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

Высокопроизводительные параллельные вычисления на кластерных системах (HPC–2010)

Материалы X Международной конференции

(г. Пермь, 1–3 ноября 2010 г.)

В двух томах

Том 1

Издательство
Пермского государственного технического университета
2010

УДК 004.382.2:51
ББК 32.973
В93

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

Финансовая поддержка конференции

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

Информационная поддержка конференции

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

ISBN 978-5-398-00503-5

1–3 ноября 2010 года под эгидой Суперкомпьютерного консорциума университетов России на базе Пермского государственного технического университета при непосредственном участии Нижегородского государственного университета им. Н.И. Лобачевского и поддержке Российского фонда фундаментальных исследований, Министерства промышленности, инноваций и науки Пермского края, корпораций «AMD» и «IBM», компаний «Т-Платформы», «Делкам-Урал», «Тесис», научно-технической программы Союзного государства «РСК СКИФ», газеты научного сообщества «Поиск», информационно-аналитического центра Parallel.Ru информационно-аналитического журнала «Rational Enterprise Management» и Молодежного парламента при Законодательном собрании Пермского края проведена X Международная конференция «Высокопроизводительные параллельные вычисления на кластерных системах».

Работа конференции была организована по следующим направлениям:

- принципы построения кластерных систем и методы управления параллельными вычислениями;
- параллельные методы и программные системы решения сложных вычислительных задач;
- программные среды, средства и инструменты для разработки параллельных программ;
- технологии распределенных вычислений и GRID-технологии;
- научная визуализация и машинная графика для высокопроизводительных вычислений;
- проблемы подготовки специалистов в области параллельных вычислений;
- решение прикладных задач на кластерных системах.

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

Проведение конференции поддержано грантом РФФИ № 10-07-06092-г.

ПЛЕНАРНЫЕ ДОКЛАДЫ

В.Ю. Петров, Н.А. Шевелев, В.Я. Модорский

Пермский государственный технический университет

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ НА ВЫСОКОПРОИЗВОДИТЕЛЬНОМ ВЫЧИСЛИТЕЛЬНОМ КОМПЛЕКСЕ ПГТУ

Высокопроизводительный вычислительный комплекс (ВВК) ПГТУ вступил в строй в рамках реализации приоритетного национального проекта «Образование» и инновационной образовательной программы Пермского государственного технического университета в 2008 году. На момент пуска кластер занял 13-е место в рейтинге самых мощных суперкомпьютеров СНГ ТОП-50. Пермский ВВК имеет производительность 4,096 TFlops и включает 64 вычислительных узла, 128 четырехядерных процессоров AMD Opteron «Barselona-3» с объемом системы хранения информации 12 Тбайт и объемом оперативной памяти 512 Гб. При построении кластера компанией «DATA Технологии» (г. Москва) была реализована разноплатформенность и применены программируемые ускорители Celoxica, позволяющие поднять производительность вычислений на определенном классе задач при достаточной квалификации пользователей на 1–2 порядка по сравнению с центральным процессором. Отметим, что на момент пуска это был первый в мире кластер на процессорах AMD «Барселона-3». Особое внимание было уделено использованию технологий AMD, обеспечивающих максимальное соотношение «производительность/энергопотребление».

Машинный зал ВВК оснащен современным оборудованием для поддержания безотказности и корректного автоматического завершения работы всех приложений в аварийных ситуациях. Помещения под кластер выделены с перспективой увеличения вычислительной мощности.

С момента пуска в эксплуатацию прошло два года. Можно подвести первые итоги, провести анализ проблем, возникших в процессе освоения и эксплуатации уникальной техники, обозначить планы на будущее.

Работа с кластером разворачивалась по двум направлениям. С одной стороны, для пользователей было приобретено современное лицензионное программное обеспечение, автоматически распараллеливающее процесс расчета и позволяющее работать с многопроцессорным комплексом без специальных знаний в области распределенных вычислений. Это пакеты инженерного анализа, «тяжелый САПР», такие как ANSYS, ABAQUS, LS Dyna, Star CD, Flow Vision и др. Эти пакеты позволяют проводить уникальные междисциплинарные расчеты для сложных технических объектов. При этом решалась задача широкого привлечения инженеров-расчетчиков и исследователей к работе на кластере. С этой же целью для исключения известного психологического барьера, первоначально возникающего при использовании суперкомпьютеров, было принято решение о работе с инженерными пакетами на привычной платформе от Microsoft – Windows Compute Cluster Server. Некоторое снижение производительности позволяет преодолеть покупка операционной системы Windows HPC Server 2008. С другой стороны, для пользователей, разрабатывающих собственные параллельные приложения, были созданы условия для работы в специализированной среде Visual Studio Professional 2008. Создание виртуальных машин позволило преодолеть конфликты «тяжелых пакетов» и «экспериментаторов-программистов». Таким образом, особое внимание было уделено созданию условий, позволяющих превратить пермский ВВК в центр коллективного пользования. Активная реализация этой идеи позволила за короткий срок создать свыше ста удаленных точек для работы на кластере. Каждому пользователю при регистрации выдаются индивидуальные пароли и логины для реализации защищенного VPN-соединения.

Приобретение уникального и дорогостоящего оборудования позволило вывести теоретическую часть исследовательских проектов на совершенно новый уровень. Использование много-

процессорной техники повысило качество подготовки аспирантов, докторантов, научных работников, преподавателей, специалистов предприятий и организаций, вышускников по направлениям аэрокосмического профиля, информационных технологий, прикладной математики и информатики и др. Активные пользователи кластера защитили за это время несколько кандидатских и докторскую диссертации.

Вместе с тем для развития идей многопроцессорных расчетов, привлечения новых пользователей, преодоления имеющей место некоторой «изолированности» этой области знаний были предприняты активные шаги по организации системы обучения распределенным вычислениям. Утверждено свыше десятка программ дополнительного профессионального образования, в том числе «Многопроцессорные вычислительные системы и параллельное программирование», «Эксплуатация высокопроизводительных вычислительных систем и многопроцессорных программных комплексов», программы для освоения распределенных вычислений при проведении прочностных расчетов и решения задач вычислительной гидро- и газодинамики, связанных задач аэротермоупругости и обработки трехмерных изображений. Объем программ от 72 до 500 часов. По результатам обучения слушателям выдается документ государственного образца.

На первых этапах было организовано обучение профессорско-преподавательского состава, аспирантов и докторантов ПГТУ, затем – студентов. В рамках дополнительного образования обучение прошли около 200 слушателей. Часть выпускников активно включается в решение задач, проблематика которых продиктована требованиями промышленных предприятий Пермского региона. В чтении лекций и проведении практических занятий со слушателями приняли участие специалисты Нижегородского государственного университета им. Н.Н. Лобачевского (г. Н. Новгород), Саровского инженерного центра (г. Саров), ОАО «Авиадвигатель» (г. Пермь), таких специализированных организаций, как «DATA Технологии» (г. Москва), «Тесис» (г. Москва, г. Н. Новгород), «Делкам-Урал» (г. Екатеринбург) и др. Пермские специа-

листы прошли обучение по программе повышения квалификации по приоритетным направлениям профессорско-преподавательского состава вузов, находящихся в ведении Федерального агентства по образованию, «Высокопроизводительные вычисления» (ННГУ, г. Н. Новгород). При этом специалисты ПГТУ активно развивали контакты с такими ведущими центрами распределенных вычислений в России, как НИВЦ МГУ, ННГУ им. Лобачевского, ЮФУ, ЮУрГУ, УГАТУ и др. О результатах работ в области распределенных вычислений специалисты ПГТУ неоднократно докладывали на научных конференциях, таких как научная конференция «Численные методы, параллельные вычисления и информационные технологии» (ИВМ РАН, НИВЦ МГУ, г. Москва, 13 февраля 2008 г.), семинар «Высокопроизводительные вычисления и параллельное программирование» (Intel и Microsoft, г. Москва, 26 марта 2008 г.), VI открытая Всероссийская конференция «Преподавание информационных технологий в Российской Федерации» (ННГУ, г. Нижний Новгород, 12–13 мая 2008 г.), совещание по результатам реализации инновационных образовательных программ ПНПО в части формирования актуальных компетенций в области информационно-коммуникационных технологий, в том числе с использованием установленных суперкомпьютеров, и развитие межвузовского взаимодействия (ННГУ, г. Нижний Новгород, 5 сентября 2008 г.), всероссийская научно-техническая конференция «Ракетно-космические двигательные установки» (МГТУ имени Н.Э. Баумана, г. Москва, 16–17 октября 2008 г.), всероссийская конференция «Технологии Microsoft в теории и практике программирования» для студентов, аспирантов и молодых ученых Российской Федерации (ННГУ им. Н.И. Лобачевского, г. Нижний Новгород, 11–12 марта 2009 г.), международная научно-практическая конференция «Инженерные системы–2008» (компания «Тесис» и РУДН, г. Москва, 6–9 апреля 2008 г.), всероссийская суперкомпьютерная конференция «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность» (г. Новороссийск, 21–26 сентября

2009 г.), XI Международный семинар «Супервычисления и математическое моделирование» (ВНИИЭФ, г. Саров, 5–9 октября 2009 г.) и др.

После запуска кластера, с 2008 года, успешно функционирует секция «Системы автоматизированного проектирования и высокопроизводительные вычислительные комплексы» в рамках ежегодной всероссийской научно-технической конференции «Аэрокосмическая техника, высокие технологии и инновации» (ПГТУ, Пермь, 2008, 2009). В работе секции принимают участие специалисты из г. Москвы, Санкт-Петербурга, Н. Новгорода, Екатеринбурга, Челябинска, Уфы, Сарова и, конечно, Перми.

