параллельного программирования и требует существенных усилий от программиста, в общем-то, не связанных напрямую с его прикладной задачей, а лишь с её реализацией на конкретном классе вычислителей [2]. В дальнейшем с развитием параллельных вычислителей достижение высокой эффективности прикладных программ будет только усложняться. В связи с этим большое значение имеют системы параллельного программирования, автоматизирующие ту часть работы прикладного программиста, которая связана с системным параллельным программированием. Один из важных подходов в автоматической оптимизации исполнения параллельных программ – это *профилирование* – форма динамического анализа программы, при которой фиксируются некоторые важные особенности поведения программы, такие как продолжительность выполнения операций и коммуникаций, дисбаланс загрузки процессоров и т.п. Собранная статистика называется *профилем программы* и используется как источник дополнительной информации для последующей оптимизации программы.

Автоматическая оптимизация исполнения параллельных программ на основе профилирования является привлекательным подходом, т.к. с точки зрения пользователя означает, что каждый последующий запуск программы будет работать быстрее, чем предыдущий, приближаясь к

* Проект поддержан грантом РФФИ №10-07-00454а.

некоторому субоптимальному значению по времени. Однако профилирование имеет и ряд недостатков, которые будут «унаследованы» и в настоящей работе. Во-первых, профиль программы всегда привязан к конфигурации вычислителя. Это означает, что при последующем запуске прикладной параллельной программы на другом вычислителе профиль становится непригодным. Во-вторых, профиль программы привязан к *входным данным* программы, т.е. при запуске той же программы на вычислителе той же конфигурации но на другом наборе входных данных временны́е характеристики её исполнения могут отличаться. Как следствие, оптимизация на основе профилирования может понизить эффективность программы. Тем не менее, несмотря на описанные недостатки, профилирование является весьма востребованным, т.к. в численном моделировании вполне обычной является ситуация, когда один и тот же численный эксперимент проводится многократно с различными параметрами. При э́том изменение параметров далеко не всегда сказывается на временны́х характеристиках исполнения программы. В некоторых программах (например, операциях над плотными матрицами) конкретные значения переменных вообще практически не влияют на время и ход работы программы. В подобных случаях профилирование является востребованным.

Работа посвящена попытке реализации подхода автоматической оптимизации исполнения параллельных программ на основе профилирования для системы фрагментированного программирования LuNA [3], разрабатываемой в ИВМиМГ СО РАН (г. Новосибирск). Эта система ориентирована на реализацию больших численных моделей для суперкомпьютеров. Одной из особенностей этой системы является представление прикладного алгоритма в виде множеств фрагментов данных и вычислений, представляющих соответственно переменные и операции прикладного алгоритма [1]. Такая фрагментированная структура программы сохраняется и во время исполнения, благодаря чему возможно профилирование программ на уровне фрагментов безотносительно их содержимого (т.е. операций и значений переменных).

Необходимые определения

Здесь приводятся упрощенные формальные определения фрагментированной программы, рекомендаций и профиля. Упрощения сделаны для простоты изложения и не затрагивают существенных аспектов с рассматриваемой точки зрения. Более подробный формализм представлен, например, в [4].

Фрагментированная программа (ФП) – это кортеж $\langle CF, \rho \rangle$, где CF – это потенциально бесконечное множество фрагментов вычислений (ФВ), представляющих агрегированные операции прикладного алгоритма. Каждый ФВ реализуется вызовом некоторой процедуры над конкретным набором значений входных фрагментов данных. Отношение $\rho \subseteq CF \times CF$ выражает информационные зависимости между ФВ. Исполнение ФП состоит в запуске всех ФВ в порядке, не противоречащем ρ, т.е. если $f_1, f_2 \in CF$ и $f_1 \; \rho \; f_2$, то запуск ФВ f_2 можно осуществлять только после того, как завершился запуск ФВ f_1.

Рекомендации – это отображение $L{:}CF \rightarrow PE$, где PE – это множество узлов мультикомпьютера, на которых происходит запуск ФП. Если для некоторого $f \in CF \; L(f)=p$, то это означает, что ФВ f будет запущен на узле p.

Профиль – это пара функций $\langle S,E \rangle$, определённых на множестве CF, которые обозначают соответственно время начала и окончания срабатывания произвольного ФВ.

Исполнение ФП происходит с учётом рекомендаций. Именно исполнительная система, которая исполняет ФП, старается запустить ФВ на тех вычислительных узлах, которые заданы в рекомендациях. Распределение ФВ по узлам существенно влияет на эффективность исполнения ФП, но при этом никак не сказывается на содержательной части работы программы, т.е. не влияет на значения, полученные в ходе вычислений. Таким образом, задавая те или иные рекомендации, возможно влиять на эффективность исполнения программы.

Формально задача оптимизации исполнения ФП на основе профилирования ставится как задача построения нового отображения L' на основе $\langle CF, \rho, L, S, E \rangle$. Неформально можно сказать, что исполнение программы с отображением L' должно занимать меньше времени, чем с отображением L.

Описание алгоритма

Идея алгоритма состоит в том, что профиль анализируется на предмет наличия дисбаланса загрузки вычислительных узлов в те или иные моменты времени. Под дисбалансом понимается ситуация, когда один узел занят исполнением более чем одного ФВ, в то время как другой узел простаивает без работы. При обнаружении дисбаланса загрузки ФВ с перегруженного узла назначаются на другие, свободные вычислительные узлы.

Таким образом, алгоритм направлен на то, чтобы по факту возникновения дисбаланса скорректировать рекомендации, чтобы этот

дисбаланс исчез или стал меньше при следующем запуске. Для положительного эффекта от алгоритма необходимо, чтобы на перегруженных узлах было какое-то количество независимых ФВ. Если же ФВ связаны информационными зависимостями, то их параллельное исполнение невозможно и, следовательно, назначение их на разные узлы не ускорит выполнение программы, а скорее всего несколько замедлит из-за дополнительной нагрузки на коммуникации.

Недостатком алгоритма является отсутствие анализа информационных зависимостей между ФВ и соответственно нагрузок на коммуникационную подсистему. Как следствие, для успешной оптимизации необходимо, чтобы накладные расходы на коммуникационную подсистему были меньше, чем выигрыш, полученный за счёт перераспределения вычислительной нагрузки.

Тестирование

Работа алгоритма была протестирована на задаче умножения плотных матриц, реализованной в системе фрагментированного программирования LuNA. Особенностью задачи является относительно большое количество операций на единицу данных, что делает эту задачу удобной для предложенного алгоритма оптимизации исполнения на основе профилирования. Тестирование проводилось на кластере НКС Сибирского суперкомпьютерного центра (г. Новосибирск). Для вычислительных кластеров характерна высокая производительность сетевой подсистемы по сравнению с другими типами вычислителей. В частности, для такой задачи, как умножение плотных матриц, влияние коммуникаций мало (порядка 10% для матриц размера 6000×6000), что также является выгодным обстоятельством для предложенного алгоритма.

Для сравнения была выбрана реализация той же задачи в MPI, оптимизированная вручную с использованием алгоритма блочного умножения матриц. Схема распараллеливания была взята из систолических алгоритмов умножения матриц для двумерной решетки вычислительных узлов. Во ФП в качестве фрагментов кода использовалась та же процедура блочного умножения матриц. Таким образом, ручная реализация являлась теоретически достижимым пределом по времени работы для ФП. Отставание по времени работы ФП связано с двумя причинами. Во-первых, это неоптимальность в распределении ресурсов (предмет исследования) и, во-вторых, накладные расходы на работу самой исполнительной системы. На рис. 1 изображена зависимость времени работы программы для последовательности запусков. Каждый

следующий запуск был оптимизирован описанным алгоритмом на основе предыдущих запусков. Начальные рекомендации (на первом запуске) выбирались различными в очень широком диапазоне, вплоть до случайного распределения или назначения всех ФВ на один узел. Но каждый раз наблюдалась одна и та же картина, приведенная на рис. 1.

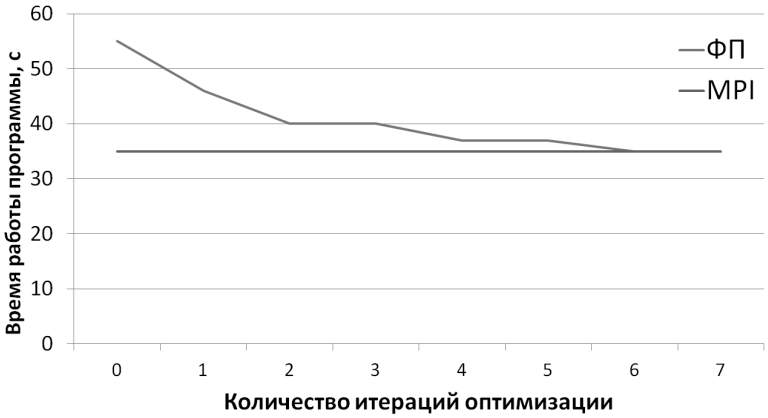


Рис. 1. Результаты тестирования

Как видно из графика, время работы программы уменьшается с каждым следующим запуском, доходя практически до времени работы ручной реализации в MPI. Этот результат означает, что для этой задачи одного только предложенного алгоритма оказалось достаточно, чтобы автоматически сконструировать распределение ресурсов, по качеству не уступающее ручной реализации. Разумеется, не для каждой задачи можно ожидать такого результата. Во-первых, в задаче существенную часть вычислений должны составлять массовые вычисления (т.е. однотипные независимые операции). Во-вторых, определяющими для времени работы программы («узким местом») должны быть вычисления, а не коммуникации. Задача умножения матриц оказалась подходящей по обоим этим критериям. Для успешной работы на других классах задач подход оптимизации исполнения ФП должен быть усовершенствован с учетом вышеописанных проблем.

Заключение

Рассмотрен подход автоматической оптимизации исполнения параллельных программ на основе профилирования в системе фрагмен-

тированного программирования LuNA. Предложен алгоритм конструирования рекомендаций на основе профиля исполнения ФП. Предложенный алгоритм реализован и протестирован на задаче умножения плотных матриц. Показана целесообразность использования рассматриваемого подхода. В дальнейшем планируется усовершенствование алгоритма автоматического конструирования рекомендаций на основе профиля исполнения ФП.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 127 с.
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, № 6. – P. 755–765.
3. **Malyshkin V.E., Perepelkin V.A.** LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem //Proceedings of the 11th Conference on Parallel Computing Technologis, 2011, LNCS 6873. – N.Y.: Springer. – P. 53–61.
4. **Malyshkin V.E., Perepelkin V.A.** Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system //The Journal of Supercomputing. – N.Y.: Springer, 2011. – P. 1–14. DOI: 10.1007/s11227-011-0649-6.

Система управления распределёнными структурами данных

Е.Г. Стадник
Новосибирский государственный технический университет

Рассмотрены основные проблемы, возникающие при работе с распределёнными данными, а также их возможные решения. Представлена библиотека для работы с распределёнными данными. Особое внимание уделено работе с асинхронной передачей данных.

Введение

Многие задачи численных методов связаны с обработкой большого объема данных. Часто задачи такого типа требуется решать с использованием мультикомпьютеров. При этом большие массивы разрезаются на части и части обрабатываются с выполнением коммуникаций для удовлетворения зависимостей по данным. Такой подход требует существенных усилий для разработки и реализации алгоритмов организации распределенной обработки данных.

Целью проекта является разработка библиотеки DDS (Distributed Data System), реализующей распределенные структуры данных.

Наиболее распространённой структурой данных для численных методов является n-мерный массив. Для распределённой реализации такой структуры можно выделить следующие требования:

- Должна сохраняться регулярная структура. Это позволит обрабатывать данные с помощью циклов так же, как и в исходном массиве.
- Должны удовлетворяться все зависимости по данным на границах разреза посредством коммуникаций.
- Коммуникации должны выполняться на фоне вычислений.
- Должна быть предусмотрена возможность выполнения балансировки вычислительной нагрузки.

Чтобы система удовлетворяла первому требованию, необходимо использовать блоки данных той же размерности и структуры, что и исходный массив. Под сохранением структуры понимается аналогичный способ размещения данных в памяти (например, построчное хранение массивов в Си). Второе требование предполагает хранение границ соседних блоков, что позволит уменьшить количество пересылок данных. Третье требование подразумевает асинхронную работу потоков пересылки данных и вычислений. Это можно обеспечить, размещая на одном узле несколько блоков: одни участвуют в коммуникаци-

ях, над другими производятся вычисления. Такие блоки будем называть фрагментами данных. Для удовлетворения четвертого требования нужно либо уметь перестраивать блоки в ходе вычислений, либо иметь на каждом узле множество блоков и переносить их для балансировки.

Анализ существующих систем

Наиболее известные системы для автоматизации распределения многомерных массивов – HPF университета William Marsh Rice (Хьюстон, Техас, США) и DVM Института прикладной математики им. М.В. Келдыша РАН. Системы имеют в основе сходные принципы распределения данных. Рассмотрим систему DVM подробнее. Инструмент представляет собой расширение стандартного языка Си или FORTRAN средствами спецификации параллельного выполнения программы:

- распределение элементов массива между процессорами;
- распределение витков цикла между процессорами;
- спецификация параллельно выполняющихся секций программы (параллельных задач) и отображение их на процессоры;
- организация эффективного доступа к удаленным (расположенным на других процессорах) данным;
- организация эффективного выполнения редукционных операций – глобальных операций с расположенными на различных процессорах данными (таких, как их суммирование или нахождение их минимального значения).

Достоинства:

- Повышение скорости разработки параллельных программ, реализующих численные методы с регулярными структурами данных и вычислений.
- Возможность контролировать процесс распараллеливания, добиваясь максимальной эффективности программ.
- Имеется встроенный отладчик и профайлер.

Недостатки:

- Отсутствие средств для автоматизации динамической балансировки.
- Элементы массивов не могут иметь произвольный тип, только стандартные int, long, float, double.
- Не обеспечивается правильная обработка указателей на элементы массивов при перераспределении данных.

В целом система DVM представляет собой удобный инструмент, позволяющий легко получить параллельную программу из последова-

тельной. Однако для того чтобы параллельная реализация была эффективной, на исходную программу должны быть наложены существенные ограничения. Во-первых, итерации при работе с распределёнными данными должны быть независимы. Во-вторых, время выполнения итераций должно быть примерно одинаково. Система даёт хороший результат на задачах, не требующих динамических свойств параллельной программы. Перераспределение вычислений в DVM означает глобальные действия по перераспределению всего массива, что может быть неэффективно при использовании большого количества процессоров и когда часто возникает потребность в балансировке.

Описание библиотеки DDS

Последовательность действий пользователя при работе с библиотекой выглядит следующим образом: определяется переменная типа «распределённый массив» и для неё указываются требуемые параметры распределения. На основании полученных данных система формирует локальные фрагменты на каждом вычислительном узле и передаёт управление пользователю. Далее пользователь получает возможность обращаться к локальным фрагментам и обрабатывать содержащиеся в них данные. Фрагменты данных, расположенные на одном узле, представляются пользователю как контейнер фрагментов. Доступ к фрагментам возможен посредством итератора. По окончании обработки фрагмента пользователь уведомляет об этом систему и информация о граничных элементах автоматически обновляется по мере готовности соседних фрагментов. Обработка данных фрагмента реализуется пользователем в основном потоке, а приём данных с других узлов автоматически осуществляется в параллельном потоке, что позволяет снизить временные затраты на синхронизацию за счёт передачи данных в фоновом режиме.

Интерфейс библиотеки:

Функции инициализации и завершения работы:

```
//инициализация библиотеки
int DDS_Init(int *argc, char ***argv);
//запуск потока для асинхронного приёма данных
int DDS_Begin();
//завершение работы с библиотекой
int DDS_Finalize();
```

Функции для работы с контейнером фрагментов (класс DDS_Array):

```
//задаёт размеры и количество фрагментов
int set_all_size(int _Nx, int _Ny, int _kx, int _ky);
```

//задаёт идентификатор переменной и тип распределения фрагментов

int set_parameters(int _id, int _distrib_type);

//возвращает итератор на первый фрагмент

iterator begin();

//возвращает итератор на последний фрагмент

iterator end();

//возвращает итератор фрагмента с индексом i

iterator operator[] (unsigned i);

Функции для работы с фрагментом данных (класс DDS_Array_fragment):

//возвращает указатель на данные фрагмента

T* Get_Data();

//заменяет данные фрагмента

int Replace_Data(T* source, int delete_flag);

//возвращает координаты фрагмента

void Get_block_index(int *number_x, int *number_y);

//возвращает размеры фрагмента

void Get_block_dim(int *count_x, int *count_y);

//возвращает глобальные координаты элемента с индексом (0,0)

void Get_start_element_index(int *x, int *y);

//указывает системе, что фрагмент готов для обмена данными

int border_ready();

Библиотека написана на языке C++ с использованием средств MPI и POSIX THREAD. В структуре системы можно выделить 3 уровня:

1. **Транспортный**. На этом уровне формируются пакеты данных, которые отсылаются на другие вычислительные узлы средствами MPI. Также обеспечивается работа потока для асинхронного приёма данных. Для обмена данными между локальными фрагментами используются буферы.

2. **Организационный**. Этот уровень отвечает за разбиение пользовательских данных на фрагменты. Здесь хранится информация о расположении фрагментов на узлах, а также задаются связи, определяющие зависимости по данным и алгоритмы балансировки.

3. **Прикладной** уровень содержит интерфейс для доступа к данным, а также команды управления: задание параметров распределения, команды синхронизации и ручной балансировки.

Разбиение на уровни даёт широкие возможности для модификации библиотеки. Например, реализация новой структуры данных потребует внесения изменений только на 2-м уровне.

Тестирование

Для тестирования и оценки эффективности использовалась задача численного решения уравнения Пуассона явным методом на пятиточечном шаблоне. Результаты измерений вычислительных затрат представлены на графике (рис. 1).

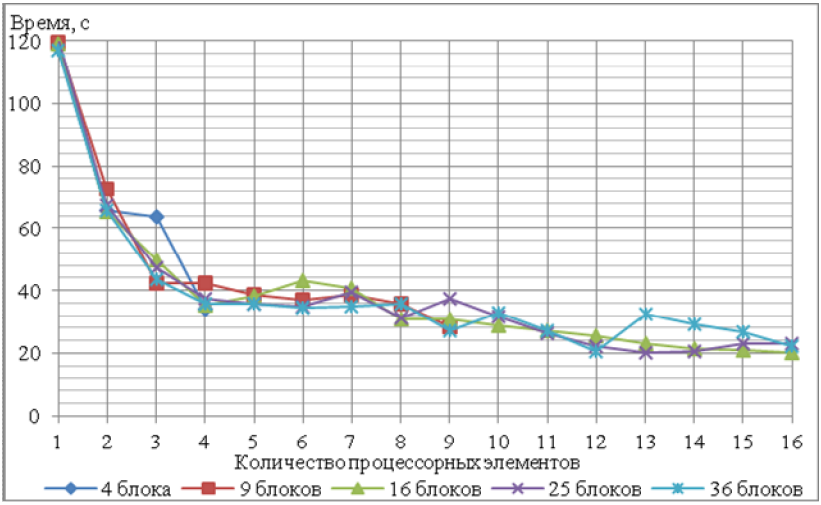


Рис. 1. Зависимость времени выполнения программы от числа процессорных элементов

В результате тестирования виден прирост скорости с увеличением числа вычислительных узлов, особенно это заметно при равномерной загрузке вычислителей: когда на процессорные элементы приходится равное количество блоков. Не удалось получить значительного ускорения за счёт увеличения количества фрагментов на узле. Также видна недостаточная масштабируемость системы. Это связано с накладными расходами на коммуникации между локальными фрагментами, а также отсутствием балансировщика.

Заключение

Разработана библиотека для работы с распределёнными данными. Реализована поддержка асинхронных коммуникаций. Библиотеку удобно применять для численных методов на сетках. Программисту предоставляются инструменты автоматического распределения и синхронизации данных. Его задача сводится к описанию алгоритма решения задачи на отдельных фрагментах. Асинхронные коммуникации

полностью реализуются системой и остаются невидимы для пользователя.

Система балансировки на данный момент находится в стадии разработки. За счёт перераспределения фрагментов ожидается повышение производительности, особенно на алгоритмах, где время фрагментов существенно варьируется и заранее неизвестно.

В перспективе планируются модификация и оптимизация алгоритмов передачи данных между фрагментами и реализация автоматических систем балансировки, также будет расширяться функционал для реализации более сложных численных методов, таких как метод частиц в ячейках.

Литература

1. **Бахтин В.А., Воронков А.В., Голубев А.С. и др.** Использование языка Fortran-DVM/OpenMP для разработки больших задач // Труды Всероссийской научной конференции «Научный сервис в сети Интернет: решение больших задач», сентябрь 2008 г., г. Новороссийск. – М.: Изд-во МГУ, 2008. – С. 185–191.
2. **Merlin J.H., Hey A.J.G.** An introduction to High Performance Fortran // Scientific Programming. – 1995. – Vol. 4. – P. 87–113.

Интегрированный программный комплекс для поддержки параллельных вычислений в рамках информационно-вычислительного портала

Н.Н. Окулов
Кемеровский государственный университет

Рассмотрен программный комплекс, предназначенный для повышения удобства и эффективности процесса разработки параллельных программ, а также позволяющий решать широкий спектр задач по использованию кластеров, включая обеспечение деятельности вычислительных центров удобной средой для управления учетными записями пользователей, программным обеспечением и вычислительными ресурсами.

Информационно-вычислительный портал и его подсистемы

Информационно-вычислительный портал (ИВП) предоставляет пользователям единую среду для осуществления широкого спектра работ по разработке и оптимизации параллельных программ для высокопроизводительных вычислений и проведения численного эксперимента на различных вычислительных ресурсах, независимо от их программно-аппаратного обеспечения, в удобном и доступном Web-интерфейсе. Также ИВП обеспечивает ряд дополнительных функций, упрощающих разработку параллельных приложений.

ИВП выступает платформой для интеграции нескольких информационных систем, образующих единый программный комплекс, позволяющий пользователю решать широкий перечень задач в сфере параллельных вычислений.

Основой портала является система удаленного доступа и управления распределенными вычислительными ресурсами, обеспечивающая выполнение вычислительных заданий в удаленном режиме, а также управление и хранение пользовательских объектов. Данная система сочетает в себе возможности удаленного доступа с графическим интерфейсом пользователя, пакетной обработки заданий и мониторинга состояния ресурсов.

Информационная система «Виртуальная лаборатория» разработана с целью организации виртуального лабораторного практикума с использованием высокопроизводительных вычислительных ресурсов в удаленном режиме. Данная система может быть использована в научных целях, а также в учебном процессе вузов в рамках курсов по высо-

копроизводительным вычислениям. «Виртуальная лаборатория» предполагает исследование эффективности параллельных программ, реализующих определенный численный алгоритм, и получение сведений для их дальнейшей оптимизации.

Одним из инструментов, существенно сокращающим цикл разработки параллельного приложения, является система отладки, производящая анализ на наличие локальных и глобальных ошибок в MPI-программах.

В портал интегрирована библиотека параллельных программ HydroParaLib, предназначенная для решения задач гидродинамики со свободными границами. Предусмотрен механизм подключения библиотек программ для других областей численных расчетов.

Системы мониторинга и биллинга предназначены для оперативного отслеживания состояния вычислительных ресурсов и учета использованного процессорного времени соответственно и носят служебный характер.

Структура системы УД и УРВР

Структура системы УД и УРВР включает три звена: клиент, сервер и менеджер кластера. В состав серверной части входят: сервер приложений, сервер баз данных и сервер управления кластерами (менеджер вычислительных ресурсов, МВР). Общая структура системы приведена на рис. 1.

Логически систему можно разбить на 2 взаимодействующие части:

– Система удаленного доступа к распределенным вычислительным ресурсам (СУД РВР), обеспечивающая пользовательский интерфейс для размещения своих объектов (файлов исходного кода, исполняемого кода и др.) и управления заданиями, и для обработки результатов. Данная подсистема включает в себя сервер БД под управлением СУБД Oracle, выступающий в качестве хранилища данных, и сервер приложений Tomcat, который осуществляет динамическую генерацию Web-страниц с использованием пакета KemsuWeb. Доступ пользователя к системе осуществляется посредством Web-браузера.

– Система управления распределенными вычислительными ресурсами (СУ РВР). Она отслеживает состояние кластеров, загружает на них новые задания и сохраняет результаты. В эту подсистему входят сервер БД, кластерные агенты, расположенные на главных узлах вычислительных кластеров, и сервер МВР, обеспечивающий передачу файлов и запросов на генерацию исполняемого кода между агентами и хранилищем данных, а также выполнение расчетов.

Взаимодействие менеджера вычислительных ресурсов с сервером БД осуществляется через прикладной интерфейс OCCI (Oracle C++ Call Interface).

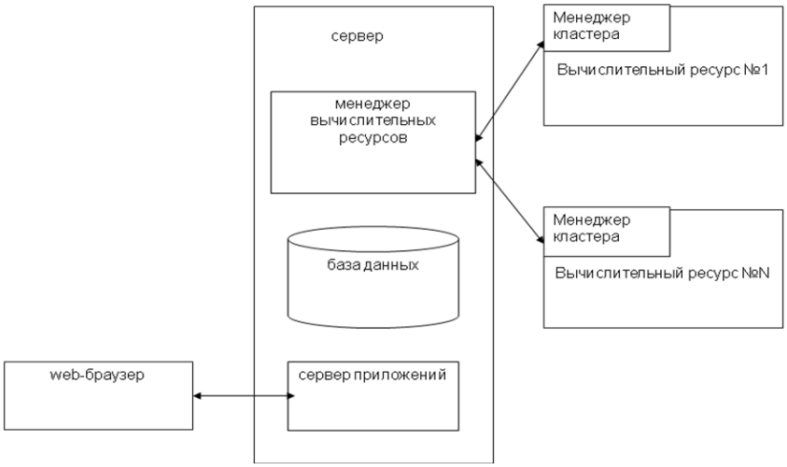


Рис. 1. Структура системы УД и УРВР

Менеджер кластера выполняет следующие функции:

– автоматическая компиляция заданий пользователей на кластере;
– автоматический запуск заданий (параллельных и последовательных) на кластере с последующим возвращением результатов пользователю;
– мониторинг состояния узлов кластера.

Все данные в систему передаются через NFS в виде управляющих файлов и файлов данных. В случае если менеджер кластера не запущен, то после его запуска произойдет считывание файлов команд и запуск заданий. При этом менеджер вычислительных ресурсов должен равномерно распределить нагрузку между кластерами, так как после запуска менеджера кластера он запускает все новые задания сразу. Файлы выходных данных сохраняются в NFS, затем передаются на сервер и далее клиенту по его запросу. Также возможен запрос промежуточных результатов. При этом серверу будут отправлены файлы, образовавшиеся в ходе работы программы, включая файл, в который перенаправляется стандартный вывод.

Система мониторинга состоит из агента сервера, агентов, расположенных на управляющих узлах кластера, и SNMP процессов, запу-

щенных на узлах кластера. Данные с узлов собираются агентом на хост-узле, вычисляются средние значения и отправляются агенту сервера. Далее они могут быть агрегированы и обработаны для последующего распределения нагрузки внутри кластера.

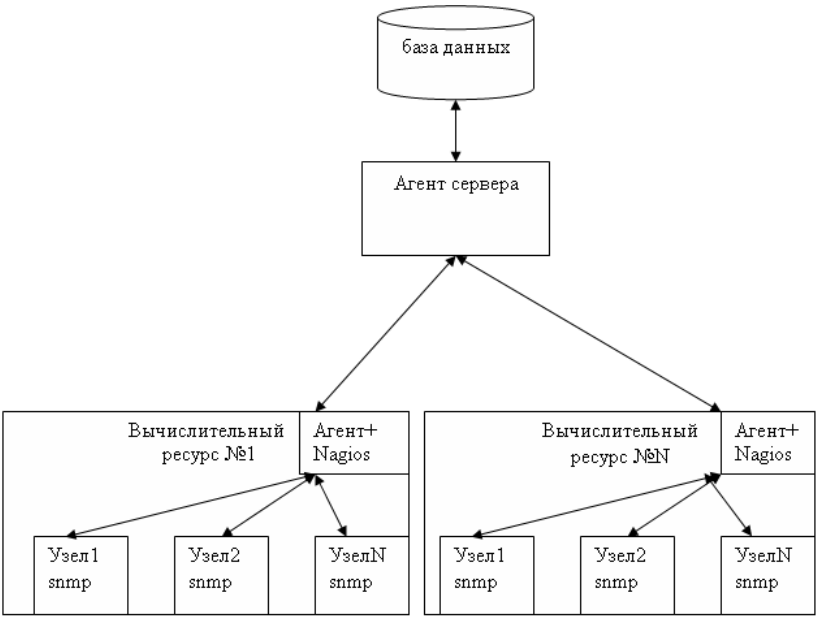


Рис. 2. Общая схема сбора статистики

В целях визуализации параметров мониторинга была выбрана система Nagios, способная предоставлять мгновенные значения параметров.

Система «Виртуальная лаборатория»

Система «Виртуальная лаборатория» предназначена для решения двух основных типов задач, определяющих возможность ее применения как в образовательной, так и в научной сферах:

1. Проведение серий вычислительных экспериментов с целью получения сведений для повышения эффективности параллельной реализации алгоритма, включая определение зависимости времени выполнения программы от задаваемых

пользователем параметров задачи; вычисление показателей ускорения и эффективности исследуемого параллельного кода, а также оценку и прогноз времени, необходимого для выполнения расчета.

2. Организация виртуального лабораторного практикума студентов на высокопроизводительных вычислительных ресурсах.

Основным пользовательским объектом в системе является лабораторная работа, которая содержит набор входных параметров работы и список проектов. Поддержка создания нескольких проектов в рамках лабораторной работы необходима для обеспечения возможности запуска лабораторной работы на вычислительных ресурсах с различным программно-аппаратным обеспечением (имеют значение вычислительная архитектура кластера, операционная система, используемые компиляторы). Значения входных параметров работы (например, количество итераций или размер матрицы) можно варьировать при запуске.

Для определения степени влияния параметров на время выполнения используются методы корреляционного анализа. В случае нескольких независимых входных параметров исследования решается уравнение множественной регрессии. Коэффициенты детерминации, полученные в результате применения данного метода, отражают степень зависимости времени выполнения программы от каждого из исследуемых параметров запуска.

Для оценки предполагаемого времени выполнения программы с заданными значениями параметров применяется интерполяция и экстраполяция на основе данных, полученных при предыдущих запусках.

Сформированные системой сведения пользователь может применить для повышения эффективности своей параллельной программы, а также предварительно оценить расчетное время выполнения своей программы. Благодаря возможности создания нескольких проектов для одной лабораторной работы и запуска на различных вычислительных ресурсах, пользователь также может сравнить между собой данные времени выполнения и выбрать оптимальную среду для запуска своей программы.

Литература

1. **Окулов Н.Н.** Разработка информационного портала параллельных вычислений для проведения научных и инженерных расчетов в режиме on-line / К.Е. Афанасьев, Н.Н. Окулов, С.В. Стуколов // Пя-

тая Сибирская конференция по параллельным и высокопроизводительным вычислениям. – Томск: Изд-во Том. ун-та, 2010. – С. 10 –14.
2. **Григорьева И.В.** Система удаленного доступа и управления распределенными вычислительными ресурсами / И.В. Григорьева, А.В. Демидов // Вычислительные технологии. – 2008. – Т. 13. – Специальный выпуск 5. – С. 28–32.

Язык фрагментированного программирования Freepal

Г.А. Щукин
Новосибирский государственный технический университет

Разрабатываемый язык фрагментированного программирования высокого уровня Freepal предназначен для создания параллельных фрагментированных программ. Программа на Freepal представляется в виде структурированных множеств фрагментов данных, кода и вычислений и не содержит привязки к конкретной архитектуре вычислительной системы или жестко заданного способа исполнения. За исполнение Freepal-программы отвечает runtime-система. Freepal-компилятор переводит инструкции программы в обращения к runtime-системам, при этом он строит управление на множестве фрагментов вычислений для корректного исполнения программы.

Введение

Развитие высокопроизводительных параллельных вычислительных систем требует наличия высокоуровневых средств создания эффективных параллельных программ.

Широко распространенные в настоящее время средства параллельного программирования, такие как MPI и OpenMP, являются инструментами низкого уровня, т.к. предоставляют только некоторый набор низкоуровневых примитивов для разработки параллельных программ.

Создание эффективной параллельной программы с помощью таких низкоуровневых средств требует от программиста существенных познаний во многих областях: от программирования до архитектуры параллельных вычислительных систем. Кроме того, такие динамические свойства программы, как балансировка загрузки или настройка на ресурсы, зачастую приходится проектировать и реализовывать заново для каждой новой задачи. Вследствие всего этого время и затраты на разработку и отладку программы существенно возрастают тем больше, чем сложнее является задача.