Приоритетными направлениями использования ВВК в Пермском крае являются следующие. Без применения передовых информационных технологий немыслимо ускоренное проектирование современных газотурбинных двигателей и энергетических установок на пермских предприятиях ОАО «Авиадвигатель» и «Протон-ПМ». Разработка конкурентоспособной продукции предполагает большой объем инженерных и научных расчетов по направлениям гидро- и газодинамики, прочности и акустики. Проведение вычислительного эксперимента на суперкомпьютере призвано сократить количество дорогостоящих стендовых испытаний. ОАО «Авиадвигатель» выступил с софинансированием научно-образовательного комплекса «Распределенные вычисления». Разработка нанотехнологий и композитных материалов совместно с предприятиями «Уральский НИИ композиционных материалов», «НИИ полимерных материалов», ГП «Возрождение», ОАО «Новомет» и других эффективна на основе создания и развития математических моделей сложных объектов. В частности, совместно с ОАО «НПК Уралвагонзавод» развернуты работы по проектированию элементов конструкций подвижного состава железных дорог из нанокомпозитов. При этом появляется возможность снизить объем порожних перевозок и сократить расходы на капитальный ремонт вагонов-хопперов. Для финансирования производства уникальной продукции обратились с заявкой в Роснано.

Совместно с ОАО «Уралкалий», ОАО «Сильвинит» и ГП «Возрождение» в ПГТУ решается актуальная задача обеспечения работоспособности и износоустойчивости оборудования шахт и обогатительных фабрик по производству калийных удобрений на основе решения связанных задач гидрогазодинамики и напряженно-деформированного состояния конструкций при динамических нагрузках с учетом износа. Оптимизация этих задач на базе математического трехмерного моделирования проведена на Пермском кластере.

На основе математического моделирования эффективно использование ВВК для постоянного мониторинга окружающей среды, контроля переноса выбросов предприятий в окружающей среде в режиме реального времени и в масштабах региона. На ВВК ПГТУ по заказам предприятий проводятся вычислительные эксперименты по моделированию газогидродинамических процессов в каналах переменного сечения газоходов, выступающих в роли очистных сооружений для промышленных отходов.

Большой объем вычислительных экспериментов проводится для моделирования аэродинамики и внутрикамерных газодинамических процессов в двигателях летательных аппаратов. Решается ряд задач, связанных с работой подвижных элементов конструкций в потоке жидкости.

Активная работа ведется по вычислительному моделированию поведения строительных конструкций, предупреждению осадки фундаментов и возникновения трещин.

В сотрудничестве с пермскими вузами – Государственным университетом, медицинской, фармацевтической и сельскохозяйственной академиями реализуется возможность проведения многомерных вычислительных экспериментов применительно к задачам медицины, водоочистки, сохранения и переработки сельхозпродукции, деревозаготовительного и деревообрабатывающего комплексов, актуальным для Пермского края. В частности, решаются задачи моделирования процессов, связанных с оценкой проницаемости межклеточного пространства живых

тканей с учетом динамического напряженно-деформированного состояния мышц и кровотока. Этот класс задач важен для создания эффективных лекарственных препаратов и методик лечения тяжелых заболеваний. Создано уникальное программное обеспечение «Тороскопия» для автоматизированного формирования методики проведения операций на легких с учетом многофакторного анализа и экспертных оценок.

Назрела необходимость решения на пермском ВВК задач эффективного моделирования и прогнозирования распределения транспортных потоков по улично-дорожной сети Перми и Пермского края, связанных с повышением безопасности дорожного движения. Работы по транспортному планированию и развитию городской транспортной системы, в том числе строительству мостов, путепроводов, маршрутной сети пассажирского транспорта планируются совместно с Правительством Пермского края и администрацией г. Перми.

В последнее время в связи с заказами предприятий появились новые задачи для кластера. Речь идет о создании и анализе моделей, описывающих колебательные режимы при эксплуатации газоперекачивающих агрегатов с учетом существующих технологических погрешностей изготовления. Кроме того, в рамках реализации совместного проекта ПГТУ и ОАО «Протон-ПМ» по созданию высокотехнологичного производства для испытания газотурбинных установок мощностью до 40 МВт на многоцелевом адаптивном экологичном стенде с общим объемом финансирования 600 млн руб. на кластере ПГТУ развернуты работы по исследованию кавитационных эффектов технологического оборудования и анализу газогидродинамических процессов и напряженно-деформированного состояния конструкций в связанной постановке.

Принципиально новые возможности суперкомпьютер обеспечивает и в организации учебного процесса в университете. Вычислительные мощности кластера, доступные как в пакетном режиме, так и в режиме удаленного доступа, удобны для

пользователей университета при организации самообразования, контроле качества обучения, проведении научных и инженерных расчетов, требующих больших вычислительных ресурсов. Организован процесс обучения магистрантов. Регулярно, примерно раз в неделю, в режиме телемоста, проводятся трансляции лекций по параллельным вычислениям ведущих преподавателей из г. Москвы, С.-Петербурга, Н. Новгорода для слушателей из ПГТУ.

Перспективным направлением развития будет создание виртуальной сети, объединяющей несколько суперкомпьютеров для решения задач более высокого уровня. Работы в данном направлении продолжаются по программе «Университетский кластер». Появляется возможность ускоренной реализации межрегионального стратегического проекта «Белкомур», включающего северные территории в единый транспортный поток России.

Пермский государственный технический университет вошел в Суперкомпьютерный консорциум университетов России. Пермскому государственному техническому университету присвоена категория Национального исследовательского университета России. В связи с этим можно ожидать нового подъема эффективности реализации высокопроизводительных вычислений на кластере ПГТУ для фундаментальных и прикладных научных исследований и нужд промышленных предприятий Пермского региона.

В.П. Матвеенко, А.Г. Масич, Г.Ф. Масич

Институт механики сплошных сред УрО РАН, г. Пермь

РЕГИОНАЛЬНАЯ НАУЧНО-ОБРАЗОВАТЕЛЬНАЯ ОПТИЧЕСКАЯ СЕТЬ УРО РАН – ФУНДАМЕНТ КИБЕРИНФРАСТРУКТУРЫ

Эволюция R&E-коммуникаций. Отработанные технологии построения сетей привели в середине 90-х годов к коммерциализации Интернета и стремительному техническому подъему. В этот период по всему миру прокладывалось оптоволокно со скоростью более 8000 км/час (70 миллионов км в 1999 году !)

[1]. Чтобы расширить полосу пропускания, внедрялась технология плотного волнового мультиплексирования (DWDM), обеспечивающая возможность передавать 10–40–100 Гбит/с по каждому «лямбда»-каналу. Это позволяло увеличить пропускную способность без прокладки новых дорогостоящих волоконно-оптических линий связи (ВОЛС). Таким образом, можно с уверенностью утверждать, что для протяженных высокоскоростных линий связи альтернативы оптическому волокну в обозримом будущем отсутствуют.

В этот же период времени (90-е годы) R&E-сообщество осознало [2], что потеряло контроль над дефицитным ресурсом (каналами) телекоммуникационной промышленности. Развернутые за последние три десятилетия XX века R&E-сети были построены на арендованных каналах. Однако в 2000 году стремительный технический подъем сменился техническим кризисом, ряд компаний по протяженным каналам связи обанкротились, и ввиду переизбытка пропускной способности компании–поставщики телекоммуникационных услуг были готовы обсудить вопрос о продаже или долгосрочной аренде неиспользуемого ими оптоволокна (dark fiber – темного) или лямбда-каналов. Возможностью приобретения «темных» волокон воспользовались многие R&E-организации в мире. В этом случае судьба пропускной способности оказывается в руках конечных клиентов, а не у поставщиков сервисов и потоков. Результат – стратегическая обеспеченность R&E-сообщества. Примеры таких подходов освещены в [3]. Это следующие сети: NLR (National LambdaRail) в США, более 17000 км оптического волокна, скорость 6,4 Тбит/с; GIGA в Бразилии, национальная исследовательская и образовательная сеть, 700 км «темного» оптоволокна, WDM-оборудование; CSI (Cyber Science Infrastructure) в Японии; TWAREN – тайваньская научно-образовательная сеть объединяет магистралью 10 Гбит/с одиннадцать региональных центров; GEANT2 (Gigabit European Academic Network Technology) – седьмое поколение Европейской научно-образовательной сети.

В России история создания R&E-сетей в национальном масштабе началась с 1980 года, когда в СССР была разработана про-

грамма и проект создания Академсети, структурно представляющая собой совокупность девяти взаимосвязанных региональных вычислительных подсетей (РВПС) СССР. В 1986–1990 годах была создана рабочая зона РВПС «УРАЛ» Академсети (1,2–2,4 Кбит/с). В период 1991–2000 годов со снятием эмбарго на ввоз высоких технологий создание и развитие региональной информационно-вычислительной сети (РИВС) УрО РАН происходило на новой технической базе.