Наличие средств высокого уровня позволило бы решить эти и многие другие проблемы.

Фрагментированное программирование

Попытки создания высокоуровневых систем параллельного программирования предпринимаются уже давно. Большинство таких систем ос-

новано на идее расширения существующих последовательных языков дополнительными операторами/функциями параллельного программирования, предназначенными для порождения в программе ветвей для параллельного исполнения, синхронизации и т.д.

Другой подход используется в разрабатываемой технологии фрагментированного программирования [1]. В модели фрагментированного программирования программа представляется как множество объектов – фрагментов. Фрагменты подразделяются на три типа: фрагменты данных (данные, которые нужно обработать/вычислить), фрагменты кода (операции, которые можно применить к фрагментам данных) и фрагменты вычислений (указания, какие фрагменты кода применять к каким фрагментам данных и в каком допустимом порядке).

Фрагментированная программа является параллельной, если есть фрагменты вычислений, которые можно исполнить параллельно. В дальнейшем нас будут интересовать именно такие фрагментированные программы.

Особенностью технологии фрагментированного программирования является сохранение фрагментированной структуры программы во время ее исполнения, что дает возможность работать с фрагментами данных и вычислений как с отдельными объектами. Меняя размер, количество, распределение по вычислительным узлам и порядок исполнения фрагментов, возможно автоматическим образом обеспечить динамические свойства программы: коммуникации на фоне счета, настройку на ресурсы, динамическую балансировку нагрузки и т.д.

Исполнение фрагментированной программы на многопроцессорной вычислительной системе с разделяемой памятью является непростой задачей. Поэтому управление процессом исполнения программы возложено на runtime-систему [2, 3]. Использование runtime-системы позволяет иметь единый механизм исполнения для всех фрагментированных программ и обеспечить реализацию их динамических свойств.

Runtime-система содержит высокоуровневые функции для создания и исполнения фрагментов, их размещения и перемещения по вычислительным узлам и т.д. Тем не менее написание фрагментированных программ непосредственно с помощью интерфейса runtime-системы не обеспечивает нужный уровень абстракции и платформонезависимости. Желательно наличие специального языка высокого уровня как средства создания программ.

Язык Freepal

Язык Freepal является частью разрабатываемой системы фрагментированного программирования и предназначен для описания фрагментированных программ.

Основные особенности языка исходят непосредственно из модели фрагментированного программирования.

Во-первых, Freepal предназначен для задания множеств фрагментов данных, кода и вычислений (основных составляющих фрагментированных программ), их параметров и зависимостей между ними. Программа на Freepal не содержит жесткой привязки к определенной архитектуре вычислительных систем или строгой спецификации способа своего исполнения. Одна и та же Freepal-программа может быть исполнена на вычислительных машинах разного типа, с общей или разделяемой памятью, с разным числом процессоров и топологией связей узлов. Таким образом, на языке составляется только описание фрагментированного алгоритма, а его исполнение на конкретной вычислительной системе реализует скрытая от программиста runtime-система.

Во-вторых, Freepal является параллельным языком, т.к. ориентирован на описание множеств параллельно исполняющихся фрагментов вычислений. Тем не менее он использует некоторые черты последовательных императивных языков для упрощения написания параллельных программ, их разбора и отладки.

Фрагменты данных

В языке Freepal фрагмент данных играет роль переменной программы. Существуют два основных вида фрагментов данных: атомарные и структурированные.

Атомарный фрагмент данных – базовый неделимый объект программы. Для декларации атомарного фрагмента данных используется выражение **df** <Тип> <Имя>, где тип – один из базовых встроенных типов языка (для целых или вещественных чисел и т.д.). Если атомарный фрагмент данных – массив, для типа дополнительно указываются размеры массива.

В следующем примере a – целое число, b – массив из 10 вещественных чисел:

df int a;

df double[10] b.

Структурированный фрагмент данных – именованное множество фрагментов данных (атомарных или также структурированных). Приме-

ры: массив (множество однотипных индексированных элементов) или запись (множество элементов разного типа).

В следующем примере c – одномерный массив из 20 целых чисел, d – двумерный массив массивов вещественных чисел:

df int c[20];

df double[10] d[30, 50].

Главное отличие структурированных фрагментов данных состоит в том, что в ходе исполнения программы их элементы могут быть распределены по узлам мультикомпьютера и обрабатываться по отдельности.

Фрагменты кода и вычислений

Фрагмент кода в языке Freepal – функция, которая принимает входные фрагменты данных и вырабатывает из них выходные фрагменты данных. Фрагмент вычислений – вызов фрагмента кода с фактическими фрагментами данных в качестве аргументов.

Фрагменты кода, как и фрагменты данных, могут быть атомарными и структурированными.

Атомарный фрагмент кода является неделимой единицей исполнения и может принимать на вход и вырабатывать только атомарные фрагменты данных. При этом сам код, обрабатывающий данные, может быть написан на стороннем (последовательном) языке (таком, как C или Fortran). Это позволяет использовать внутри атомарных фрагментов кода уже существующие библиотеки оптимизированных функций. Можно сказать, что атомарный фрагмент кода во Freepal аналогичен элементарной операции в процедурном языке.

В отличие от атомарного структурированный фрагмент кода описывается с помощью выражений Freepal и может работать со структурированными фрагментами данных. Структурированный фрагмент кода – аналог процедуры в процедурном языке.

Кроме входных и выходных аргументов, атомарные и структурированные фрагменты кода могут иметь список параметров – дополнительный входной набор (скалярных) значений, которые могут использоваться как константы внутри кода.

Для декларации (атомарного/структурированного) фрагмента кода используется выражение **code**/**func** [<Выход>] <Имя> **<<**<Параметры>**>>** **(**<Вход>**)** <Тело>, где вход, выход и параметры – списки формальных аргументов в виде последовательностей <Тип> <Имя> (каждый из трех списков является необязательным и может отсутствовать), тело – код для исполнения.

Для декларации фрагмента вычислений используется выражение <Выход> = <Имя><<Параметры>>**(**<Вход>**)**, где имя – имя вызываемого фрагмента кода, вход и выход – списки имен фактических фрагментов данных, параметры – список фактических значений параметров (размеры списков и типы фрагментов данных должны совпадать с указанными в декларации фрагмента кода).

Выражения языка

В языке Freepal поддерживаются следующие выражения для построения структурированных фрагментов кода:

- Декларация фрагмента данных.
- Декларация фрагмента вычислений.
- Блок – последовательность выражений, заключенная в фигурные скобки: **{** <Выр1> <Выр2> … <ВырN> **}**. Тело структурированного фрагмента кода является блоком. Если блок содержит декларацию фрагмента данных, этот фрагмент данных считается принадлежащим этому блоку и может использоваться только в его пределах. Блоки могут быть вложены друг в друга.
- Индексированное множество – описание множества выражений (чаще всего фрагментов вычислений) в виде **[**<Индексация>**]** <Выражение>, где индексация – последовательность конструкций <Переменная индексации>=<Нижний предел>**..**<Верхний предел> (нижний и верхний пределы могут задаваться произвольными арифметическими выражениями). Для каждой комбинации значений переменных индексации из указанных пределов порождается копия выражения (для которой переменные индексации являются параметрами-константами с соответствующими значениями). Аналог такого выражения в других языках – параллельный цикл for.
- Итерационное множество – описание счетного (потенциально бесконечного) множества: **while** <Логическое условие> <Выражение>. Если условие истинно, выражение исполняется, после чего условие проверяется снова и т.д. Исполнение завершается, когда условие становится ложным. Аналог цикла while.
- Условный оператор: **if** <Логическое условие> <Выр1> [**else** <Выр2>]. В зависимости от истинности условия исполняется первое или второе выражение. Часть с **else** может отсутствовать.

Порядок исполнения

Важным моментом во Freepal является определение допустимого порядка исполнения фрагментов. Так как язык является параллельным, по умолчанию предполагается, что все объявленные фрагменты вычислений могут быть исполнены параллельно.

Однако такое полностью параллельное исполнение практически не встречается в реальных задачах. В подавляющем большинстве случаев между фрагментами вычислений имеются зависимости по фрагментным данным (например, в виде производитель-потребитель). В связи с этим в языке используется следующий принцип: если один фрагмент вычислений вырабатывает фрагмент данных, а далее в тексте программы находится другой фрагмент вычислений, который его потребляет, между ними устанавливается порядок по исполнению, контролирующий, чтобы второй фрагмент вычислений запустился только после (но не обязательно сразу) завершения исполнения первого. Этот принцип позволяет обеспечить корректный порядок исполнения для зависимых фрагментов вычислений и в то же время сохранить параллельное исполнение для остальных независимых фрагментов вычислений.

Обнаружение зависимостей и построение порядка исполнения производится компилятором Freepal при компиляции фрагментированной программы. Если не все зависимости можно определить при компиляции, построение и уточнение зависимостей производится во время исполнения runtime-системой. Уточнение может производиться до уровня атомарных фрагментов данных и вычислений.

Для тех случаев, когда нужно изменить порядок исполнения по умолчанию, могут использоваться следующие операторы языка:

- **<<**: <ФВ1> **<<** <ФВ2> – последовательное исполнение фрагментов вычислений, даже если между ними нет явных зависимостей по данным.
- ||: <ФВ1> || <ФВ2> – параллельное исполнение фрагментов вычислений, даже если между ними есть зависимость. Может применяться в случае редукции в один фрагмент данных, когда порядок вычисления элементов редукции не важен (runtime-система обеспечит корректный доступ к переменной редукции).

Компиляция и исполнение

При поддержке runtime-системой основных конструкций языка компиляция Freepal-программы может быть произведена путем трансляции выражений языка в обращения к runtime-системе. Таким образом, для

каждого фрагмента кода генерируется его эквивалент, использующий функции runtime-системы, а исполнение фрагментов вычислений заключается в исполнении такого сгенерированного кода.

Из этого следует, что наличие эффективной runtime-системы [3] обеспечивает эффективность самого языка в целом.

Заключение

Разработаны синтаксис, семантика и основные принципы языка фрагментированного программирования Freepal.

В дальнейшем планируется исследование и разработка алгоритмов и приемов оптимизации фрагментированной Freepal-программы на стадии компиляции.

Литература

1. **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.
2. **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.
3. **Щукин Г.А.** Runtime-система FP RTS управления исполнением фрагментированных программ на мультикомпьютере // Сборник докладов Пятой Сибирской конференции по параллельным и высокопроизводительным вычислениям, Томск, 1–3 декабря 2009. – Томск: Изд-во Том. ун-та, 2009. – С. 182–186.

Методика фрагментации численных алгоритмов для библиотеки параллельных численных подпрограмм*

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

Представлены первые результаты разработки методики фрагментации численных алгоритмов. Выделены принципы построения фрагментированных алгоритмов. Рассмотрен вопрос фрагментации регулярных структур, образуемых объектами алгоритма.

Введение

Часто метод моделирования требует, чтобы реализующая его параллельная программа обладала многими динамическими свойствами, такими как настройка на доступные ресурсы, балансировка загрузки с учетом поведения модели, выполнение вычислений на фоне межузловых обменов данными. Эти задачи относятся к области системного программирования и требуют наличия специальных знаний и навыков у специалистов по численному моделированию.

Технология фрагментированного программирования призвана автоматизировать решение задач системного параллельного программирования при реализации численных моделей. Она основана на методе синтеза параллельных программ [1, 2]. Суть ее состоит в следующем:

- Параллельная программа представляется в виде двух явным образом разделенных слоев:
  - теоретико-множественное описание алгоритма – переносимое, независимое от архитектуры;
  - реализационная часть – исполнительная система, обеспечивающая эффективное исполнение алгоритма на конкретной вычислительной системе.
- Алгоритм представляется в специальном фрагментированном виде: в виде множества фрагментов данных, множества фрагментов вычислений и отношений на них. Разделение на фрагменты сохраняется до самого момента исполнения – это позволяет исполнитель-

* Проект поддержан грантом РФФИ № 10-07-00454а.

ной системе динамически выполнять настройку на имеющиеся ресурсы.

Фрагментированный алгоритм содержит только минимальное управление, определяемое зависимостями по данным, и не содержит распределения ресурсов. Таким образом, он допускает множество способов исполнения, что обеспечивает его переносимость. Задача исполнительной системы – выполнить отображение объектов алгоритма (переменных, операций) на ресурсы конкретной вычислительной системы, автоматически обеспечивая все необходимые динамические свойства параллельной программы [3]. Фрагментация – это технологический прием, позволяющий уменьшить число объектов алгоритма и тем самым упростить задачу построения эффективного распределения ресурсов и управления.

В качестве первого приложения технологии фрагментированного программирования разрабатывается библиотека фрагментированных численных подпрограмм [4] на базе экспериментальной системы фрагментированного программирования LuNA [3]. В настоящий момент выполнена фрагментация ряда алгоритмов матричных операций для плотных и разреженных матриц, алгоритмов вычислений по явной схеме на структурированных сетках, задачи статистического моделирования. Ведется работа над фрагментацией метода частиц-в-ячейках.

Настоящая работа посвящена проблеме построения таких фрагментированных алгоритмов, которые могли бы эффективно исполниться на современных параллельных вычислительных системах при наличии соответствующей исполнительной системы. На основе накопленного опыта фрагментации численных алгоритмов формируется методика построения фрагментированных алгоритмов, которая может быть поддержана исполнительной системой.

Основные определения

Будем пользоваться следующим определением алгоритма. Алгоритм – это совокупность следующих множеств и отношений:

- множество переменных единственного присваивания;
- множество операций единственного срабатывания;
- отношение вход-выход на множествах переменных и операций.

Алгоритм эквивалентным образом можно представить как множество функциональных термов (возможно, потенциально бесконечное).

Пусть задан алгоритм с атомарными (неделимыми) переменными и операциями. Будем называть его исходным алгоритмом. Далее будем

рассматривать процесс построения фрагментированного алгоритма из исходного. В процессе фрагментации происходят следующие преобразования:

- некоторые атомарные переменные исходного алгоритма объединяются в подмножества, каждое из которых формирует одну новую агрегированную переменную;
- некоторые части исходного алгоритма (подмножества переменных и операций) объединяются в новые агрегированные операции, причем переменная может войти в агрегированную операцию, только если все связанные с ней операции (через отношение вход-выход) входят в ту же агрегированную операцию;
- связи вход-выход соответствующим образом объединяются.

Фрагментированный алгоритм – это алгоритм с агрегированными переменными и операциями. Агрегированные переменные будем называть фрагментами данных, агрегированные операции – фрагментами вычислений.

Принципы фрагментации

Исходя из требования эффективного исполнения фрагментированной программы, можно сформулировать следующие принципы построения фрагментированных алгоритмов:

1. Число объектов и связей алгоритма при фрагментации должно уменьшаться. Чем меньше число объектов и связей в алгоритме, тем проще исполнительной системе решить задачу отображения их на ресурсы.
2. Фрагменты алгоритма должны быть как можно более единообразными (во всех смыслах). Единообразие фрагментов алгоритма позволяет использовать одни и те же способы обработки для всех фрагментов, что упрощает устройство исполнительной системы и повышает ее эффективность.
3. Фрагментированный алгоритм должен иметь возможность настраиваться на вычислительную систему. Фрагментацию алгоритма можно выполнять по-разному. Разные способы фрагментации могут быть оптимальными для разных вычислительных систем. Необходимо, чтобы фрагментированный алгоритм имел явно задаваемые параметры, оптимальные значения которых можно выбрать вручную или автоматически для конкретной вычислительной системы. Таким параметром, например, является размер фрагментов. При изменении размера задачи оп-

тимальные значения параметров алгоритма меняться не должны, должно меняться только число фрагментов.

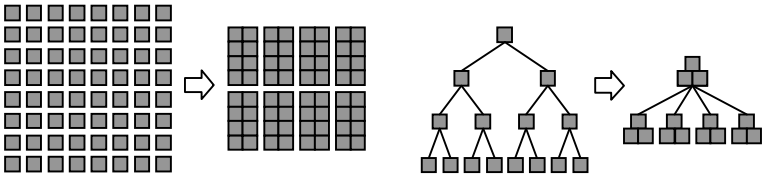
4. Фрагментированный алгоритм должен быть масштабируемым – число фрагментов должно быть пропорционально объему вычислений. Это значит, что если с увеличением размера задачи объем вычислений увеличится в 2 раза, то и число фрагментов должно увеличиться примерно в 2 раза.

Конечно, не каждый алгоритм можно фрагментировать так, чтобы он удовлетворял всем принципам. Но чем больше принципов соблюдено при построении фрагментированного алгоритма, тем лучше будут свойства полученной на его основе фрагментированной программы.

Фрагментация регулярных структур

Большинство численных алгоритмов имеют регулярным образом организованные данные и вычисления. Фрагментацию регулярных структур, образуемых объектами алгоритма, следует рассмотреть отдельно.

Исходя из принципа единообразия, объекты алгоритма должны быть объединены во фрагменты одинаковым образом (рис. 1).



(а) (б)

Рис. 1. Примеры фрагментации регулярных структур алгоритма

В исходном алгоритме подтермы составляют некоторые структуры, каждая из которых имеет, как правило, один тип регулярности. После фрагментации такой структуры типов регулярности становится два. Будем называть их внутренней структурой и внешней структурой:

- Внутренняя структура фрагмента данных – структура данных, образуемая атомарными переменными внутри фрагмента данных.
- Внешняя структура фрагментов – структура, образуемая фрагментами данных или вычислений одной регулярной структуры алгоритма.

Например, на рис. 1, *а* и внешняя структура – двумерный массив 2×4, а внутренняя структура – двумерный массив 4×2. На рис. 1, *б*

внешняя структура – дерево из 5 элементов, а внутренняя – дерево из 3 элементов.

При фрагментации алгоритмов встает задача выбора внешней и внутренней структуры для каждой регулярной структуры исходного алгоритма. Внешняя структура выбирается, прежде всего, исходя из рассмотренных выше принципов. Выбор внутренней структуры относится уже к интерпретации объектов алгоритма. Можно назвать следующие критерии выбора внутренней структуры: минимизация размера фрагмента, минимизация числа операций, наличие оптимизированных библиотечных операций.

Для примера рассмотрим фрагментацию алгоритма умножения разреженной матрицы на вектор. Пусть A – матрица размера $N \times N$, x,y – векторы размера N. Алгоритм умножения произвольной матрицы на вектор описывается формулой $y_i = \sum_{j=1}^{N} a_{ij} x_j$, где a_{ij} – элементы матрицы A; x_j, y_i – элементы векторов x и y.

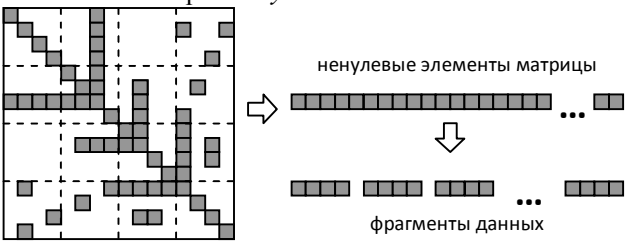


Рис. 2. Фрагментация разреженной матрицы. Способ 1

Пусть вектор x разделен на P фрагментов равного размера. Рассмотрим два способа фрагментации матрицы A. На рис. 2 представлен первый способ фрагментации данных: разделение списка ненулевых элементов матрицы на фрагменты одинакового размера.

Пусть число фрагментов матрицы будет Q. Для записи алгоритма умножения матрицы на вектор потребуется описать $P \times Q$ фрагментов вычислений, выполняющих перемножение элементов, т.к. нет простого способа определить, какой элемент матрицы в каком фрагменте окажется. Если мы добавим еще один фрагмент матрицы, то число фрагментов вычислений увеличится на P. Таким образом, данный алгоритм плохо масштабируется.

На рис. 3 представлен второй способ фрагментации матрицы A, при котором внешняя структура – двумерный массив, а внутренняя – список ненулевых элементов матрицы, попавших в соответствующий блок массива. При таком способе фрагментации фрагменты матрицы оказы-

ваются разного размера. Но число фрагментов вычислений для алгоритма умножения равно числу фрагментов матрицы, поэтому данный алгоритм является хорошо масштабируемым.

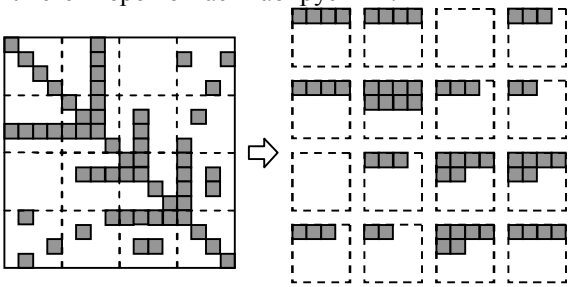


Рис. 3. Фрагментация разреженной матрицы. Способ 2

Заключение

Представлены первые результаты разработки методики фрагментации численных алгоритмов. Выделены принципы построения фрагментированных алгоритмов. Рассмотрен вопрос фрагментации регулярных структур.

Дальнейшая работа по разработке методики фрагментации связана с рассмотрением конкретных внешних и внутренних структур для реализации распространенных численных методов и алгоритмов. Кроме того, формируется список требований к исполнительной системе, которым она должна удовлетворять для обеспечения должного качества исполнения фрагментированных программ.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 128 с.
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, № 6. – P. 755–765.
3. **Malyshkin V.E., Perepelkin V.A.** Optimization of Parallel Execution of Numerical Programs in LuNA Fragmented Programming System // MTPP–2010 revised selected papers. – N.Y.: Springer, LNCS 6083, 2010. – P. 1–10.

4. **Kireev S.E., Malyshkin V.E.** Fragmentation of Numerical Algorithms for Parallel Subroutines Library // The Journal of Supercomputing. – 2011. – Vol. 57, №. 2. – P. 161–171.

Организация эффективных вычислений в распределенной среде NumGRID

М.А. Городничев
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Представлены результаты экспериментов по объединению вычислительных кластеров на основе программного комплекса NumGRID для решения задач численного моделирования. Показано, что существуют задачи численного моделирования, которые демонстрируют приемлемый уровень эффективности использования вычислительных ресурсов при запуске в распределенной среде NumGRID.

Введение

Программный комплекс NumGRID [1] предназначен для объединения разнородных вычислительных кластеров в единый вычислительный ресурс на основе частичной реализации стандарта MPI-2.2 [2]. NumGRID обеспечивает возможность запускать MPI-приложение так, чтобы процессы приложения были распределены по рабочим узлам нескольких кластеров. При этом процессы составляют один коммуникатор MPI и имеют возможность обмениваться между собой средствами MPI.

NumGRID позволяет:

- решать задачи, для которых недостаточно ресурсов отдельных кластеров;
- продлять жизнь устаревающего оборудования за счет объединения с новым;
- распределять части комплексных задач между специализированными кластерами в соответствии с индивидуальными требованиями частей к оборудованию и программному обеспечению;
- повысить гибкость при планировании распределения задач между кластерами в грид;
- планировать постепенное наращивание мощностей распределенной системы.

Обзор NumGRID

Схема устройства объединенного вычислительного ресурса NumGRID представлена на рис. 1.

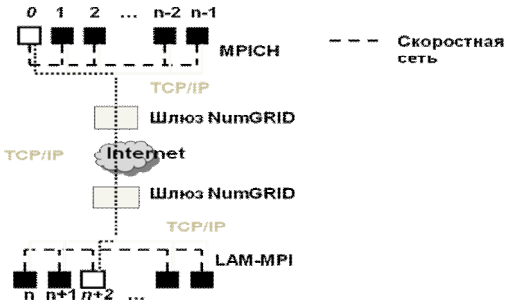


Рис. 1. Устройство NumGRID

В примере объединяются два кластера через Internet. Каждый кластер имеет рабочие узлы, объединенные высокоскоростной сетью (Myrinet, Infiniband и др.). Также узлы связаны с головным узлом сетью с меньшей пропускной способностью, поддерживающей протокол TCP. Процессы приложения MPI, находящиеся на рабочих узлах одного кластера, обмениваются между собой через высокоскоростную сеть средствами специфичной для кластера библиотеки MPI, позволяющей использовать возможности высокоскоростной сети. В примере это библиотеки MPICH и LAM-MPI. Для процессов, распределенных по рабочим узлам двух кластеров, поддерживается глобальная адресация в рамках одного коммуникатора MPI. Сообщения между процессами, расположенными на разных кластерах, проходят путь через головные узлы кластеров, где для этих целей перед стартом приложения запускаются шлюзы.

Пользователь вначале запускает шлюзы на каждом кластере. Шлюзы устанавливают между собой связь. Пользователь загружает на головные узлы кластеров исходный код своего приложения и исходные данные, собирает приложение на каждом кластере с библиотекой NumGRID-MPI, запускает необходимое число процессов на каждом кластере как локальные задачи MPI. Локальная задача получает в параметрах информацию о месте данной группы процессов в общей распределенной задаче, что позволяет обеспечить глобальную адресацию процессов. В конце работы пользователь забирает с каждого кластера результаты работы программы.

Экспериментальное исследование NumGRID

Эксперименты по запуску приложений в NumGRID проводились на объединении кластеров Сибирского суперкомпьютерного центра

(ССКЦ) [8] и Новосибирского государственного университета (НГУ). Кластеры построены на основе двухпроцессорных узлов с процессорами Intel Xeon E5540 (4 ядра) и внутренней коммуникационной сетью InfiniBand. Пропускная способность канала между головными узлами кластеров – 10 Гб/с.

Решение волнового уравнения

Ниже представлены графики, демонстрирующие характеристики исполнения программы решения волнового уравнения с помощью двухслойной явной схемы, размер задачи 10000x10000. На всех диаграммах запись «P (NxS)» означает «всего P ядер, из них N – на кластере 1 и S – на кластере 2».

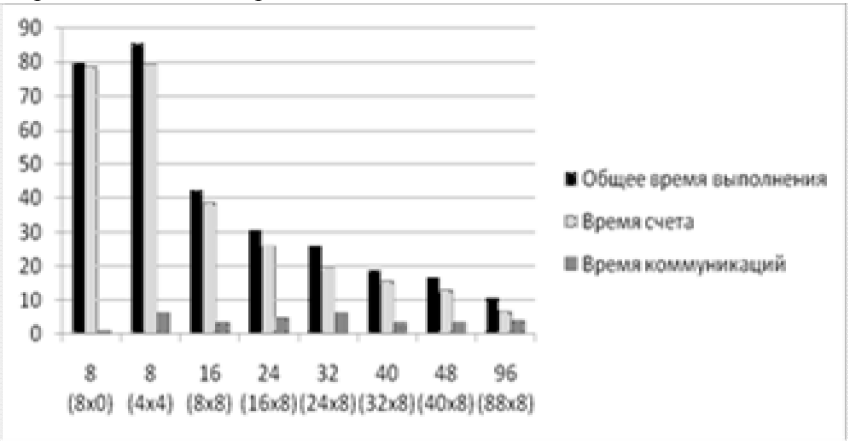


Рис. 2. Время решения волнового уравнения явным методом на NumGRID, с

Из рис. 2 видно, что при переходе к распределенным вычислениям (с 8x0 на 4x4) увеличивается общее время работы программы за счет увеличения расходов на коммуникации. Расходы на коммуникации в данном примере увеличиваются в 5 раз, что приводит к росту времени работы программы на ~6%.

Генерация случайных чисел и расчет статистик

Ниже представлены результаты тестирования программы, которая параллельно генерирует случайные величины и вычисляет статистики. Программа используется для моделирования физических процессов методами Монте-Карло. Программа осуществляет редкие коллектив-

ные коммуникации для сбора статистик, в остальное время процессоры выполняют существенные по объему вычисления независимо. Это позволяет ожидать, что относительно плохая пропускная способность сети между кластерами незначительно скажется на общей производительности программы.

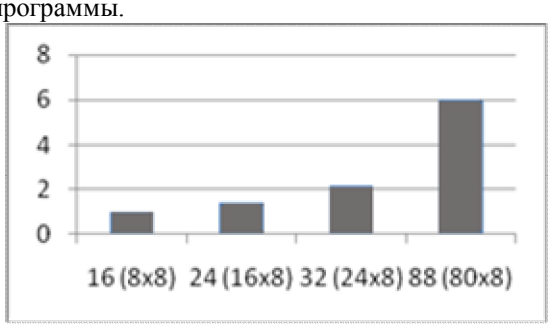


Рис. 3. Время генерации 2000 значений случайной величины и расчета статистик на NumGRID, с. При этом время генерации 2000 значений случайной величины и расчета статистик на одном ядре кластера 1: 0,0109 с

Время исполнения программы при переходе от решения на одном ядре к решению на 16 ядрах, по 8 ядер на каждом кластере, уменьшается в 21 раз. Объяснить подобное сверхлинейное ускорение можно более эффективным использованием кэшей процессоров при увеличении количества процессоров и низкими коммуникционными расходами, свойственными данной задаче. При дальнейшем увеличении количества вовлеченных ресурсов так же отмечается сверхлинейное ускорение. Рисунок 4 дает более ясное представление об ускорении времени решения при увеличении количества процессоров.

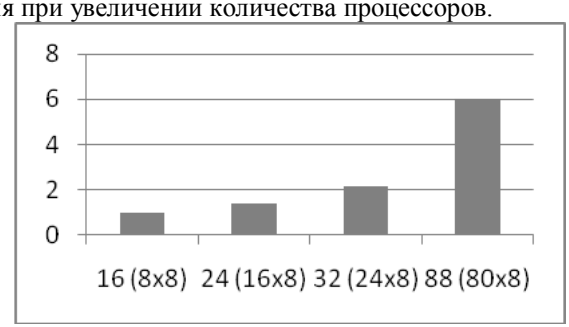


Рис. 4. Ускорение генерации 2000 значений случайной величины и расчета статистик на NumGRID относительно времени работы программы на двух узлах: один узел кластера НГУ (8 ядер) и один узел кластера ССКЦ (8 ядер)

В ходе экспериментов показано, что использование дополнительных процессоров из другого кластера позволяет уменьшать время выполнения приложения при определенных конфигурациях запуска. В частности, на задаче решения волнового уравнения достигается эффективность 90% на 40 процессорных ядрах относительно производительности программы на 8 ядрах, и 65% – на 96 ядрах. На задаче генерации случайных чисел достигается эффективность 92% на 24 ядрах и 109% на 88 ядрах.

Несмотря на существенную разницу в производительности систем коммуникации внутри кластеров и между кластерами, время решения задач увеличивается умеренно для решения волнового уравнения на том же количестве процессоров и уменьшается для решения задачи поиска статистик. Можно заключить, что существует класс задач, которые могут быть решены на объединении кластеров за приемлемое для пользователей время. Вместе с тем объединение кластеров может позволить решать задачи большего объема, чем был бы способен решить один кластер за то же время.

Заключение

Проведены испытания системы NumGRID по объединению кластеров Сибирского суперкомпьютерного центра и Новосибирского государственного университета. Эксперименты демонстрируют, что существуют классы задач численного моделирования, для решения которых распределенные вычислительные системы на основе NumGRID применимы.

Дальнейшая работа заключается в расширении поддержки стандарта MPI-2.2, адаптации существующих приложений для работы в среде NumGRID, создании методов и средств разработки программ численного моделирования для неоднородных вычислительных сред.

Литература

1. **Fougere D., Gorodnichev M., Malyshkin N. et al.** NumGrid Middleware: MPI Support for Computational Grids // Parallel Computing Technologies: 8th International Conference, PaCT 2005, Krasnoyarsk, Russia, September 5–9, 2005. Proceedings. – N.Y. : Springer, 2005. – LNCS Vol. 3606. – P. 313–320.

2. MPI Documents [Электронный ресурс]. – Режим доступа: http://www.mpi-forum.org/docs/docs.html (дата обращения: 15.12.2011).

Применение суперкомпьютеров для краткосрочного прогноза качества воздуха над территорией г. Томска

А.А. Барт, А.В. Старченко, А.З. Фазлиев
Томский государственный университет

Описывается информационно-вычислительная система, ядром которой является математическая модель переноса примеси. Основная задача системы – выполнение прогностических расчетов качества атмосферного воздуха над территорией г. Томска на основе метеорологического прогноза по оперативной глобальной модели Гидрометцентра РФ. Модель переноса примеси реализована на кластере ТГУ СКИФ Cyberia, что позволяет получать прогноз в короткие сроки.

Введение

Атмосфера Земли – сложная система, в которой ряд физических и химических процессов протекает одновременно. Математические модели, в свою очередь, предоставляют необходимый «каркас» для объединения представлений об отдельных атмосферных процессах и их взаимодействии между собой.