Однако по настоящее время R&E-сети в России, национальные (RBNet – Russian Backbone Network, RUNNet – Russian University Network, RASNet) и региональные (например, сеть УрО РАН), все еще арендуют дорогостоящие потоки 45–155 Мбит/с. Так, например, сеть УрО РАН состоит из созданных и развиваемых в шести научных центрах собственных оптических инфраструктур: Архангельск – АНЦ, Сыктывкар - КомиНЦ, Ижевск – УдНЦ, Пермь – ПНЦ, Челябинск – ЧНЦ, Оренбург – ОНЦ и Екатеринбург. Так, сеть Пермского научного центра (ПНЦ) объединяет оптическим волокном на скорости 1 Гбит/с по IP/GE-технологии четыре института и Президиум ПНЦ (рис. 1), связана с оптическими сетями Пермского государственного технического университета (ПГТУ) и Пермского государственного университета (ПГУ). Оптическая инфраструктура построена в кооперации с различными ведомствами и организациями для уменьшения затрат на приобретение части волокон и последующее обслуживание линий связи. Волокна не выкуплены по причине их обилия в городе, как это происходило в США и других странах, а построены собственные волоконно-оптические линии связи (ВОЛС).

Подключение научных центров к Екатеринбургу выполнено по арендуемым потокам на скорости 2–4 Мбит/с. Централизованный доступ в Интернет осуществляется по арендуемому потоку 75 Мбит/с «Екатеринбург–Москва». И увеличение, например, на два порядка (0,2 Гбит/с) скоростей в магистралях между научными центрами, по нашей оценке, потребует увеличения почти на два порядка ежегодных затрат на аренду, что бесперспективно, дорого, да и скорости не мирового уровня.

Рис. 1. Существующая оптическая инфраструктура сети ПНЦ УрО РАН

Инициатива GIGA UrB RAS [4] направлена на достижение стратегической обеспеченности Уральского отделения РАН скоростными коммуникациями путем преодоления отрицательного влияния сложившейся в России практики аренды дорогостоящих каналов связи посредством собственного «темного волокна» и DWDM-технологии (рис. 2). В ИМСС УрО РАН найдены решения для объединения существующих и развиваемых оптических сетей научных центров в городах Екатеринбурге, Перми, Ижевске, Сыктывкаре и Архангельске, обеспечивая сравнимые с мировым уровнем скорости. Экономическая целесообразность Инициативы GIGA UrB RAS оценена на примере сопоставления разовых и ежегодных затрат:

- предлагаемое решение 40 Гбит/с ~ 400 млн руб. разово;
- возможная аренда 0,2 Гбит/с ~ 400 млн руб. ежегодно.

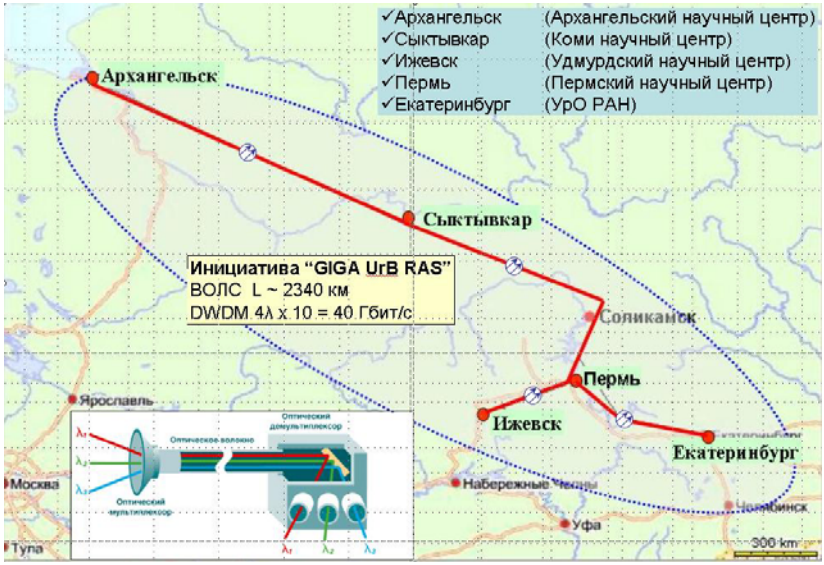


Рис. 2. Создаваемая региональная научно-образовательная оптическая сеть УрО РАН (Инициатива GIGA UrB RAS)

Разовые затраты на приобретение двух академических оптических волокон (срок эксплуатации 25–30 лет) для нужд научно-образовательного сообщества Урала и прилегающих территорий и заложенная возможность поэтапного наращивания производительности DWDM-системы до терабитных скоростей обеспечивают защиту инвестиций и согласуются с мировой тенденцией развития научно-образовательных сетей УрО РАН (Инициатива GIGA UrB RAS).

Идея «Инициативы GIGA UrB RAS» одобрена и реализуется Уральским отделением РАН в рамках постановления и распоряжения о создании «Региональной научно-образовательной оптической сети», объединяющей научные центры Уральского отделения РАН. Соучастие в проекте образовательных учреждений Уральского региона и прилегающих территорий формируется в рамках соглашений и договоров с Уральским отделением РАН. Начата реализация первого этапа проекта на участке «Пермь–Екатеринбург».

Эволюция технологий супервычислений. Для доступа к распределенным в пространстве вычислительным ресурсам R&E-сообщество создало грид-технологии, в которых изначально использовались разделяемые между всеми пользователями Интернета TCP/IP-сети (публичный, public Интернет). Однако если передавать терабайтные файлы по public Интернет, то передача будет длиться очень долго и ее использование станет крайне неэффективно.

Следующий этап развития национальных и региональных оптических R&E-сетей привел к созданию международной виртуальной организации GLIF (Global Lambda Integrated Facility) [5] – глобальной лямбда-системы, продвигающей парадигму глобальных лямбда-сетей. Деятельность участников GLIF направлена на интеграцию своих «лямбд» в глобальную систему для их использования учеными и проектами, требующими передачи большого количества данных, например, по схеме «точка-точка» по одной или нескольким лямбда-каналам. Сообщество GLIF разделяет взгляды в построении новых парадигм грид-вычислений, в которых центральный архитектурный элемент – это оптические сети, а не компьютеры. Эта парадигма называется LambdaGrid, направлена на поддержку наиболее требовательных к скоростям научных приложений этого десятилетия, основывается на использовании параллелизма, как и в суперкомпьютинге десятилетие назад. Однако у GLIF параллелизм заключен в многочисленных длинах волн света (лямбд) в одном оптическом волокне.

Киберинфраструктура. Национальный научный фонд США (NSF) в понятии «киберинфраструктура» в начале XXI столетия [6] в более общей постановке сформулировал задачу развития IT-отрасли. «Всестороннюю инфраструктуру, которая должна объединять все усовершенствования в сфере информационных технологий назвали киберинфраструктурой (Cyberinfrastructure – CI). CI объединяет в себе аппаратные средства для вычислений, данные и сети, цифровые датчики, обсерватории и экспериментальные средства, взаимодействующий набор программного обеспечения и услуг микропрограммных средств

и инструментов. Миссия NSF касательно киберинфраструктуры (CI) [7]. Разработка антропоцентрической CI, которая управляется возможностями образования и исследований в науке и технике; обеспечение научным и техническим сообществам доступа к инструментальным средствам и услугам CI мирового класса. Продвижение CI для расширения участия национальной рабочей силы во всех областях науки и техники. Обеспечение жизнеспособной CI, которая является безопасной, эффективной, надежной**,** доступной, вполне используемой и способной к взаимодействию и которая развивается как основная национальная инфраструктура для обеспечения образования и исследований в науке и технике. Интеграция и совместное использование активов киберинфраструктуры, развернутых и поддерживаемых на национальном, региональном, местном уровнях и на уровне университетских городков». В этом смысле киберинфраструктура – это движущая сила науки и техники, а высокоскоростные коммуникации – фундамент киберинфраструктуры.

Суть проекта «Распределенный PIV» [7] – обработка в реальном времени получаемых в ИМСС (Пермь) на PIV-экспериментальной установке изображений на удаленном суперкомпьютере. Широко используемый метод PIV (Particle Image Velocimetry) [8] основан на цифровой трассерной визуализации, позволяет в плоскости и толщине лазерного ножа фиксировать цифровыми камерами перемещения трассеров и рассчитывать, например, поля скорости. Точность измерений зависит как от характеристик видеокамер, так и от возможностей алгоритмов расчета.

Экспериментальная установка PIV, установленная в ИМСС (Пермь), генерирует поток экспериментальных данных на скорости 1–10–100 Гбит/с, зависящей от разрешения и числа камер и частоты проводимых измерений. Ограниченность вычислителя экспериментальной установки сдерживает развитие математического аппарата и возможности управления экспериментов в реальном времени. Перенос вычислений на многопроцессорные системы позволит использовать ресурсоемкие, но высокоточные алгоритмы, избегать хранения гигантских объемов избыточной информации, обрабатывать измерения «на лету» и проводить

эксперименты с обратной связью. Пакет программ на супервычислителе реализует алгоритм параллельного расчета мгновенных полей скорости по изображениям, поступающим с измерительной установки.

На первом этапе (2009) [7] использован суперкомпьютер СКИФ (НИВЦ МГУ, Москва, 60 Tflops) по временно организованному магистральному каналу связи протяженностью ~1500км на скорости 1 Гбит/с (прототип LambdaGrid). Инфраструктура эксперимента изображена на рис. 3. Цель первого этапа работ – оценка существующих возможностей научно-образовательных коммуникаций и вычислительных ресурсов России и уточнение модели распределенного в пространстве эксперимента реального времени. На первом этапе выявлен ряд нерешенных задач как в части протоколов передачи данных, так и способов ввода интенсивных потоков данных в супервычислитель.

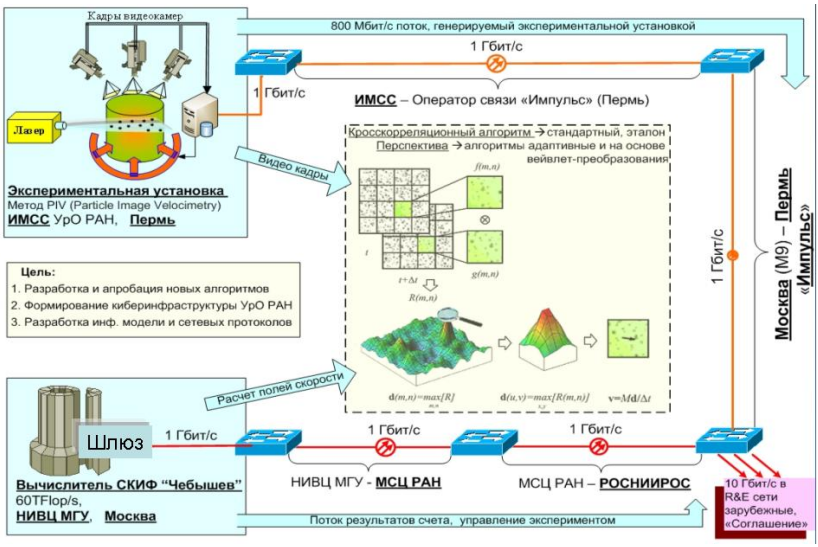


Рис. 3. Инфраструктура проекта «Распределенный PIV» (Этап 1, 2009 г.)

На втором этапе [9, 10] (2010) разработано «идеальное» архитектурное решение, заключающееся в прямом вводе в вычислительные узлы супервычислителя экспериментальных дан-

ных, в том числе протокола обмена данными между экспериментальной установкой и удаленной многопроцессорной системой (протокол PIV). Отладка и апробация «идеальной» модели выполнялись в связке экспериментальной установки PIV с МВС-1000/16П (ИМСС, Пермь) по L2 каналу 0,1 Гбит/с и с МВС ИММ «um64» (ИММ УрО РАН, Екатеринбург) через public Интернет на скорости 15Мбит/с . Подробное описание результатов второго этапа исследований выходит за рамки этой статьи, однако следует отметить состоятельность предложенной модели организации обработки экспериментальных данных, в том числе и направления последующих работ. Создание переносимого системного программного обеспечения (например, в рамках MPI-библиотек), доступного для использования в задачах обработки и передачи интенсивных потоков данных; организация скоростных портов ввода/вывода супервычислителя; анализ, исследование и разработка эффективных протоколов передачи данных.

Общепризнано сегодня, что исследователи из академических институтов и университетов играют ключевую роль и в разработке фундаментальных интернет-технологий, и их апробации в крупномасштабных испытательных моделях. Яркий тому пример – амбициозная «Инициатива GIGA UrB RAS», которая не уступает аналогичным проектам науки и образования США, Европы и Японии и хорошо ложится на организационно-территориальную структуру УрО РАН. Поскольку чем больше протяженность и чем выше скорость передачи, тем более заметны преимущества оптических сетей.

Сравнение тенденций роста мощности у технологий вычислений, хранилищ и сетевых показало, что производительность сети растет значительно быстрее, чем у вычислителей и хранилищ данных. Это означает, что становится более экономичным построить или приобрести высокоскоростные сети связи для доступа к удаленным вычислительным ресурсам и памяти, чем дублировать и поддерживать эти ресурсы на местах.

Проект «Распределенный PIV» иллюстрирует возможности новых парадигм организации супервычислений, основанных на возможности использования больших скоростей передачи данных, и является одним из примеров создаваемой киберинфраструктуры УрО РАН.

Список литературы

1. Philip Papadopoulos, Larry Smarr. Introduction // CTWatch Quarterly. Vol. 1. No. 2, may 2005. P 2–4.

2. David Farber, Tom West. The National LambdaRail. Cyberinfrastructure for Tomorrow's Research and Education // CTWatch Quarterly. Vol. 1 No. 2, may 2005. P 14–19.

3. Масич А.Г., Масич Г.Ф. От Инициативы GIGA UrB RAS к киберинфраструктуре УрО РАН. // Вестн. Перм. науч. центра (октябрь–декабрь 4/2009). – Пермь: Изд-во ПНЦ УрО РАН, 2009. – С. 41–56.

4. Масич А.Г., Масич Г.Ф. Инициатива GIGA UrB RAS // Вычислительные технологии, Вестник КазНУ им. Аль-Фараби. Серия «Математика, механика, информатика». № 3 (58). – Алматы, 2008.

5. Global Lambda Integrated Facility. – URL: http://www.glif.is/ 20.12.2009.

6. Cyberinfrastructure Vision for 21st Century Discovery, National Science Foundation Cyberinfrastructure Council, March 2007.

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

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

9. Масич А.Г., Масич Г.Ф. GIGA UrB RAS-подход к LambdaGrid парадигмам вычислений // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: тр. Междунар. суперкомпьютерной конф. – М.: Изд-во МГУ, 2010. С. 4–11.

10. Скоростной I/O канал супервычислителя и протокол для обмена интенсивным потоком экспериментальных данных / А.Г. Масич [и др.] // Высокопроизводительные параллельные вычисления на кластерных системах (HPC–2010) (1–3 ноября 2010 г., г. Пермь): материалы X Междунар. конф. – Пермь: Изд-во Перм. гос. техн. ун-та, 2010.

[1,3]В.Н. Снытников, [1]Э.А. Кукшева, [1,3]Т.В. Маркелова, [2]Н.В. Снытников, [1]О.А. Стадниченко, [1]О.П. Стояновская

[1]Институт катализа им. Г.К. Борескова, г. Новосибирск
[2]Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск
[3]Новосибирский государственный университет

СУПЕРКОМПЬЮТЕРНЫЕ ВЫЧИСЛЕНИЯ С ЭКЗАФЛОПНОЙ ПРОИЗВОДИТЕЛЬНОСТЬЮ В АСТРОФИЗИКЕ И АСТРОКАТАЛИЗЕ

Сегодня особый интерес в естественных науках вызывает образование околозвездных дисков при гравитационном коллапсе среды в газопылевых туманностях. В этих дисках формируются планеты, которые обнаружены у сотен звезд помимо нашего Солнца. С начальными этапами зарождения звезд и планет связана химическая эволюция. Она обеспечила в Солнечной системе синтез предбиологических соединений, зарождение жизни и возникновение на Земле биосферы.

Целью наших исследований в астрофизике и астрокатализе с использованием суперкомпьютеров стало создание численных моделей околозвездных дисков для сравнения с наблюдательными данными. Нас интересуют условия в среде, при которых шла химическая и предбиологическая эволюция, с тем

чтобы воспроизвести в лабораторных экспериментах отдельные этапы этой эволюции. Необходим научный ответ на вопросы, где, как и когда, в каких условиях произошел абиогенный синтез необходимых для зарождения жизни сложных органических соединений [1].

Диапазон изменения временных, пространственных и других масштабов при образовании околозвездных дисков с гравитационным коллапсом составляет много порядков. Численное решение пространственно трехмерных, нестационарных и неустойчивых задач о коллапсе даже однофазной газовой среды с самосогласованной гравитацией является предельно сложным для существующих суперкомпьютеров. При учете таких важных физических факторов, как магнитное поле, излучение, наличие пылевого компонента, а также химических реакций, постановка этих задач будет еще долго опережать возможности суперкомпьютеров, быстро наращивающих свою вычислительную мощь, и стимулировать их развитие к уровню экзафлопной производительности.

Алгоритмы и численные методы. В подобных задачах свое развитие получают и численные методы решения систем уравнений математической физики в частных производных. Такие системы находят применение в самых широких областях науки и техники. Для их решения часто используются методы расщепления по физическим процессам. Создание основы решения базовой системы уравнений, без которой невозможно обойтись, позволяет продолжать наращивать математическую и численную модель с включением в нее новых физико-химических процессов. В нашем случае минимально необходимой математической моделью служит пространственно трехмерная нестационарная система уравнений Эйлера для газа с гравитацией, дополненная уравнением Пуассона для гравитационного поля. Это система уравнений общего вида, а наличие эллиптического уравнения Пуассона, для которого характерна бесконечная скорость передачи возмущений, вызывает к жизни физические неустойчивости, включая неустойчивость Джинса с коллапсом газа. Состояние межзвездного газа из смеси водорода

и гелия, в диапазоне температур от нескольких десятков до десятков тысяч градусов по шкале Кельвина и при концентрациях 10^4–10^{19} частиц/см^3, меняется так, что эффективный показатель адиабаты лежит в пределах от 1 до 5/3. Течение вращающегося газа при коллапсе имеет области сверх- ,транс- и дозвуковой динамики с набором ударных волн и других разрывов, а также затопленные струи газа.

Для моделирования трехмерной динамики облака газа был использован численный код [2], созданный авторами доклада. В основе кода лежит метод FLIC (метод крупных частиц, метод Белоцерковского-Давыдова). Метод использует расщепление по физическим процессам. На первом этапе учитываются изменения скорости за счет градиентов давления и гравитационного потенциала. В качестве начального условия берется значение функций с предыдущего шага по времени. Специальный оператор аппроксимации градиентов давления и потенциала на 27-точечном шаблоне позволяет уменьшить влияние эффекта выделенных направлений. Первый этап имеет второй порядок аппроксимации по пространству и первый порядок по времени. На втором этапе рассчитываются потоки масс через границы эйлеровой сетки с начальным условием из результата первого этапа. Этот алгоритм обеспечивает первый порядок аппроксимации по пространству и по времени. На третьем этапе решается уравнение Пуассона для СЛАУ на 7-точечном шаблоне методом быстрого преобразования Фурье. Общий порядок аппроксимации метода по пространству и по времени – первый. В коде использована равномерная сетка в декартовой системе координат.