Успешное решение задач прогноза уровня загрязнения воздуха основано на использовании математических моделей, учитывающих физические особенности распространения примесей в атмосфере, связи между концентрациями примесей и метеорологическими параметрами: скоростью и направлением ветра, инверсией (повышение температуры воздуха с высотой), осадками, туманами.

Математическая модель переноса примеси

Для расчета концентрации компонентов примеси с учетом химических реакций применяется эйлерова модель турбулентной диффузии, включающая транспортные уравнения с описанием адвекции, турбулентной диффузии и химических реакций:

$$\frac{\partial C_i}{\partial t}+\frac{\partial UC_i}{\partial x}+\frac{\partial VC_i}{\partial y}+\frac{\partial WC_i}{\partial z}=-\frac{\partial}{\partial x}\langle c_i u\rangle-\frac{\partial}{\partial y}\langle c_i v\rangle-\frac{\partial}{\partial z}\langle c_i w\rangle- \quad (1)$$
$$-\sigma_i C_i+S_i+R_i, i=1,..,n_s.$$

Здесь C_i, c_i – осредненная и пульсационная составляющие концентрации i-й компоненты примеси; U, V, u, v – осредненные и пульсационные составляющие вектора горизонтальной скорости ветра; W, w – осредненная и пульсационная составляющие вертикальной компоненты скорости примеси; $\langle\,\rangle$ – осреднение по Рейнольдсу; S_i – источниковый член, представляющий выбросы компонентов примеси в атмосферу; R_i описывает образование и трансформацию вещества за счет химических и фотохимических реакций с участием компонентов примеси; σ_i – скорость влажного осаждения примеси за счет осадков; n_s – количество химических компонентов примеси, концентрации которых необходимо определить; x, y – горизонтальные координаты, ось Ox направлена на восток, Oy – на север; z – вертикальная координата; t – время; $0 \le t \le T$, $-L_x/2 \le x \le L_x/2$, $-L_y/2 \le y \le L_y/2$, $0 \le z \le h$, T – время моделирования, Lx, Ly – горизонтальные размеры области, h – высота расчетной области [1].

Задание параметров атмосферного пограничного слоя

Для реализации модели переноса примеси необходимо определить для каждого момента времени профили компонент вектора скорости, температуры и турбулентных характеристик. Они получались в результате усвоения результатов 48-часового метеорологического прогноза, выполняемого с использованием глобальной модели ПЛАВ [2]. ПЛАВ – это полулагранжева модель атмосферы с высоким разрешением (горизонтальная сетка $0{,}72^{\circ}$ по широте, $0{,}9^{\circ}$ по долготе, 28 вертикальных слоев), принятая Гидрометцентром в качестве оперативной модели для среднесрочного прогноза с выдачей результатов через 6 ч.

Для усвоения данных прогноза по модели ПЛАВ в разрабатываемой системе прогноза химической погоды в городе в настоящее время используется одномерная нестационарная математическая модель атмосферного пограничного слоя (АПС), которая позволяет подробно рассчитывать вертикальную структуру нижней тропосферы и ее турбулентные характеристики [2].

Расчет химических реакций

Моделирование химических и фотохимических реакций в (1) проводилось на основе химического механизма, предложенного A. Zakey (Датский метеорологический институт) на основе скорректированного сокращенного химического механизма образования приземного озона, взятого из работ [4, 5]. В сокращенной кинетической схеме учитываются 20 химических реакций между следующими компонентами: NO_2,

NO, $O(^1D)$, $O(^3P)$, O_3, HO, H_2O_2, HO_2, CO, SO_2, HC, HCHO, RO_2 , O_2, N_2, H_2O. Значения концентраций O_2, N_2, H_2O , в силу малого влияния на них рассматриваемых химических реакций, принимаются постоянными [4, 5].

Реакции фотохимической схемы, используемой в DMI

Реакции фотолиза	***Неорганические реакции***
$NO_2+hv \rightarrow O(^3P)+NO$	$O(^3P)+O_2 \rightarrow O_3$
$O_3+hv \rightarrow O(^1D)+O_2$	$O(^1D)+N_2 \rightarrow O(^3P)+N_2$
$HCHO+hv \rightarrow 2HO_2+CO$	$O(^1D)+O_2 \rightarrow O(^3P)+O_2$
$HCHO+hv \rightarrow H_2+CO$	$O(^1D)+H_2O \rightarrow 2HO$
Органические реакции	$HO_2+NO \rightarrow NO_2+HO$
$RO_2+NO \rightarrow NO_2+HO_2+HCHO$	$O_3+NO \rightarrow NO_2+O_2$
$HCHO+HO+O_2 \rightarrow HO_2+CO+H_2O$	$CO+HO \rightarrow HO_2+CO_2$
$RO_2+HO_2 \rightarrow ROOH+O_2$	$HC+HO \rightarrow RO_2+H_2O$
$RO_2+RO_2 \rightarrow prod$	$HO+NO_2 \rightarrow HNO_3$
	$HO_2+HO_2 \rightarrow H_2O_2+O_2$
	$2HO_2+H_2O \rightarrow H_2O_2+H_2O+O_2$
	$HO+SO_2 \rightarrow H_2SO_4+HO_2$

Численный метод и его параллельная реализация

При построении разностной схемы для уравнения (1) использовалась сетка с переменным по z шагом, сгущающаяся к поверхности Земли. При замене дифференциального уравнения (1) его разностным аналогом применялись явно-неявные разностные схемы с первым порядком аппроксимации по времени и вторым по координатам. Все члены уравнения (1) аппроксимируются на k-м слое по времени, за исключением вертикальной диффузии и членов, отвечающих за влажное осаждение примеси, поступление примеси от источников и химические реакции, которые берутся на (k+1)-м слое по времени.

Для получения конечно-разностного аналога уравнения (1) используется метод конечных объемов. Для аппроксимации диффузионных потоков используется центрально-разностный оператор. Для аппрок-

симации адвективных членов применяется направленная противопотоковая схема MLU [3].

При моделировании переноса примеси используются среднегодовые данные антропогенной эмиссии загрязнителей городского воздуха. Источники делятся на 3 категории: точечные, линейные (дороги) и площадные (крупные предприятия). Для каждой из ячеек поверхности задается расход выбросов 17 различных по составу химических веществ.

Параллельная версия алгоритма была построена с использованием двумерной декомпозиции сеточной области. Декомпозиция осуществлялась по направлениям *Ox* и *Oy*.

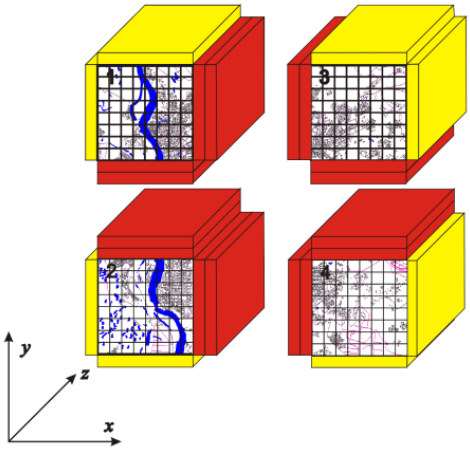


Рис. 1. Схема двумерной декомпозиции расчетной области

На рис. 1 присутствуют светлые блоки по краям расчетной области, являющиеся фиктивными границами, темные блоки, расположенные со стороны разделения области, являются блоками обмена данными между процессорами. При такой декомпозиции можно использовать большое количество процессоров суперкомпьютера, а доля временных затрат, связанных с необходимостью на каждом шаге по времени передавать данные между процессорными элементами, меньше, чем при одномерной декомпозиции.

Ускорение и эффективность

Для исследования ускорения и эффективности модели были проведены расчеты на новом (1 узел – 2 шестиядерных процессора Intel Xeon 5670 2,93 ГГц с 8 Gb RAM) и старом (1 узел – 2 двуядерных процессора Intel Xeon 5150 2,66 ГГц с 8 Gb RAM) сегменте кластера ТГУ СКИФ Cyberia [7]. Расчеты выполнялись на 4, 16, 25 и 100 ядрах. На рис. 2 представлены графики зависимости ускорения и времени, требуемого на расчет, от количества используемых ядер.

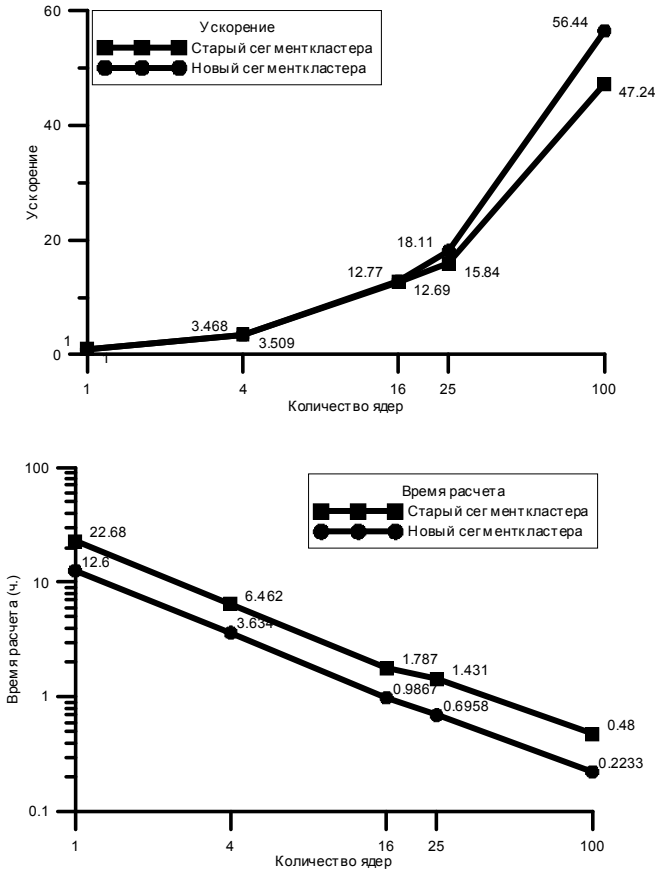


Рис. 2. Графики ускорения и времени, затрачиваемого на прогноз модели переноса примеси

Для ежедневного расчета используется 16 ядер кластера. Количество ядер, как видно из графика ускорения (см. рис. 2), выбрано оптимально для выполнения расчетов. При использовании 16 ядер время выполнения одного расчета составляет 1 ч. В случае оперативного расчета присутствует возможность запуска на большем количестве ядер: при запуске моделирования на 100 ядрах расчет выполнится за 15 мин. Но при увеличении количества ядер с 16 до 100 (в 6,25 раза) время, затрачиваемое на прогноз, уменьшается всего в 4 раза. Кроме того, не всегда имеется возможность получить 100 ядер для расчета.

Сравнение результатов моделирования с данными измерений

Проверка полученных результатов моделирования осуществлялась путем сравнения расчетов с данными измерений, проведенными на ТОР-станции Института оптики атмосферы СО РАН в Томске [7].

Расчетная область представляет собой часть атмосферы размером 50x50x2 км, выбранная таким образом, что ее квадратное основание содержит участок подстилающей поверхности с г. Томском в центре. В расчете учитывается эмиссия примеси от линейных, площадных и точечных источников, находящихся на поверхности или приподнятых над землей. Область покрывается вычислительной сеткой с размерами 100x100x28 узлов. Период прогностического моделирования обычно составляет 24 ч.

На рис. 3 сплошной линией представлены результаты численного моделирования, символами отмечены результаты измерений, проводимых на ТОР-станции ИОА, для каждого измерения вдоль оси значений приведена погрешность. На графиках дается сравнение измерений и расчетов силы и направления ветра, а также концентрации таких загрязнителей, как монооксид углерода (CO), диоксид азота (NO_2) и озон (O_3). Из графиков сравнения видно, что математическая модель хорошо предсказывает изменение ветра и значений рассматриваемых концентраций в течение суток.

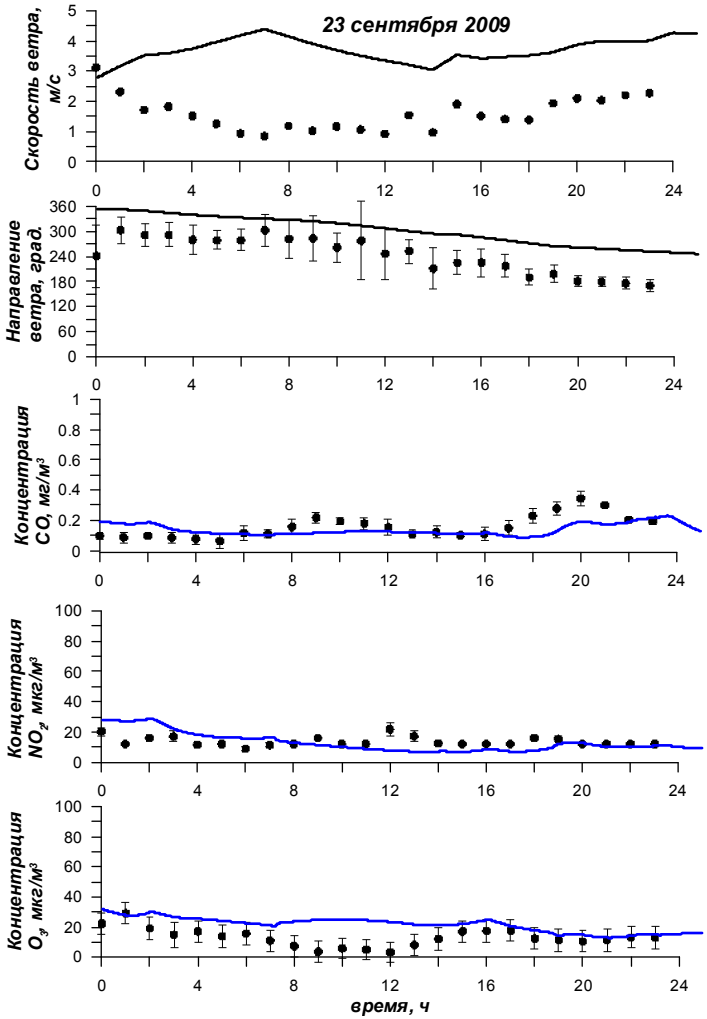


Рис. 3. Сравнение результатов моделирования с данными ТОР-станции 23 сентября 2009 г.

Заключение

Представленная в работе информационно-вычислительная система имеет ряд преимуществ, позволяющих использовать ее в реальных условиях в оперативном режиме. Такая возможность появилась благо-

даря усвоению данных глобального прогноза метеорологических характеристик, а также за счёт параллельной реализации модели переноса примеси при помощи двумерной декомпозиции расчетной области. Ежесуточно информационно-вычислительная система в автоматическом режиме производит получение начальных данных и запуск процесса моделирования на кластере ТГУ СКИФ Cyberia. Полученные результаты представляются в сети Internet в виде полей прогноза распространения в течение суток примеси, наложенных на карту местности.

Литература

1. **Беликов Д.А., Старченко А.В.** Исследование образования вторичных загрязнителей (озона) в атмосфере г. Томска // Оптика атмосферы и океана. – 2005. – Т. 18, №05-06. – С. 435–443.
2. **Толстых М.А., Богословский Н.Н., Шляева А.В., Юрова А.Ю.** Полулагранжева модель атмосферы ПЛАВ // Гидрометцентру России 80 лет. – М.: Триада, 2010. – С. 193–216.
3. **Барт А.А., Беликов Д.А., Старченко А.В.** Математическая модель для прогноза качества воздуха в городе с использованием суперкомпьютеров // Вестник Томского государственного университета. – 2011. – № 3. – С. 15–24.
4. **Noll B.** Evaluation of a bounded high-resolution scheme for combustor flow computations // AIAA Journal. – 1992. – Vol. 30, № 1. – P. 62–68.
5. **Srivastava R.K., McRae D.S., and Odman M.T.** Simulation of a reacting pollutant puff using an adaptive grid algorithm // J. Geophys. Res. – 2001. – Vol. 106. – P. 24245–24257.
6. **Stockwell W.R., Gotliff W.S.** Comment on «Simulation of a reacting pollutant puff using an adaptive grid algorithm» by R.K. Srivastava et al. // J. Geophys. Res. – 2002. – Vol. 107. – P. 4643–4650.
7. Межрегиональный супервычислительный центр ТГУ http://skif.tsu.ru
8. Сайт лаборатории климатологии атмосферного состава Института оптики атмосферы СО РАН. – Режим доступа: http://lop.iao.ru/

Параллельные алгоритмы для численного решения обратных задач переноса примеси с использованием данных мобильных измерений

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

Рассматривается параллельная реализация численного решения обратных задач переноса примеси с использованием мобильных измерений. При построении параллельных алгоритмов решения обратной задачи использовались функциональная, геометрическая декомпозиции и их комбинации, которые отличаются простотой реализации и высокой эффективностью, что продемонстрировали тестовые расчеты, проведенные на кластере ТГУ СКИФ Cyberia.

Введение

В современном мире наряду с приборным контролем качества воздуха широко используются методы математического моделирования при анализе распространения примесей. Методы математического моделирования позволяют быстро и с малыми затратами выполнять прогноз распределения загрязнителей в атмосфере. Но сложность процессов распространения примеси делают модели оценки качества воздуха громоздкими и требовательными к вычислительным ресурсам. Перспективным способом решения этих проблем является использование суперкомпьютерной техники, которая обеспечивает существенное ускорение получения результатов расчетов.

Физическая постановка прямой задачи

Распространение примеси в приземном слое атмосферы над промышленным центром и его окрестностями исследуется при следующих условиях. Рассматривается ограниченный в прямоугольнике участок территории, на котором в течение времени исследования происходит выброс от различных источников. Мощность источников зависит от времени. Распределение по области исследования поступающей примеси определяется метеорологическими условиями. Кроме того, сделано предположение, что примесь не вступает в химические реакции с другими веществами и скорость ее переноса совпадает со скоростью движения окружающего воздуха.

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 [1]:

$$\frac{\partial C}{\partial t}+U\frac{\partial C}{\partial x}+V\frac{\partial C}{\partial y}+W\frac{\partial C}{\partial z}+\sigma C= \\ =\frac{\partial}{\partial x}\left[\Gamma\frac{\partial C}{\partial x}\right]+\frac{\partial}{\partial y}\left[\Gamma\frac{\partial C}{\partial y}\right]+\frac{\partial}{\partial z}\left[K_z\frac{\partial C}{\partial z}\right]+Q, \tag{1}$$

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

Необходимые для моделирования переноса и рассеяния газообразного выброса параметры (U,V,W,Γ,K_z,σ) берутся из анализа данных метеорологических наблюдений или результатов численного моделирования с использованием мезомасштабных теорий различного уровня замыкания [2].

Начальные и граничные условия для уравнения (1) представляются в виде

$$t=0:\ C(0,x,y,z)=C_0(x,y,z); \\ x=0:\frac{\partial C}{\partial x}=0;\ x=L_x:\frac{\partial C}{\partial x}=0; \\ y=0:\frac{\partial C}{\partial y}=0;\ y=L_y:\frac{\partial C}{\partial y}=0; \\ z=0:K_z\frac{\partial C}{\partial z}=\alpha C;\ z=L_z:\frac{\partial C}{\partial z}=0, \tag{2}$$

где α – скорость сухого осаждения примеси.

Физическая постановка обратной задачи переноса примеси

Требуется по известным метеорологическим параметрам атмосферы и результатам измерений концентрации газообразной примеси в N точках, проводимых в течение некоторого периода времени T, определить параметры (мощность, координаты и время срабатывания) источников примеси.

Математическая постановка сопряженной задачи переноса примеси

Получим математическую постановку обратной задачи. Для этого будем использовать метод, который основан на решении уравнения, сопряженного с полуэмпирическим уравнением турбулентной диффузии и двойственным представлением функционала от концентрации примеси [1]. Сопряженная постановка задачи получается с использованием задачи (1) – (2). Для этого уравнение (1) умножается на функции $C_k^* = C_k^*(t,x,y,z)$ и интегрируется по времени и пространству. Тогда используя интегрирование по частям, приходим к сопряженным постановкам:

$$-\frac{\partial C_k^*}{\partial t}-\frac{\partial UC_k^*}{\partial x}-\frac{\partial VC_k^*}{\partial y}-\frac{\partial WC_k^*}{\partial z}+\sigma C_k^*-\frac{\partial}{\partial x}\Gamma\frac{\partial C_k^*}{\partial x}-\frac{\partial}{\partial y}\Gamma\frac{\partial C_k^*}{\partial y}-$$
$$-\frac{\partial}{\partial z}K_z\frac{\partial C_k^*}{\partial z}=P_k \qquad (3)$$

со следующими начальными и граничными условиями:

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

где $P_k=\delta(x-x_k)\delta(y-y_k)\delta(z-z_k)\delta(t-t_k)$, $k=1,...,N$; N – количество измерений концентрации $C(t,x,y,z)$ в точках с координатами (x_k,y_k,z_k) в момент времени t_k.

Двойственное представление функционала можно записать в виде

$$J(C)=\int_0^T\int_0^{L_x}\int_0^{L_y}\int_0^{L_z}CP_k\,dzdydxdt=C_k=\int_0^T\int_0^{L_x}\int_0^{L_y}\int_0^{L_z}C_k^*Q\,dzdydxdt,\ k=1,...,N. \qquad (5)$$

При этом нахождение решения систем уравнений (3) – (4) и (5) позволит установить параметры источников загрязнения атмосферного воздуха.

Численное решение дифференциальных уравнений

Для численной реализации задачи (3) – (4) использовались метод конечного объёма и явные разностные схемы: противопотоковая схема, схема MLU Ван Лира или схема Ботта аппроксимации адвективных членов. Диффузионные слагаемые в уравнениях аппроксимируются со вторым порядком точности.

Параллельная реализация сопряженной задачи

При решении обратной задачи определения характеристик источников на каждом шаге по времени решается не одна, а N независимых сопряженных задач, следовательно, они могут решаться параллельно.

При использовании *функциональной декомпозиции* [3] распараллеливание метода численного решения обратной задачи переноса примеси производится с использованием принципа «master-slave». В этом случае 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 возвращают управляющему процессу, который ищет глобальный минимум функционала, позволяющий определить параметры источников загрязнения.

Также при решении сопряженных задач (3) – (4) использовалась одномерная (вдоль оси OY) *геометрическая декомпозиция* [3] сеточной области. При таком способе декомпозиции каждому процессору выделяется часть области исследования и столб атмосферы над ней. Перед началом вычислений каждый процессор получает информацию о физических параметрах своей подобласти и подготавливается к вычислениям. При этом во время вычислений каждый процессор обменивается с соседними процессорами данными, которые были получены при его выполнении. Обмен происходит вдоль вертикальных границ подобластей (Рис. 1) [4].

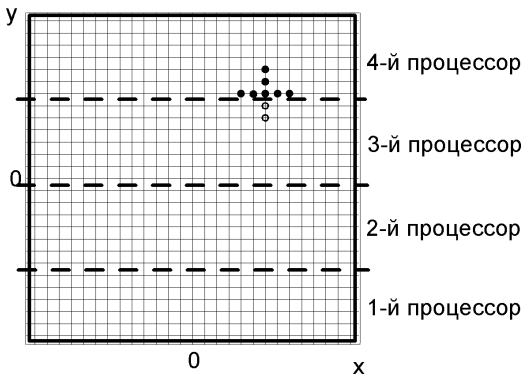


Рис. 1. Одномерная декомпозиция
(четырехпроцессорное вычислительное устройство)

В табл. 1 представлены значения времени выполнения параллельной программы, реализующей функциональную и геометрическую декомпозиции при решении обратной задачи переноса примеси при различном количестве измерений концентрации выброса.

Таблица 1. Время счета, с, для функциональной и геометрической декомпозиции

Количество измерений, *N*	Число процессоров	Время счета (функциональная декомпозиция), с	Время счета (геометрическая декомпозиция), с
5	5	323	627
10	10	337	643
20	20	354	743

Рассмотренные выше подходы создания параллельных версий алгоритмов решения обратной задачи переноса примеси показали неплохие результаты по времени проводимых расчетов, однако количество используемых активных процессоров в них ограничено: в случае функциональной декомпозиции – числом проведенных измерений, а при одномерной геометрической декомпозиции – размером вычислительной сетки и выбранным сеточным шаблоном. Поэтому был предложен *комбинированный метод* распараллеливания [3], в котором предлагается для увеличения в расчетах количества используемых активных процессоров совместить применение функциональной и геометрической декомпозиции. Его применение показало хорошую масштабируемость параллельной программы до 400 процессоров, т.е. обеспечение эффективности не ниже 50% при увеличении числа используемых

процессоров при неизменных остальных параметрах решаемой задачи, что наглядно видно из табл. 2.

Таблица 2. Время счета для комбинированного метода

Количество измерений, N	Время счета (последовательная программа), с	Число процессоров	Время счета (комбинированный метод), с
5	1626	100	33
10	3922	200	42
20	7799	400	59

Определение городских районов – загрязнителей атмосферного воздуха с использованием мобильных измерений

При решении рассматриваемой задачи оценки характера распределения и интенсивности источников загрязнения атмосферного воздуха в качестве входных параметров использовались результаты мобильных измерений монооксида углерода, которые были сделаны сотрудниками Института оптики атмосферы СО РАН при помощи мобильной станции АКВ-2 в г. Томске (Рис. 2). Область исследования имеет площадь размером 50х50 км, в ее центральной части находится городская территория. Верхняя граница расчетной области располагается на высоте 2000 м (Рис. 3).

Рис. 2. Станция АКВ-2

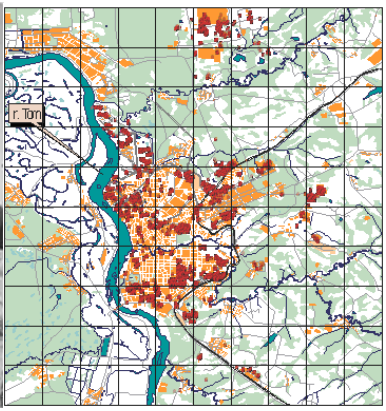


Рис. 3. Область исследования с нанесенной горизонтальной сеткой

Тестирование правильности решения обратной задачи проводилось с использованием результатов численного решения прямой задачи в качестве необходимых входных данных. На рис. 4 представлены изолинии интенсивности выброса примеси $\bar{Q}(x, y)$, полученные при решении обратных задач.

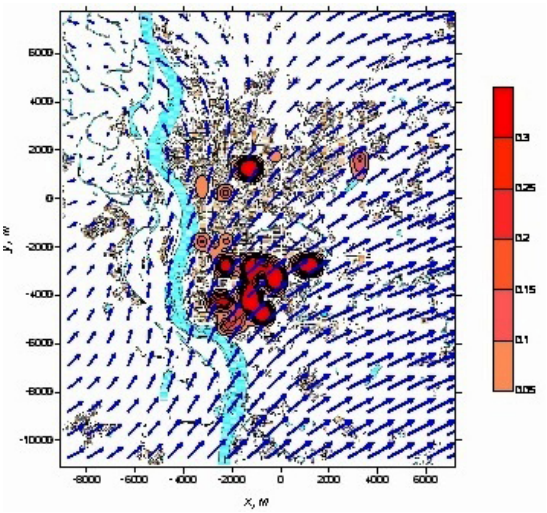


Рис. 4. Результаты решения обратной задачи переноса примеси для определения городских районов-загрязнителей (→ – векторное поле скорости)

В целом предсказанная численно картина соответствует расположению основных городских районов, ответственных за выброс в атмосферу CO. Кроме того, предложенная в работе методика позволяет также достаточно надежно оценить за короткий период времени и интенсивность таких выбросов.

Литература

1. **Марчук Г.И.** Математическое моделирование в проблеме окружающей среды. – М.: Наука, 1982. – 315 с.
2. **Старченко А.В., Беликов Д.А.** Численная модель для оперативного контроля уровня загрязнения городского воздуха // Оптика атмосферы и океана. – 2003. – №7. – С. 657–665.
3. **Panasenko E.A., Starchenko A.V.** Parallel Algorithms for Solution of Air Pollution Inverse Problems // Second Russia – Taiwan Symposium, MTPP 2010: Revised Selected Papers / C.H. Hsu and V. Malyshkin (Eds.). –Berlin; Heidelberg: Springer-Verlag, 2010. – P. 251–259.
4. **Старченко А.В.** Моделирование переноса примеси в однородном пограничном слое // Труды Международной конференции ENVIROMIS2000. – Томск: ЦНТИ, 2001. – С. 77–84.

О применении системы компьютерной алгебры в параллельных средах для решения одной задачи теории групп*

А.И. Макосий, А.В. Тимофеенко
Хакасский государственный университет
Красноярский педагогический университет

На примере задачи поиска порождающего множества инволюций с определенными условиями рассматриваются вопросы применения параллельных вычислений в теории групп.

Немногим более пятидесяти лет назад в исследовании по теории групп были использованы компьютеры. С этого момента, когда впервые была продемонстрирована мощь компьютеров в алгебраических исследованиях, компьютерная индустрия существенно поменяла свой облик и возможности. Наряду с метаморфозой в технологии, параллельно в алгебре, произошла разительная перемена в способах и методах исследования. Сегодня компьютерные вычисления не только облегчают математическое открытие, они могут являться неотъемлемой частью доказательства теорем.

И если вначале опыт применения вычислений в каждом случае был уникальным, то сейчас этот процесс «индустриализирован» компьютерными системами *GAP* и *MAGMA*. Их широкое использование как инструментов исследования алгебраических и комбинаторных структур, в свою очередь, поставило множество алгоритмических и иных вопросов, повлекших дальнейшее развитие самой теории групп. Очень важным и интересным аспектом компьютерных доказательств является то, что после получения результата часто его можно проверить без использования компьютеров или переформулировать результат, не используя при его доказательстве вычисления или минимизируя их использование.

Рассмотренная ниже задача может быть отнесена к классу комбинаторных задач. За редким исключением перебор (порождение) элементов группы и их опробование (тестирование на обладание нужным свойством) является единственным способом решения такого рода задач. Чаще всего алгоритмы решения имеют экспоненциальную сложность и требуют, как правило, больших вычислительных ресур-

* Работа поддержана грантами РФФИ № 09-01-00395, 10-01-00509, грантом Красноярского госпедуниверситета № 16-11-2/МП.

сов. Поэтому даже в параллельном варианте они имеют предел для своего применения. Тем не менее «параллелизация» вычислений приводит к существенному уменьшению времени расчетов.

Модели групп и все вычисления с ними были реализованы с использованием системы компьютерной алгебры *GAP* [1]. Вычисления проводились на суперкомпьютере Межведомственного суперкомпьютерного центра, а также вычислительном кластере Красноярского государственного педагогического университета и кластере Института вычислительного моделирования СО РАН (г. Красноярск).

Предварительные сведения

Напомним определения некоторых основных используемых понятий из теории групп. Недостающие сведения могут быть найдены в учебнике [3].

Пусть G – произвольная группа. Элементы $a,b \in G$ называются сопряженными в G, если $a=g^{-1}bg$ для некоторого элемента $g \in G$. В этом случае также используется обозначение $a=b^g$. Множество $h^G = \{h^g \mid g \in G\}$ называется классом сопряженных с h в группе G элементов. Если H – подгруппа группы G, то множество $g^{-1}Hg$ называется *сопряжением* подгруппы H с помощью элемента g и обозначается как H^g.

Подгруппа H удовлетворяющая условию $H^g \subseteq H$ для любого элемента $g \in G$ называется *нормальной подгруппой* (нормальным делителем) группы G, и записывается как $H \lhd G$. Группа, не содержащая нормальных подгрупп, называется *простой*. Исключительная роль конечных простых групп объясняется тем, что из них может быть построена любая конечная группа.

Пересечение любого количества подгрупп вновь является подгруппой. Это позволяет определить подгруппу, порожденную множеством M, как наименьшую подгруппу, содержащую подмножество M, т.е. пересечение всех подгрупп группы G, содержащих множество M. Подгруппа, порожденная множеством M, будет обозначаться как $<M>$. Легко проверить, что $<M>$ является множеством всевозможных произведений элементов из M и обратных к ним. *Инволюцией* группы G называется ее элемент порядка два, т.е. $i \in G, i \neq 1$, такой что $i^2=1$.

Теория конечных групп – старейшая и в то же время наиболее активно развивающаяся ветвь теории групп. Одним из крупнейших результатов явилось завершение классификации конечных простых неабелевых групп, включающей серии групп лиева типа, знакопеременные группы A_n при $n \geq 5$ и 26 спорадических групп. Среди спорадических одна из наибольших групп Baby Monster имеет порядок

$2^{41}\cdot3^{13}\cdot5^{6}\cdot7^{2}\cdot11\cdot13\cdot17\cdot19\cdot23\cdot31\cdot47$, а ее элементы могут быть представлены как матрицы размерности 4370 и уже обычное умножение элементов этой группы представляет собой нетривиальную вычислительную задачу.