На этапе потери газа из ближней к протозвезде зоны околозвездного диска необходимо учесть динамику первичных тел, которые двигаются с редкими столкновениями друг с другом на временах нескольких оборотов вокруг протозвезды. Математическая модель для динамики самогравитирующих тел представляет собой кинетическое уравнение Власова (иные термины, уравнение Лиувилля, бесстолкновительное уравнение Больцмана – аналоги нестационарного уравнения Шредингера). Для решения уравнения Власова используется метод частиц в ячейках.

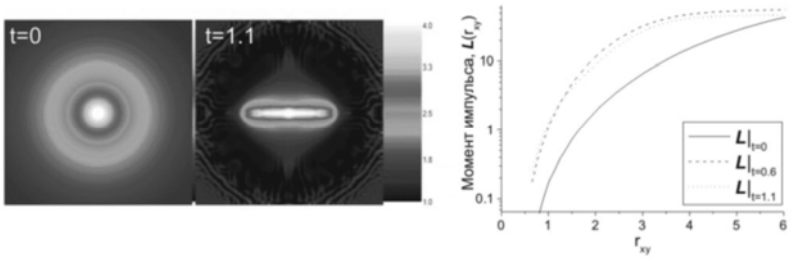
Однако система из уравнения Власова и уравнения Пуассона без газодинамических уравнений представляет самостоятельный интерес в большом числе приложений, в частности, как основа метода молекулярной динамики. В нашем случае это уравнение для функции 6 переменных в фазовом пространстве решается методом частиц в ячейках для самосогласованного поля. Точность метода зависит от используемого в расчетах числа частиц и требует свыше 400–1000 частиц в ячейке для удовлетворительного воспроизведения возможных неустойчивостей. Для сетки 1000^3 это примерно 10^{11}–10^{12} частиц. Нестационарные задачи с такими параметрами вычислений требуют от суперкомпьютеров экзафлопную производительность.

Результаты вычислительных экспериментов и обсуждение. Для проведения численных экспериментов использовались: 8-процессорная (XeonQC, 2 узла) HP-машина с общей памятью 40 Гбайт в ИК СО РАН, машина SMP16x256 с 4 четырехядерными процессорами XeonX7350 с общей памятью 256 Гб (ССКЦ, Новосибирск); машина с распределенной памятью NKS-160 с 84 вычислительными узлами HP и процессорами Intel Itanium2/1.6 GHz с памятью 4 Гб на узел (ССКЦ, Новосибирск), а также MVS-100K (МСЦ, Москва). Доступ на ССКЦ и МСЦ осуществлялся по интернет-каналу СО РАН. Визуализация проводится с помощью собственной разработки Gala Э.А. Кукшевой.

На SMP проведено численное моделирование динамики гравитирующего газа, описывающее в изотермическом газе режимы формирования протозвезд и протозвезд вместе с околозвездными дисками. Используемые размеры сеток – 128^3 и 256^3. Расчеты по программе, созданной без тщательной оптимизации, занимают на SMP несколько суток на сетке 256^3 при числе временных шагов свыше 10^4 для физически интересных результатов. Более подробные сетки 512^3 уже требуют организационных мер на SMP для заказа необходимой общей памяти. Поскольку при этом требуется более чем в 2 раза уменьшать шаг по времени, то это в целом более чем в 20 раз увеличивает время счета – свыше месяца. Физически интересные задачи на сетках 1000^3 и более для выявления деталей формирования Солнечной системы от орбиты Меркурия до сотни астрономических единиц

(AU), на которых находятся кометы и другие объекты Кипера–Белта, будут рассчитываться годы. Такие затраты очевидно неприемлемы, что требует кардинального увеличения скорости вычислений к экзафлопному уровню.

Результаты вычислительных экспериментов, полученные на SMP, приведены на рис. 1. Рассчитывается динамика изотермического газа с развитием сильной гравитационной неустойчивости, которая приводит к коллапсу газа. Показано, что во вращающемся и сжимающемся изотермическом газе существуют режимы формирования протозвезд вместе с околозвездными дисками. Масса центрального тела примерно в 10 раз превосходит массу диска, что хорошо соотносится с наблюдениями среднемассивных околозвездных дисков на поздних стадиях их формирования. Момент импульса в зависимости от радиуса в цилиндрической системе координат сформированной структуры (рис. 1, *б*) из протозвезды с околозвездным диском распределяется неравномерно. Внутренним областям, сосредоточившим в себе до 90 % массы облака, передается около 1 % начального момента импульса, и они сжимаются в протозвезду. Внешним областям передается до 98–99 % момента импульса, и они формируют плотный диск, вращающийся вокруг центрального тела. Такое распределение наблюдается в Солнечной системе, где большая часть массы системы сосредоточена в Солнце, а подавляющая часть момента импульса – во внешних планетах.



а *б*

Рис. 1. Распределение $\lg\rho(x,0,z)$ логарифма плотности при $t = 0$, 1,1 (*а*); Распределение момента импульса $L(r_{xy})$ при $t = 0$ (сплошная линия), $t = 0{,}6$ (штриховая линия), $t = 1{,}1$ (пунктирная линия) (*б*)

В работе [3] приведены наши результаты решения на кластерных суперкомпьютерах ССКЦ и МСЦ системы из уравнений Власова и Пуассона. На практике в первую очередь мы сталкиваемся с проблемой увеличения размера вычислительного массива для решения уравнения Пуассона, которое в нашем случае решается методом Фурье по трем координатам. На кластерных системах при увеличении массива приходится увеличивать число процессоров, участвующих в вычислениях. Как показано в [3], даже при фиксированном размере массива при увеличении числа процессоров растет время решения уравнения Пуассона. Другими словами, эффективность распараллеливания падает. При более подробном исследовании этой ситуации было выяснено, что время в основном тратится на пересылки между процессорами. Таким образом, при работе на кластерах мы упираемся в ограничение на используемое число процессоров для нашей задачи. Вероятно, коммуникационные сети кластеров слишком медленные для нашей задачи.

С другой стороны, у нас алгоритм состоит из сильносвязанной подзадачи из решения уравнения Пуассона с большим массивом для потенциала и слабосвязанной подзадачи из решения уравнения Власова, которое интегрируется через большое число ОДУ, связанных с потенциалом. Теоретически в целом задача органичнее укладывается на архитектуру вычислительных систем с общей памятью, нежели на архитектуру систем с распределенной памятью. Для проверки этого предположения была написана соответствующая версия программы, использующая OpenMP. На машине с общей памятью Института катализа СО РАН эффективность распараллеливания этой программы составила около 70 % на 8 процессорах для сетки 256×256×64. Эффективность распределенной версии программы с MPI для данной задачи на числе процессоров 256 близка к нулю. Но 8 процессоров слишком мало, чтобы делать общий вывод о том, что данную задачу нужно решать на машине с общей памятью. Необходимы проверки на практике этого предположения на машине с увеличением числа процессоров.

По параллельному коду для нестационарных пространственно трехмерных задач на кластерах ССКЦ и МСЦ были проведены вычислительные эксперименты по изучению устойчивости бесстолкновительных гравитирующих систем. На рис. 2 приведены результаты моделирования динамики формирования сгустков в плоском диске из тел в их самосогласованном гравитационном поле при развитии неустойчивости Джинса. Исследовалось поведение численного решения при сгущении вычислительной сетки и увеличении числа частиц. Обезразмеривание проведено в единицах общей массы, гравитационной постоянной и линейного масштаба в системе. Начальный радиус диска был равен 3, его масса – 1, центрального тела и внешних потенциалов не было, начальные дисперсии скоростей dvr = 0,02, dvφ = 0,01, dvz = 0,1. Расчетная область задавалась как 10×10×10. Для выбранных параметров характерное значение длины Джинса было около 0,1. Интерес представляют расчеты с шагом вычислительной сетки много меньшим, чем 0,1.

Из приведенных рисунков следует, что макроскопическая динамика тел для всех вариантов близка друг к другу. Частицы разлетаются в самосогласованном гравитационном поле, толщина диска увеличилась, плотность и температура (дисперсии) функции распределений по скоростям упали. В центральной области диска возникли условия для развития неустойчивости Джинса в бесстолкновительной системе. Однако на сетке 128^3, где шаг сетки примерно равен длине Джинса (рис. 2, *а*) и средним числом модельных частиц в ячейке порядка 10^3 обнаруживается только тенденция к формированию сгустков. Уменьшение шага сетки в 2 раза на сетке 256^3 позволило выявить формирование сгустков, хотя число частиц в ячейке уменьшилось в 2–4 раза. Дальнейшее уменьшение шага сетки в 4 раза по двум направлениям с заданием сетки 1024×1024×256 узлов и увеличение общего числа частиц в 10 раз для приближенного сохранения числа частиц в ячейке приблизительно 10^3 позволило надежно рассчитать структуру сгустков, возникающую при развитии неустойчивости Джинса на ее нелинейных стадиях (рис. 2, *в*).

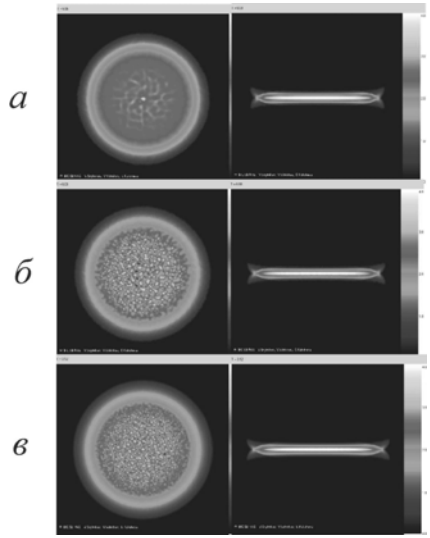
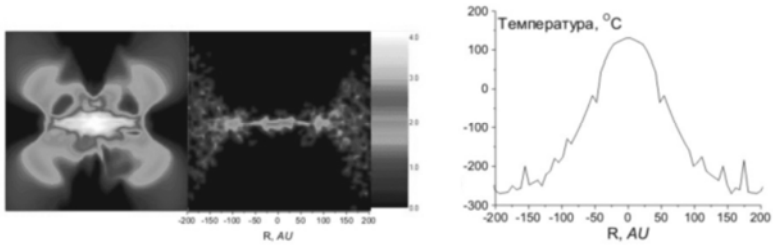


Рис. 2. Распределение логарифма поверхностной плотности вещества в экваториальной и меридианальной плоскостях диска в момент времени $T = 6{,}08$ для трех расчетов: *а* – сетка 128^3 узлов и 10^8 модельных частиц, *б* – сетка 256^3 узлов, число частиц 10^8, *в* – сетка 1024^2·256, число частиц – 10^9

В ССКЦ на SMP проведены расчеты по двухфазной модели с включением твердой фазы с прогнозом условий для синтеза сложных молекул в околозвездных дисках. В качестве начальных условий были заданы адиабатический газ и диск из частиц. Известно, что дисковые структуры более стабильны, чем чисто газовые при учете твердого компонента из первичных тел (рис. 3).

В ходе вычислительных экспериментов для двухфазной модели получены результаты исследования кинетики реакции Бутлерова и определения зоны химической эволюции в формирующемся газопылевом диске. В частности, проведены расчеты динамики двухфазной среды с первичными телами при размере расчетной области 800AU. Рис. 3 иллюстрирует появление плотного диска радиусом $\approx 40AU$. Газовый диск включает в себя около 10 % начальной массы газа. Наличие твердой фазы, начальная доля которой 0,1 от массы газа, приводит к стабилизации системы – диск не разрушается под воздействием гравитационно-

конвективной неустойчивости в течение полутора оборотов по внешнему радиусу. Основная масса газа в «бабочкообразной» структуре тянет за собой частицы, которые в результате разлетаются вдоль оси OZ. Структура, которую образуют частицы твердой фазы, не противоречит структуре Солнечной системы. На рис. 3 показано, что в сформированном диске из газа и первичных тел диапазон температур составляет от -35^{O}C на периферии до 130^{O}C в центре. Область плотного диска, температура в которой лежит в диапазоне жидкой воды, является возможной зоной химической эволюции и синтеза сложных органических молекул.



а *б*

Рис. 3. Распределение lg $F_{g,p}(x,0,z)$ – логарифма плотности газа и частиц при t = 1280 лет (*а*); температура газа $T(x,0,0)$ при t = 1280 лет (*б*)

Для моделирования квазитрехмерной динамики двухфазного гравитирующего диска создан параллельный код с учетом химических реакций и их тепловых эффектов, основанный на методе частиц-в-ячейках для решения уравнения Власова, методе SPH для решения уравнений газовой динамики и методе последовательной верхней релаксации и быстрого преобразования Фурье для решения уравнения Лапласа [4]. Проведено моделирование двух режимов динамики двухфазного диска и показано, что как кольцевые, так и спиральные волны плотности (рис. 4), возникающие в двухфазной среде, могут считаться характерными для самой системы, а не привнесенными в нее особенностями алгоритмов. На характерное время существования волн может повлиять минимальный размер возмущения, воспроизводимый при решении уравнений газовой динамики, в случае

кольцевых волн или аналогичный масштаб при решении уравнения Пуассона и Власова в случае спиральных волн. Показано, что даже при соотношении массы частиц к массе газа $1/50$ имеет место взаимное влияние газового и пылевого компонентов системы.

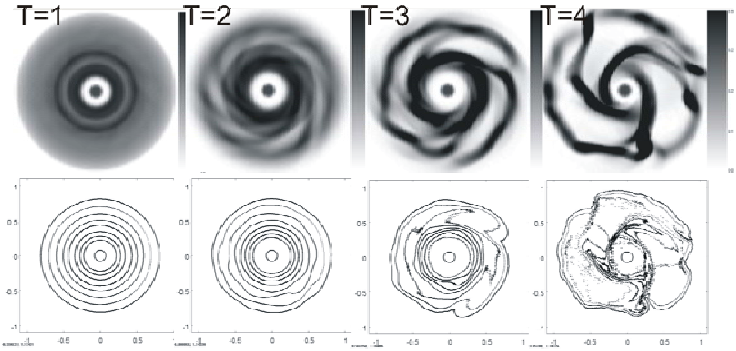


Рис. 4. Динамика спиральных волн плотности (логарифм поверхностной плотности пылевой (сверху) и газового (снизу) компонента диска для моментов времени $T = 1; 2; 4; 6$

По результатам вычислительных экспериментов получено, что к числу процессов, влияющих на устойчивость диска и ведущих к образованию сгущений, относится гравитационное взаимодействие газа и первичных тел. Процессы в субдиске первичных тел, где имеет место небольшой разброс частиц по скоростям, могут инициировать и ускорять формирование сгущений в массивном газовом диске. Присутствие массивного газового диска делает возможным гравитационный рост возмущений плотности в системе, возникших в субдиске первичных тел.

Таким образом, вычислительные эксперименты на суперкомпьютерах в астрофизике и в астрокатализе показывают, что химическая эволюция с синтезом первичных органических соединений для земной биосферы в Солнечной системе действительно могла идти на этапе околосолнечного диска. Разработаны численные коды для решения задач гравитационной газодинамики, которые позволяют моделировать условия в околозвездных

дисках, как в химических реакторах для реагентов и продуктов. С помощью этих программ выполнены прогнозы условий для синтеза сложных молекул в околозвездных дисках.

Список литературы

1. Снытников В.Н. Абиогенный допланетный синтез пребиотического вещества // Вестник РАН. – 2007. – Т. 77, № 3. – С. 218–226.

2. Стадниченко О.А., Снытников В.Н. Явный многошаговый алгоритм для моделирования динамики самогравитирующего газа // Вычислительные методы и программирование. – 2010. – Т. 11, № 1. – С. 53–67.

3. Кукшева Э.А., Снытников В.Н. Параллельный алгоритм решения задач гравитационной физики, основанный на декомпозиции области // Вычислительные методы и программирование. – 2010. – Т. 11, № 1. – С. 168–175 (http://num-meth.srcc.msu.ru/).

4. Стояновская О.П., Снытников В.Н. Особенности SPH-метода решения газодинамических уравнений при моделировании нелинейных волн в двухфазной гравитирующей среде // Математическое моделирование. – 2010. – Т. 22, № 5. – С. 29–44.

СЕКЦИОННЫЕ ДОКЛАДЫ

Д.Ю. Андреев, О.В. Джосан

Московский государственный университет им. М.В. Ломоносова

МЕТОД ОПТИМИЗАЦИИ ВВОДА-ВЫВОДА В ПРИЛОЖЕНИЯХ ДЛЯ МАССИВНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

В настоящее время в мире решается множество задач по обработке данных, параллельные алгоритмы создаются и используются на различных архитектурах вычислительных систем. При реализации таких алгоритмов перед программистом встаёт вопрос об оптимальном способе загрузки данных из внешнего хранилища информации, распределении данных между узлами вычислительной системы.

Архитектуры коммуникационных сетей и систем хранения развиваются, обеспечивая с каждым годом увеличение производительности, но также растёт и количество процессорных элементов, которые могут одновременно обращаться за данными во внешнем хранилище. При неэффективном использовании коммуникационных соединений или при неудачном сочетании обращений к жёстким дискам во внешнем хранилище производительность загрузки данных может ухудшаться из-за ограниченности пропускной способности сети или скорости работы жёстких дисков. Прикладному программисту сложно разрабатывать универсальные алгоритмы для всего разнообразия архитектур вычислительных систем.

Авторами данной статьи разрабатывается инструментальная библиотека, которая должна позволить упростить процесс загрузки данных из внешнего хранилища, разделить желаемым образом данные между вычислительными узлами перед началом этапа обработки. При этом от пользователя библиотеки скрываются все вопросы, связанные с использованием архитектурных

особенностей вычислительной системы, в то время как структура разрабатываемой библиотеки позволяет системному программисту дорабатывать и дополнять функциональность, сохраняя общую совместимость различных её частей (загрузка данных, распределение, обработчики данных).