При исследовании групп, прежде всего, интересен вопрос о строении (описании) множества групп с некоторыми заданными свойствами. Структурная теория групп в настоящее время представляет одну из самых развитых областей алгебры и математики в целом. Наиболее важные свойства конечных групп стали оформлять в виде сборника – атласа конечных групп [2]. В наш век повсеместного использования Интернета появился его электронный аналог – электронный атлас конечных групп [4].

О порождении групп тремя инволюциями, две из которых перестановочны

Ряд задач теории групп и ее приложений сводятся к проблеме нахождения множества порождающих элементов, удовлетворяющих определенным свойствам. Для конечных простых групп и близких к ним наибольший интерес вызывают порождающие множества минимальной мощности, в которых особую роль играют инволюции.

Всякая конечная простая неабелева группа содержит инволюции и порождается любым классом сопряженных инволюций. Естественно возникает вопрос: каково минимальное число инволюций (необязательно сопряженных), порождающих конечную простую неабелеву группу? К 90-м годам прошлого века стало известно, что тремя инволюциями порождена каждая конечная простая неабелева группа, исключая группу $U_3(3)$ [4].

Несколько лет назад в работах Я.Н. Нужина и В.Д. Мазурова было выяснено, какие конечные простые неабелевы группы порождаются тремя инволюциями, две из которых перестановочны. Такие тройки инволюций, если они существуют в группе, называются (2×2,2) тройками инволюций этой группы. Препринт [5] положил начало циклу работ, посвященных следующему вопросу:

Указать алгоритмы поиска (2×2,2) троек инволюций в конечных простых неабелевых группах и создать электронный атлас таких троек.

Если (i, j, k) – (2×2,2) тройка инволюций конечной простой группы G и $|ij|=2$, $|ik|=p$, $|jk|=q$, то определим множество $C_2(G)$ как множество всех таких упорядоченных пар чисел (p,q). Таким образом, две (2×2,2)

тройки инволюций не различаются, если соответствующие числа p и q равны.

В работе [6] указан алгоритм поиска (2×2,2) троек инволюций в любой конечной (простой) группе, вычислительные возможности которого ограничены только возможностями программного и аппаратного обеспечения. Схема последовательного алгоритма в общем виде может быть описана как процесс формирования массива троек инволюций и цикла перебора-проверки: являются ли они (2×2,2) тройками и формирования выходных данных. Эти процессы условно реализуются процедурой *SubmitTaskInput()* – инициализация данных и организация перебора троек инволюций, *TestTriples()* – проверка на то, что некоторая тройка инволюций является (2×2,2) тройкой и *UpdateData()* – формирование множества $C_2(G)$.

Методы и программное обеспечение распараллеливания расчетов

Для распараллеливания расчетов применялись два подхода. Первый подход базируется на использовании *ParGAP (Parallel GAP)* [6] – сторонней разработки, предназначенной для работы с *GAP* и позволяющей создавать параллельные программы на языке *GAP*. *ParGAP* реализован на подмножестве стандарта *MPI*.

Декомпозиция последовательного алгоритма была осуществлена сочетанием функциональной декомпозиции и декомпозиции данных. Уменьшение объема обмена данными между процессами было получено за счет того, что подчиненному процессу фактически выделялся некоторый пул номеров, по которому он формировал множество троек инволюций группы, и проверялось, являются ли они (2×2,2) тройками, а уже главный процесс решал – пополнять такой тройкой (если ее там еще нет) множество $C_2(G)$ или нет. При этом также была получена полная загрузка узлов во время счета. Для возобновления расчетов была организована обработка прерываний.

Проверка данных была разбита на две части – локальную (предварительную) проверку *TestTriples()* на узлах и окончательную *CheckTaskResult()* на главном процессе. Вначале главный процесс инициализирует глобальные и разделяемые данные. Глобальные данные доступны всем узлам. Они используются в процессе расчетов в режиме «только для чтения». Разделяемые данные доступны процедуре *UpdateData()* в режиме «чтение–запись», в то время как для *SubmitTaskInput(), TestTriples()* и *CheckTaskResult()* в режиме «только для чтения». Разделяемые данные должны быть вначале проиниция-

лизированы на всех процессах, а затем модифицироваться только процедурой *UpdateData()* на главном процессе. Обработка прерываний реализована при помощи стандартного механизма организации контрольных точек. «Зависания» подчиненных процессов обрабатываются внутренними механизмами *ParGAP*. Для старта задачи потребовалась небольшая адаптация скрипта инициализации заданий *mpirun*. «Жизненный цикл» работы алгоритма представлен на рис. 1:

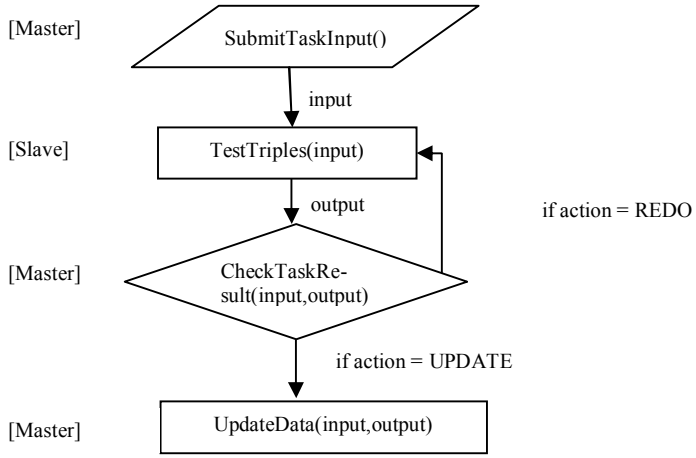


Рис. 1

Второй способ параллелизации основан на возможности прямой декомпозиции исходных данных и выполненения последовательного кода на каждом из узлов. Множества инволюций разбиваются на непересекающиеся множества и для каждого из них выполняется поиск (2×2,2) троек инволюций и построение множества $C_2(G)$. Полученная при этом избыточность расчетов (на разных узлах могут быть получены одинаковые пары чисел (p,q) (порядков неперестановочных инволюций) устраняется проверкой выходных данных после прерывания расчетов и формированием нового множества глобальных данных.

Для запуска *GAP* в параллельной среде была написана на языке *C* программа-стартер. Стартер запускал нужное количество *GAP*-процессов. Каждый процесс использовал свое множество данных и создавал выходные файлы в отдельном каталоге. После прерывания (выполнения) расчетов эти данные модифицировались и формировались единые для всей задачи выходные файлы.

По указанному алгоритму были выполнены расчеты для знакопеременных групп A_{12}–A_{16} и ряда спорадических групп.

Литература

1. **The GAP Group**. The GAP Group, GAP – Groups, Algorithms, and Programming, v. 4.4.12, 2008. http://www.gap-system.org.
2. **Conway J. H., Curtis R. T., Norton S. P. et al.** Atlas of finite groups. – Oxford: Clarendon Press, 1985. – 252 p.
3. **Каргополов М.И., Мерзляков Ю.И.** Основы теории групп. – М.: Наука, 1996.
4. **Wilson R., Parker R.A., Bray J.N.** ATLAS of Group Representations. http://web.mat.bham.ac.uk/atlas/v2.0.
5. **Нужин Я.Н., Тимофенко А.В.** Порождающие тройки инволюций некоторых спорадических групп. Препринт №13–99. – Красноярск: ИВМ СО РАН. 1999. – 20 с.
6. **Макосий А.И.** О порождающих множествах инволюций конечных групп и смежные вопросы. Вычислительный подход. – LAP LAMBERT Academic Publishing GmbH, 2011. – 78 с.
7. **Cooperman G.** Parallel GAP/MPI (ParGAP/MPI), College of Computer Science, Northeastern University, http://www.ccs.neu.edu/home/gene/pargap.html

Реализация некоторых алгоритмов обработки изображений с использованием технологии CUDA на графических устройствах

Н.Н. Богословский
Томский государственный университет

Обработки цифровых изображений в настоящее время – очень актуальная проблема. Часто обрабатывается большой объем цифровых изображений, а многие алгоритмы обработки цифровых изображений являются вычислительно трудоемкими. В работе рассматриваются два алгоритма обработки изображений: алгоритм медианной фильтрации и алгоритм увеличения изображения с использованием бикубической интерполяции. Алгоритмы реализованы с использованием технологии NVIDIA CUDA на графических процессорах.

Введение

При работе с цифровыми изображениями часто возникает задача удаления шумов, так как цифровое изображение вместе с полезным сигналом содержит, как правило, различные шумы. Один из видов шумов – это импульсный шум. Под импульсным шумом понимается искажение сигналов большими импульсными выбросами произвольных значений и малой длительности. Импульсные помехи в цифровом сигнале могут возникать из-за неправильного или ложного срабатывания одного пикселя цифровой матрицы, поврежденного участка памяти (при отсутствии контроля ошибок) или помех в каналах передачи сообщений [1]. Два основных типа импульсных шумов – это шумы типа «соль-перец» и случайные шумы. На изображении с шумом типа «соль-перец» поврежденные пиксели могут принимать только максимальное и минимальное значение в заданном динамическом диапазоне[1].

Существует множество методов удаления шумов на цифровых изображениях, обзор таких методов, например, приведен в работе [2]. Наиболее популярный нелинейный фильтр удаления импульсного шу-

[1]Динамический диапазон – характеристика устройства или системы, предназначенной для преобразования, передачи или хранения некой величины (мощности, силы, напряжения, звукового давления и т.д.), представляющая логарифм отношения максимального и минимального возможных значений величины входного параметра устройства (системы).

ма – это медианный фильтр, так как он достаточно хорошо удаляет шумы [1] и обладает хорошей вычислительной эффективностью [3].

Часто при работе с цифровыми изображениями возникает задача масштабирования изображения. Наиболее сложной при этом является задача увеличения изображения, при решении которой необходимо восстанавливать и дополнять промежуточные пиксели в изображении. Один из способов основан на использовании бикубической интерполяции.

Хотя все эти методы обладают достаточно большой вычислительной эффективностью, тем не менее остро стоит вопрос о повышении быстродействия, особенно если необходимо применять данные методы в системах реального времени, таких как дорожно-транспортные системы наблюдений, охранные системы наблюдений и др.

Одним из методов повышения скорости работы программ и алгоритмов является использование технологии GPGPU (General-purpose graphics processing units или «GPU общего назначения») [4]. Данная технология была предложена в качестве инструмента для параллельных вычислений компанией NVIDIA. В настоящее время она рассматривается в качестве эффективного средства разработки приложений для решения различных задач научно-исследовательского характера. Одна из реализаций GPGPU представлена технологией CUDA [5, 6]. CUDA (Compute Unified Device Architecture) – архитектура и программная модель для параллельных вычислений, которая позволяет производить расчёты на GPU NVIDIA со значительным ускорением [5, 6].

Медианный фильтр

Пусть $x_{i,j}$ для $(i,j) \in A = \{1,\ldots,M\} \times \{1,\ldots,N\}$ – значение уровня серого цвета в исходном цифровом изображении X размера $M \times N$ в пикселе с номером (i,j). $[s_{\min}, s_{\max}]$ – это динамический диапазон, т.е. $s_{\min} \le x_{i,j} \le s_{\max}$ для всех .

Обозначим через Y цифровое изображение с импульсным шумом. В классической модели импульсного шума типа «соль-перец» значение уровня серого цвета определяется следующим образом:

$$y_{i,j} = \begin{cases} s_{\min}, \text{ с вероятностью } p, \\ s_{\max}, \text{ с вероятностью } q, \\ x_{i,j}, \text{ с вероятностью } 1-p-q, \end{cases}$$

где $r = p + q$ определяет уровень шума. $(i, j) \in A$.

Пусть $S_{i,j}^{w}$ – это окно размером $w \times w$ с цетром в точке (i, j), т.е.

$$S_{i,j}^{w} = \{(k,l) : |k - i| \le w, |l - j| \le w\}.$$

Тогда алгоритм медианного фильтра можно представить следующим способом:

1) Фиксируем точку $(i, j) \in A$.
2) Формируем последовательность $K_{i,j} = \{x_{k,l} \in Y; (k,l) \in S_{i,j}^{w}\}$, состоящую из значений уровня серого цвета изображения Y.
3) Находим медиану[1] $x_{i,j}^{med}$ последовательности $K_{i,j}$ и полагаем значение в точке (i, j) фильтрованного изображения, равное $x_{i,j}^{med}$.
4) Если не прошли все точки изображения, то переходим к пункту 1.

Так как выборка обрабатываемых значений, попадающих в окно $S_{i,j}^{w}$, и нахождение медианы для всех точек изображения проводятся независимо, то такую фильтрацию можно проводить параллельно для всех точек.

При реализации параллельной обработки данных на GPU необходимо написать функцию, называемую ядром, которая будет выполняться всеми нитями параллельно. При реализации параллельного алгоритма медианной фильтрации в ядре были реализованы шаги 2 и 3 описанного выше алгоритма и, соответственно, каждая нить обрабатывала свой собственный пиксель исходного изображения с его окрестностью. На видеокартах используется несколько видов различной памяти, поэтому алгоритм был реализован с использованием глобальной и текстурной памяти. Результаты вычислений и времени выполнения на GPU для этих двух реализаций приводятся ниже.

Масштабирование изображения с использованием бикубической интерполяции

Пусть $x_{i,j}$ для $(i, j) \in A = \{1, \ldots, M\} \times \{1, \ldots, N\}$ – значение уровня серого цвета в исходном цифровом изображении X размера $M \times N$ в

[1]Медиана – одна из характеристик среднего для числового набора. По определению это такое число, что количество различных чисел набора, меньших данного, равно количеству различных чисел набора, больших данного.

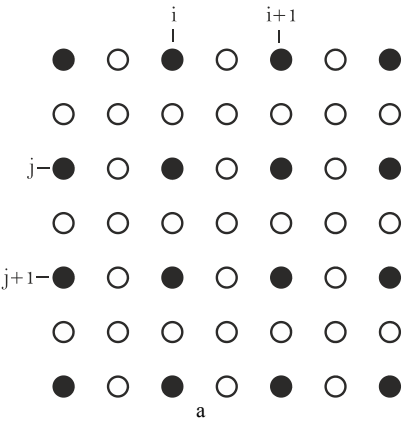
пикселе с номером (i, j). Допустим, необходимо увеличить изображение в два раза. Тогда, как показано на рис. 1, необходимо дополнительно рассчитать значения уровня серого цвета в точках, обозначенных незакрашенными кружками, расположенных между точками исходного изображения, обозначенных черными кружками.

Для расчета неизвестных значений использовалась бикубическая интерполяция. Сначала рассчитывались неизвестные значения по горизонтали с использованием кубической интерполяции по формуле (1). Затем с использованием аналогичной формулы проводились расчеты по вертикали. При расчетах в граничных точках использовались фиктивные точки.

$$p(t) = \frac{1}{2}\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}\begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix}\begin{bmatrix} x_{i-1,j} \\ x_{i,j} \\ x_{i+1,j} \\ x_{i+2,j} \end{bmatrix}. \qquad (1)$$

Расчет в разных точках при горизонтальной и вертикальной интерполяции проводится независимо, поэтому можно эффективно реализовать параллельную интерполяцию с использованием GPU. Сначала проводилась параллельно горизонтальная интерполяция, а после того как были вычислены все точки в нечетных строках – параллельно вертикальная интерполяция.

Для повышения эффективности параллельного алгоритма применялась поэтапная загрузка картинки из памяти CPU на GPU. Для этого изображение разбивалось на m полос, как показано на рис. 2. При таком разбиении горизонтальную интерполяцию можно осуществлять в разных полосах независимо. Для вертикальной же интерполяции, например в полосе 2, необходимо, чтобы уже была проведена горизонтальная интерполяция в 1-й и 3-й полосе, для того чтобы можно было провести интерполяцию в граничных узлах. В цикле, который выполнялся на CPU, было реализовано поэтапное копирование изображения с CPU на GPU, запуск ядер вертикальной и горизонтальной интерполяций на GPU осуществлялся в различных потоках в конкурирующем режиме с синхронизацией для правильного выполнения вертикальной интерполяции.