Основные идеи и понятия. Не во всех задачах загрузка входных данных и выгрузка результатов занимает значительную часть времени работы. Проблема производительности операций ввода-вывода ярко выражена в задачах обработки потока данных большого размера, например при обработке последовательности изображений. В таких задачах часто можно встретить использование данных, представимых в виде многомерных массивов. Существует не одно решение вопроса об формате хранения многомерных массивов, но наиболее популярными и признанными в научном мире являются библиотеки, ориентированные на бинарный формат хранения netCDF [1] и HDF [2]. Многие современные архитектурные решения многопроцессорных вычислительных систем предоставляют хранилище данных с возможностью параллельного доступа к файлам (с параллельными файловыми системами GPFS, PVFS, Lustre и т. д.), существуют версии библиотек netCDF и HDF, позволяющие использовать параллельный доступ к частям многомерных массивов. В концепции разрабатываемой библиотеки определено абстрактное понятие «источник данных». Указанные выше файлы на внешнем хранилище в форматах netCDF, HDF можно использовать в качестве источника данных.

В большинстве архитектур многопроцессорных вычислительных систем доступ к внешнему хранилищу неоднороден, это связано с тем, что обычно на несколько вычислительных узлов приходится один узел ввода-вывода. Например, в конфигурации системы Blue Gene/P [3], установленной на факультете вычислительной математики и кибернетики МГУ имени М.В. Ломоносова, 128 вычислительных узлов используют один узел ввода-вывода. Неравномерное распределение между каналами ввода-

вывода приводит к понижению производительности. В статье [4] демонстрируется, что в вычислительных системах с подобной организацией подсистемы ввода-вывода эффективным решением является выделение вычислительных узлов, через которые будут проходить все операции ввода-вывода. Естественным видится, что выделенные вычислительные узлы должны использовать максимальное доступное число каналов ввода-вывода. Вышеописанное приводит ко второму абстрактному понятию – «I/O вычислительные узлы».

Загрузив данные на выделенные вычислительные узлы, необходимо распределить их требуемым образом по вычислительным узлам обработки данных. В случае обработки потока данных загрузка/выгрузка данных является более времязатратной операцией, чем обработка. Эффективным решением является продолжать загрузку/выгрузку данных одновременно с обработкой на вычислительных узлах, для обработки данных соответственно используются узлы, не участвующие в операциях ввода-вывода. Назовём подмножество вычислительных узлов, обрабатывающих данные, областью обработки. В одной программе может быть несколько областей обработки, производящих разную работу над данными.

Схема работы программы в описанных терминах представлена на рисунке.

Схема применения библиотеки. При разработке библиотеки, реализующей описанные выше концепции, одной из целей было предусмотреть удобную возможность дальнейшей доработки. Части библиотеки, отвечающие разным этапам программы, связаны с друг другом только через интерфейс соответствующих классов или параметры функций. В данный момент работа ведётся над MPI-версией библиотеки.

Работа с библиотекой начинается с вызова функции инициализации библиотеки. Реализация функции инициализации может быть архитектурно зависима. Например, для системы Blue Gene/P в момент инициализации библиотеки используются

функции из библиотеки MPIX[5] для определения коммуникаторов, соответствующих множествам вычислительных узлов, использующих один или наоборот разные узлы ввода-вывода. Исходя из полученных коммуникаторов вычисляются возможные I/O вычислительные узлы. Функция инициализации должна вызываться на всех узлах, выделенных в программе, т.е. в рамках коммуникатора MPI_COMM_WORLD.

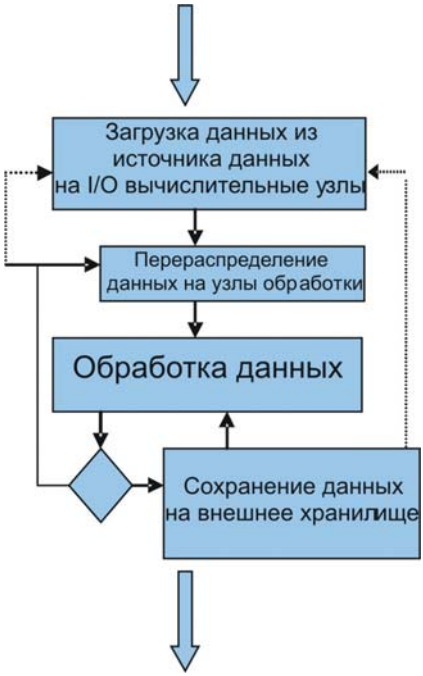


Рис. Схема работы программы в описанных терминах

Дальнейшую работу библиотеки пользователь может продолжить на всех процессорах или указать число процессоров, или коммуникатор, в рамках которого будет происходить работа библиотеки. Пользователю предоставляется возможность создать несколько независимых областей работы библиотеки в рамках подкоммуникаторов.

В рамках каждого коммуникатора библиотеки возможно указывать источники данных. Для этого предусмотрены функции вида «ParimprC_DataLoad_[Type](...)», где в качестве типа источника указывается, например, netCDF или HDF. Системный программист может добавлять источники данных, добавляя функции подобного вида. Данные из источника загружаются на I/O вычислительные узлы, после чего доступны через поля структуры в списке всех источников данных.

С помощью функции «ParimprC_LayoutCreate(...)» может быть выделена область обработки. В зависимости от того, сколько MPI-процессов будет принадлежать области обработки, пользователь может создать множество независимых областей обработки. Системный программист может дорабатывать алгоритм данной функции, изменяя стратегию объединения MPI-нитей в области обработки. В зависимости от стратегии объединения пользователь получит область обработки из соседних или удалённых вычислительных узлов. Указывая область обработки и типы распределения данных между узлами, пользователь с помощью функции «ParimprC_DataPrepare(...)» распределяет данные между вычислительными узлами области обработки.

В рамках области обработки могут быть вызваны функции – обработчики данных. Пользователь может создавать свои обработчики данных или использовать доступные в библиотеке. Обработчик данных работает в рамках MPI-коммуникатора, отвечающего за области обработки.

Разрабатываемая библиотека предназначена для облегчение решения стандартных подзадач обработки данных, связанных с загрузкой и распределением данных между вычислительными узлами. Структура библиотеки позволяет расширять её функциональность, сохраняя при этом разделённость алгоритмов, связанных с загрузкой и распределением данных и их непосредственной обработкой. Изначально библиотека разрабатывалась для задач параллельной обработки изображений и визуализации на суперкомпьютерах, но может быть использована для любой обработки данных, представленных в виде многомерных массивов.

В дальнейшей работе планируется создать набор функций-обработчиков, связанных с обработкой изображений, рассмотреть и учесть архитектурные особенности вычислительных систем СКИФ-МГУ «Чебышев» и «Ломоносов».

Работа ведётся при поддержке Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг.

Список литературы

1. NetCDF Tutorial. – URL: //http://www.unidata.ucar.edu/software/netcdf/docs.

2. HDF5 Specifications. – URL: //http://www.hdfgroup.org/HDF5/doc/Specs.html.

3. Сайт BlueGene /P, установленного в МГУ. – URL: // http://hpc.cs.msu.su.

4. Scalable I/O and analytics / A. Choudhary [et al.] // Journal of Physics: Conference Series, Vol. 180, No. 1. doi: 10.1088/1742-6596/180/1/012048.

5. Библиотека mpix. – URL: // http://www.mpix.com.

[1]М.В. Андреев, [1]Э.Р. Галеев, [1]А.Л. Штангеев, [1]А.В. Юлдашев, [2]А.А. Яковлев

[1]Уфимский государственный авиационный технический университет
[2]ООО «РН-УфаНИПИнефть», г. Уфа

ОПЫТ ПОРТИРОВАНИЯ ГЕОСТАТИСТИЧЕСКОГО СИМУЛЯТОРА НА АРХИТЕКТУРУ GPU СРЕДСТВАМИ ТЕХНОЛОГИИ PGI ACCELERATOR

В последние годы геостатистическое моделирование резервуаров становится все более востребованным компонентом исследований, связанных с оптимизацией стратегии разработки нефтегазовых месторождений. Геостатистика позволяет строить детализированные количественные модели неоднородностей

строения резервуаров в тех их частях, где геофизические параметры ненаблюдаемы и их значения неизвестны.

Недавно были начаты работы по созданию высокопроизводительного симулятора для решения задач геостатистического моделирования. В основу симулятора положен новый метод построения обусловленных геостохастических геологических моделей по скважинным данным с использованием спектрального представления стационарных случайных полей [1]. В рамках проводимых работ была разработана параллельная версия симулятора для традиционных многоядерных вычислительных систем. В нашем докладе представлены результаты портирования симулятора на архитектуру GPU средствами технологии PGI Accelerator.

Одним из перспективных путей снижения времени расчетов является использование графических процессоров (GPU), в частности, производства компании NVIDIA ввиду их высокой пиковой производительности и широкой доступности. В настоящее время наиболее популярной технологией программирования вычислений общего назначения на графических процессорах производства компании NVIDIA является CUDA. Основными преимуществами технологии CUDA являются ее доступность, наличие хорошей документации, набор готовых инструментов, кроссплатформенность [2]. Однако разработка параллельных программ средствами CUDA требует от программиста детального понимания архитектуры GPU. Кроме того, при портировании на архитектуру GPU существующего программного обеспечения вычислительные ядра необходимо существенно перерабатывать.

Отмеченных недостатков лишена более высокоуровневая технология PGI Accelerator, которая поддерживается компиляторами Fortran и C99 компании Portland Group International (PGI) [3]. Данная технология позволяет программировать вычисления на GPU с поддержкой CUDA путем добавления в исходную программу директив препроцессора в стиле OpenMP. Компиляторы с поддержкой PGI Accelerator способны анализировать

структуру программы и данных, на основе указанных программистом директив распределять части программы между CPU и GPU, а также осуществлять отображение итераций циклов *for* на ядра графического процессора.