а

б

Рис. 1. Увеличение изображения в два раза с использованием бикубической интерполяции:

а — черными показаны точки, которые необходимо вычислить;

б — иллюстрация кубической интерполяции по 4 точкам

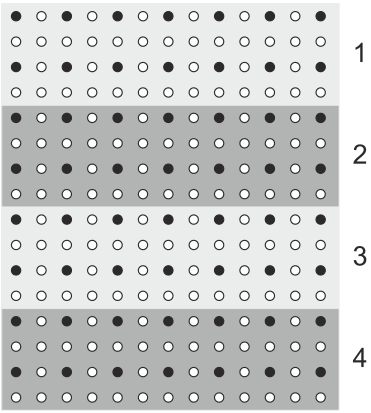


Рис. 2. Разбиение исходного изображения на несколько полос для поэтапного копирования и выполнения ядер

Результаты

Для проверки правильности параллельного алгоритма и программы, реализующей медианную фильтрацию, была сгенерирована тестовая картинка с шумом типа «соль-перец». На рис. 3 слева представлено изображение с шумом, а справа – отфильтрованное изображение.

Расчеты с использованием параллельного алгоритма проводились на компьютере под управлением ОС Windows 7 с видеокартой NVIDIA GTS 450 и на компьютере под управлением ОС Linux с видеокартами Tesla 1060 и Tesla 2070. В табл. 1 приведены результаты расчета на различных видеокартах и для изображений различной размерности.

а б

Рис. 3. Изображение с шумом типа «соль-перец» – *а*; *б* – отфильтрованное изображение с использованием параллельного медианного фильтра

Как видно из таблицы, максимальное ускорение при использовании окна фильтрации $w = 3$ получилось при расчетах с использованием текстурной памяти на видеокарте Tesla 2070. При использовании окна фильтрации размера 3 отличие в ускорении в зависимости от используемой памяти незначительно, более того, на видеокарте GTS 450 при использовании текстурной памяти происходит увеличение времени счета и как следствие ускорение падает.

Таблица 1. Время расчета и ускорение медианного алгоритма подавления шума

	CPU	GTS 450		Tesla 1060		Tesla 2070	
Размерность изображения (тип памяти)	Время на компьютере с Windows/Linux	Время, ms	Ускорение	Время, ms	Ускорение	Время, ms	Ускорение
Размер окна фильтрации $w = 3$							
512x512 (global)	95.71/ 94.98	3.65	26.2	3.04	31.24	2.43	39.08
512x512 (texture)	95.71/ 94.98	5.87	16.3	2.87	33.09	2.45	39.09
1600x1200 (global)	640.51/5 61.69	25.35	25.26	18.24	30.79	8.82	63.68
1600x1200 (texture)	640.51/5 61.69	28.00	22.86	18.92	29.68	8.69	64.63
Размер окна фильтрации $w = 5$							
512x512 (global)	506.44/ 364.26	4.32	117.04	3.65	99.79	2.11	172.31
512x512 (texture)	506.44/ 364.26	6.13	82.60	2.96	123.06	1.81	200.36
1600x1200 (global)	3395/ 2295	27.85	121.87	23.33	98.37	11.31	202.8
1600x1200 (texture)	3395/ 2295	26.06	130.24	17.59	130.47	8.76	261.9

Таблица 2. Время расчета и ускорение алгоритма увеличения изображения в два раза с использованием бикубической интерполяции

	CPU	GTS 450		Tesla 1060		Tesla 2070	
Размерность изображения (тип памяти)	Время на компьютере с Windows/Linux	Время, ms	Ускорение	Время, ms	Ускорение	Время, ms	Ускорение
Размер окна фильтрации $w = 3$							
512x512	35,04/ 29,49	1,32	26,54	1,08	27,23	1,04	28,35
1024x1024	211,25/ 191,09	6,74	31,34	2,05	44,31	3,16	60,47
2048x2048	900,38/ 790,13	25,43	35,40	13,48	66,79	8,72	90,61
4096x4096	4256,87/ 3904.92	104,38	40,78	60,68	70,15	33,78	115,59

При использовании окна фильтрации размера 5 для всех видеокарт ускорение дополнительно увеличивается в 4–5 раз, так как вычислительная сложность алгоритма повышается и более выражено проявляется эффект многопоточного выполнения программы на GPU.

В табл. 2 приведены время и ускорение для алгоритма увеличения изображения в два раза с использованием бикубической интерполяции. Наибольшего ускорения удалось достичь на самой производительной видеокарте Tesla 2070, ускорение составило 115.59. При использовании стандартной видеокарты GTS 450 тоже удалось достичь хороших результатов. Наибольшее ускорение на этой видеокарте удалось получить при увеличении изображения размерности 4096x4096.

Заключение

В ходе выполнения данной работы были реализованы два алгоритма с использованием технологии многопоточного программирования CUDA для графических видеокарт. Показано, что реализация алгоритмов для графических видеокарт и их использование оправданы, так как позволяют получить значительное ускорение времени выполнения расчетов. Использование специализированных графических процессоров, адаптированных для вычислительных алгоритмов, таких как Tesla 1060 и Tesla 2070, позволяет получить существенное ускорение по сравнению с последовательной программой.

Литература

1. **Bovik A.** Handbook of Image and Video Processing. – Academic Press, 2000.
2. **Astola J., Kuosmanen P.** Fundamentals of Nonlinear Digital Filtering. – Boca Raton, CRC, 1997.
3. **Huang T.S., Yang G.J., Tang G.Y.** Fast two-dimensional median filtering algorithm // IEEE Transactions on Acoustics, Speech, and Signal Processing. – 1979. – Vol. 1. – P. 13–18.
4. GPGPU.org: General-Purpose computation on Graphics Processing Units [Электронный ресурс]. – Электрон. дан. – URL: http://gpgpu.org/ (дата обращения: 25.10.2011).
5. CUDA [Электронный ресурс]. – Электрон. дан. – URL: http://www.nvidia.com/object/cuda_home_new.html (дата обращения: 25.10.2011).
6. CUDA: GPGPU.org [Электронный ресурс]. – Электрон. дан. – URL: http://gpgpu.org/developer/cuda (дата обращения: 25.10.2011).

Численное решение краевых задач в областях сложной геометрии

И.Е. Смирнов
Томский государственный университет

Рассматривается дискретизация эллиптического уравнения на неструктурированной сетке в рамках метода конечного объема применительно к задаче Дирихле. Расчетная сетка считается заданной, в частности, построенной при помощи сеточного генератора Gmsh с открытым кодом, доступном в Интернете. Анализируются результаты использованных численных методов, применяемых для решения задачи.

Введение

На данный момент большинство методов, разработанных для численного решения дифференциальных уравнений в частных производных, являются сеточными. Важным местом в сеточных методах является построение расчетной сетки. Создание расчетной сетки – это деление интересующей области на маленькие ячейки, в которых нужно найти неизвестные. Развитие численных методов и компьютерных технологий помогает вычислять с помощью компьютеров все более и более сложные задачи.

Обычно реальные инженерные задачи требуют расчетов в областях сложной геометрической формы, что не всегда позволяет построить единую расчетную сетку для всей рабочей области. При построении и использовании неструктурированных сеток современные сеточные генераторы позволяют создать расчетную сетку для разнообразно сложных геометрических объектов за приемлемое время.

В данной работе рассматривается подход к дискретизации уравнения Лапласа на неструктурированных сетках в рамках метода конечного объема применительно к задаче Дирихле. Расчетная сетка считается заданной, в частности, построенной при помощи сеточного генератора Gmsh с открытым кодом, доступным в Интернете. Анализируются результаты использованных численных методов, применяемых для решения задачи.

Задача Дирихле для уравнения Лапласа

Требуется определить распределение функции $u(x,y)$, удовлетворяющей уравнению Лапласа:

$$\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}=0,\ (x,y)\in\left\{(x,y)\in\Re^2\left|\left(\frac{x}{2}\right)^2+y^2<1\right.\right\} \qquad (1)$$

в овале $\left(\frac{x}{2}\right)^2+y^2\le 1$ при следующих граничных условиях:

$$u(x,y)\Big|_{\left(\frac{x}{2}\right)^2+y^2=1}=\varphi(x,y)\Big|_{\left(\frac{x}{2}\right)^2+y^2=1}. \qquad (2)$$

Выбор и построение сетки. Построение разностной задачи

Для построения сетки в исследуемой области используется некоммерческий пакет GMSH [1], разработанный для построения параметризованных твердотельных моделей, получения сеток, а также для визуализации результатов расчета. Сетка была построена с помощью триангуляции Делоне.

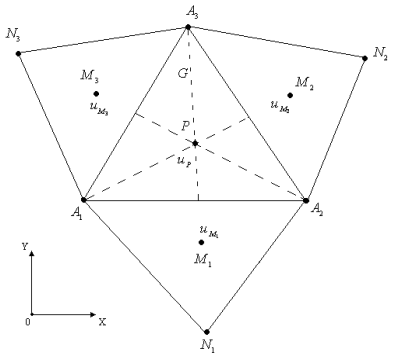


Рис. 1. Фрагмент неструктурированной сетки

Для построения разностной задачи проинтегрируем уравнение (1) по конечному объему *G,* изображенному на рис. 1, где P, M_1, M_2, M_3 – центры тяжести соответствующих треугольников; u_P – значение

скалярной величины u в точке P, A_iA_j – ребра конечного объема G:

$$\iint\limits_G \left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} \right) dG = 0. \tag{3}$$

Используя формулу Грина, получим следующее уравнение:

$$\iint\limits_G \left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} \right) dG = \oint\limits_{\partial G} (\vec{n}, gradu) dl = \oint\limits_{\partial G} \frac{\partial u}{\partial n} dl = \tag{4}$$

где $\vec{n}(n_x, n_y)$ – единичный вектор внешней нормали к границе треугольной ячейки ∂G.

$$\int\limits_{A_1}^{A_2} \frac{\partial u}{\partial n_{12}} dl + \int\limits_{A_2}^{A_3} \frac{\partial u}{\partial n_{23}} dl + \int\limits_{A_3}^{A_1} \frac{\partial u}{\partial n_{31}} dl = 0, \tag{5}$$

где $\vec{n}_{ij}$ – единичный вектор внешней нормали к соответствующему ребру конечного объема G ($i, j = 1,2,3, i \neq j$).

По формуле средних прямоугольников

$$\begin{aligned} &\int\limits_{A_1}^{A_2} \frac{\partial u}{\partial n_{12}} dl + \int\limits_{A_2}^{A_3} \frac{\partial u}{\partial n_{23}} dl + \int\limits_{A_3}^{A_1} \frac{\partial u}{\partial n_{31}} dl \approx \\ &\approx \left(\frac{\partial u}{\partial n_{12}} \right)_{m_{12}} \Delta l_{12} + \left(\frac{\partial u}{\partial n_{23}} \right)_{m_{23}} \Delta l_{23} + \left(\frac{\partial u}{\partial n_{31}} \right)_{m_{31}} \Delta l_{31} = 0, \end{aligned} \tag{6}$$

где $\left(\frac{\partial u}{\partial n_{ij}} \right)_{m_{ij}}$ – производная по направлению внешней нормали к конечному объему G на ребре A_iA_j; Δl_{ij} – длина ребра A_iA_j; m_{ij} – середина ребра (рис. 2).

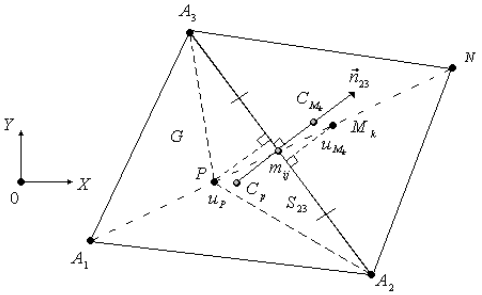


Рис. 2. Фрагмент сетки

Справедливо следующее представление функции u в точках C_p и C_{M_k} с использованием формулы Тейлора:

$$u(C_P) = u(m_{ij}) - \left(\frac{\partial u}{\partial n_{ij}}\right)_{m_{ij}} \Delta n_{ij} + \mathrm{O}(\Delta n_{ij}^2), \tag{7}$$

$$u(C_{M_k}) = u(m_{ij}) + \left(\frac{\partial u}{\partial n_{ij}}\right)_{m_{ij}} \Delta n_{ij} + \mathrm{O}(\Delta n_{ij}^2). \tag{8}$$

Используя выражения (7) и (8), получим:

$$\left(\frac{\partial u}{\partial n_{ij}}\right)_{m_{ij}} = \frac{u(C_{M_k}) - u(C_P)}{2\Delta n_{ij}} + \mathrm{O}(\Delta n_{ij}) \cdot \tag{9}$$

Предположим, что $u(C_P) \approx u(P), u(C_{M_k}) \approx u(M_k)$, и, используя (9), получим:

$$\left(\frac{u(M_1) - u(P)}{2\Delta n_{12}}\right)\Delta l_{12} + \left(\frac{u(M_2) - u(P)}{2\Delta n_{23}}\right)\Delta l_{23} + \left(\frac{u(M_3) - u(P)}{2\Delta n_{31}}\right)\Delta l_{31} = 0 \tag{10}$$

или

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

При вычислении производных на примыкающем к границе ребре используются односторонние разности. В итоге получается разностная схема, аппроксимирующая дифференциальную задачу (1)–(2) с первым порядком. С помощью принципа максимума и мажорирующей

функции Гершгорина доказана устойчивость построенной разностной схемы.

Применение численных методов

Для численного решения задачи Дирихле для уравнения Лапласа на неструктурированных сетках использовались: метод Якоби, метод сопряженных градиентов (CG), стабилизирующий метод бисопряженных градиентов (BiCGStab).

При различных вариациях вычислительных сеток получились результаты, представленные в таблице (погрешность вычислений равна 10^{-5}).

Результаты (в итерациях)

Метод	Кол-во ячеек сетки		
	13	1234	4241
Якоби	43	1693	2846
CG	8	133	585
BiCGStab	5	86	398

На рис.3 представлен график сходимости итерационного процесса.

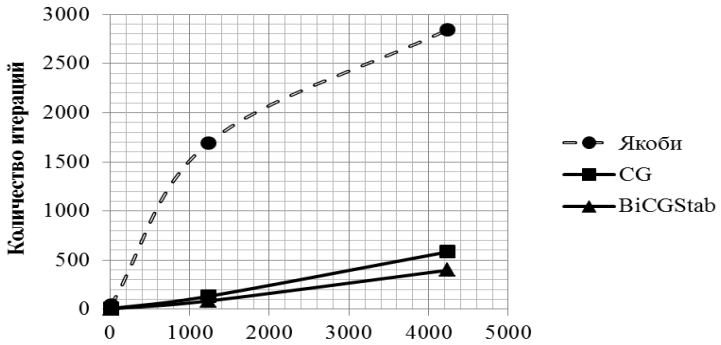


Рис. 3. Сходимость итерационного процесса. По оси *OX* указаны номера сеток: *1* – 13 ячеек; *2* – 1234 ячейки; *3* – 4241 ячейка

С ростом размера расчетной сетки наблюдается увеличение числа обусловленности матрицы, полученной в результате дискретизации исходной задачи. При расчетной сетке в 13 ячеек число обусловленности составляет 18, при сетке 1234 ячейки – 565, а для сетки в 4241 ячейки – 1048. Следовательно, с последующим ростом количества расчетных ячеек потребуется применение методов решения линейных систем с предобуславливанием.

Заключение

В работе рассмотрен способ дискретизации уравнения Лапласа на неструктурированной сетке в рамках метода конечного объема. В качестве конечного объема выбрана ячейка треугольной формы, построенная с помощью триангуляции Делоне, которая позволяет строить расчетные сетки любой сложности. Использование стабилизирующего метода бисопряженных градиентов для решения сеточных уравнений обеспечивает завершение итерационного процесса за меньшее количество итераций.

Литература

1. **Geuzaine C., Remacle J-F**. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities [Электронный ресурс]. – Режим доступа http://geuz.org/gmsh
2. **Wenneker I.** Computation of flows using unstructured staggered grids // Dissertation at Delft University of Technology. 2002. – 208 p.
3. **Фирсов Д.К.** Метод контрольного объема на неструктурированной сетке в вычислительной механике. – Томск: Изд-во ТГУ, 2007. – 46 с.

Научное издание

Шестая Сибирская конференция
по параллельным и высокопроизводительным вычислениям

Томск, 15−17 ноября 2011 года

Редактор **В.Г. Лихачёва**

Компьютерная верстка **Д.В. Деги**

Подписано в печать 18.04.12 г.

Формат 60x84 $^{1}/_{16}$. Бумага офсетная №1.

Печать офсетная. Гарнитура «Times».

Печ. л. 11,9; уч.-изд. л. 11,0; усл. печ. л. 11,2.

Заказ Тираж 100 экз.

ОАО «Издательство ТГУ», 634029, г. Томск, ул. Никитина, 4

Типография «Иван Федоров», 634003, г. Томск, Октябрьский взвоз, 3