Таким образом, использование технологии PGI Accelerator не требует значительной переработки исходного кода программного обеспечения и может быть полезным при портировании существующего программного обеспечения. К недостаткам применения технологии PGI Accelerator относительно CUDA можно отнести ограниченность свободного использования компиляторов PGI.

Исходная параллельная версия геостатистического симулятора была реализована на языке C++ с использованием технологии OpenMP. Тестирование производительности OpenMP-версии на различных многоядерных системах на базе процессоров Intel и AMD показало достаточно высокую эффективность параллельной реализации: на 4 потоках ускорение расчета реальных моделей составило от 3,3 до 3,5.

Профилирование исходной версии симулятора показало, что 98 процентов времени расходуется на выполнение четырех функций: двух функций, отвечающих за построение матрицы ковариации (A и B), функции преобразования Фурье (C), а также функции стохастического моделирования (D). В целях портирования на архитектуру GPU с каждой из функций были выполнены следующие преобразования:

- функции были переписаны с языка C++ на C в связи с тем, что технология PGI Accelerator поддерживается только компиляторами языков C и Fortran;
- с помощью директивы *#pragma acc region* в функциях были выделены максимально возможные области акселерации для выполнения на GPU;
- с помощью директивы *#pragma acc data region* и ее спецификаторов были определены массивы данных, которые необходимо передавать на GPU либо получать обратно;
- с помощью директивы *#pragma acc for* и ее спецификаторов были выбраны циклы, выполнение которых переносится

на GPU, а также заданы оптимальные с точки зрения времени выполнения параметры распределения итераций циклов на ядра GPU.

Первоначально портируемые функции компилировались отдельно с помощью компилятора языка C от PGI и компоновались в библиотеку так, чтобы оставшаяся часть кода могла быть скомпилирована альтернативным компилятором языка C++. Однако, для того чтобы получить рабочую многопоточную версию программы, где каждый из потоков OpenMP работает с отдельным GPU, при сборке потребовалось использовать исключительно компиляторы PGI.

Тестирование различных версий симулятора производилось в среде ОС OpenSuSE 10.2 на двухпроцессорной рабочей станции Fujitsu-Siemens Computers CELSIUS V840 на базе двуядерных процессоров Opteron 2214 с двухпроцессорной видеокартой GeForce GTX 295. Сборка исходной OpenMP-версии симулятора под архитектуру CPU осуществлялась компилятором Intel C++ Compiler, сборка полученной версии под архитектуру CPU+GPU – компиляторами PGI C/C++.

Результаты тестирования различных версий симулятора

Платформа	Количество потоков OpenMP	Время выполнения функций, с				
		A	B	C	D	A+B+C+D
CPU	1	235,7	426,6	167,9	384,2	1 214,4
	4	105,6	123,5	42,0	96,7	367,8
CPU+GPU	1	34,4	8,0	7,0	37,0	86,4
	2	17,3	4,1	3,7	18,7	43,7

Результаты тестирования, приведенные в таблице, показывают, что использование GPU позволяет снизить время выполнения функций в 5–30 раз. Суммарное время выполнения портированных функций снижается более чем в 8 раз. Также необходимо отметить, что использование двух процессоров видеокарты дает практически двукратный выигрыш во времени.

В итоге использование технологии PGI Accelerator позволило за достаточно короткое время получить рабочую версию симулятора для выполнения на GPU с поддержкой CUDA. Применение GPU дало существенное ускорение при решении задач геостатистического моделирования.

Список литературы

1. Байков В.А., Бакиров Н.К., Яковлев А.А. Новые подходы к теории геостатистического моделирования // Вестн. УГАТУ. – 2010. – Т. 14, № 2. – С. 209–216.

2. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.

3. PGI Fortran & C Accelerator Programming Model. – URL: http://www.pgroup.com/lit/whitepapers/pgi_accel_prog_model_1.2.pdf

Т.А. Багаутдинов[1], В.П. Гергель[1], А.В. Горшков[1], И.И. Фикс[1,2], М.Ю. Кириллин[1,2]

[1]Нижегородский государственный университет им. Н.И. Лобачевского
[2]Институт прикладной физики РАН, г. Нижний Новгород

МОДЕЛИРОВАНИЕ РАСПРОСТРАНЕНИЯ СВЕТА В МНОГОСЛОЙНОЙ СРЕДЕ МЕТОДОМ МОНТЕ-КАРЛО

В настоящее время лазерные методы получили широкое распространение в бесконтактной неразрушающей диагностике внутренней структуры различных оптически неоднородных объектов, в частности, они находят применение в медицине, биофизике, науках о материалах, физике атмосферы и других областях.

Для повышения эффективности современных методов лазерной диагностики, а также для разработки новых методов необходимо подробное изучение особенностей процесса распространения света в различных средах, включая биоткани.

Однако решение данной задачи затруднено тем, что на данный момент не существует точной теории для описания распространения света в структурно неоднородных средах, а экспериментальные исследования осложнены трудностями поддержания постоянства их структурно-динамических параметров. В связи с этим все большую роль приобретает компьютерное моделирование этого процесса [1]. Оно позволяет более тщательно изучить особенности процесса распространения лазерного пучка в модельных средах, а также исследовать зависимость получаемых результатов от различных параметров измерительной системы и исследуемого объекта, что бывает весьма затруднительно в эксперименте.

Постановка задачи. В рамках данной работы ставится задача моделирования распространения света в многослойной среде. При этом считаем, что задача решается в трехмерном пространстве, а среда состоит из набора плоскопараллельных слоев. Каждый слой является бесконечно широким, описывается толщиной и рядом оптических характеристик.

Результатами моделирования являются значения следующих физических величин:

- пространственное распределение интенсивности рассеянного назад излучения на поверхности среды;
- пространственное распределение интенсивности рассеянного вперед излучения на задней границе среды;
- объемное распределение поглощенной интенсивности в среде.

Практическое решение поставленной задачи с использованием современных процессоров требует значительного времени вычислений, поэтому актуальной является задача ускорения этого процесса.

Общее описание алгоритма решения задачи. Одним из подходов к решению задачи распространения света в многослойных средах является метод статистического моделирования Монте-Карло [1]. Под методом Монте-Карло понимается совокупность приемов, позволяющих получать необходимые реше-

ния при помощи многократных случайных испытаний. Оценки искомой величины выводятся статистическим путем.

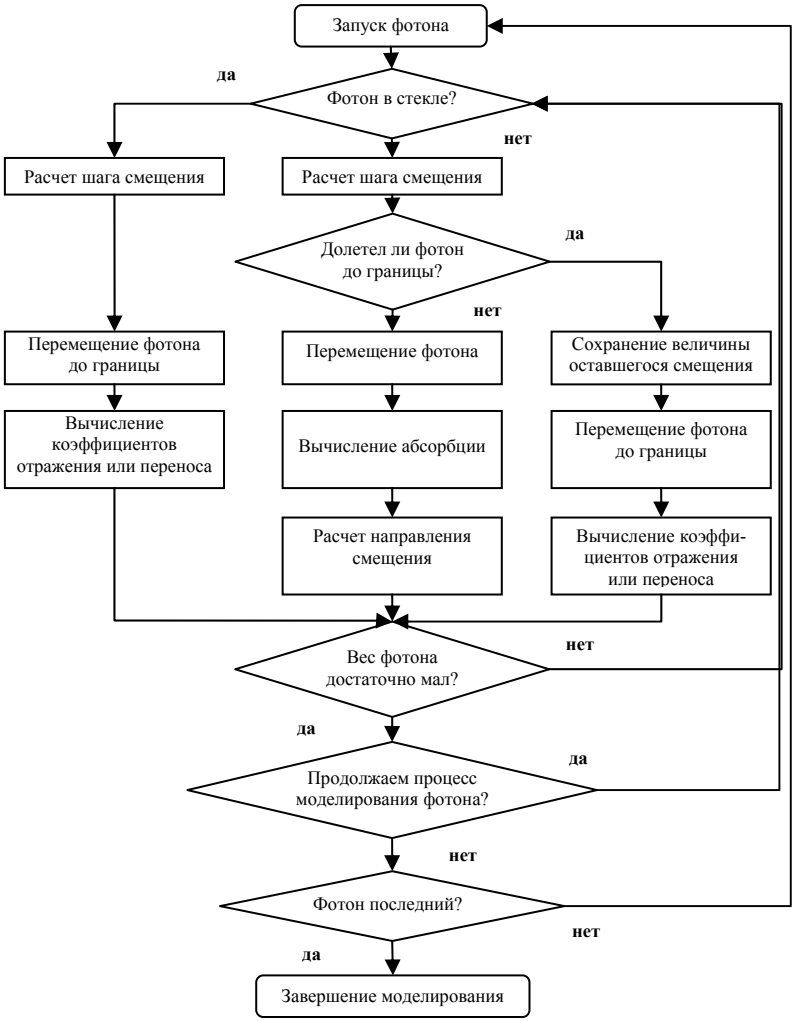


Рис. 1. Алгоритм моделирования распространения света в многослойной среде методом Монте-Карло

Применительно к задаче распространения излучения в многослойной среде метод Монте-Карло заключается в многократном повторении расчета траектории движения фотона в среде, исходя из задаваемых параметров среды.

Общий алгоритм решения задачи представлен на схеме (рис. 1) [1].

Представленный на схеме набор действий выполняется независимо для каждого фотона. Число фотонов в реальных экспериментах может быть достаточно большим ($10^8 – 10^{10}$), из чего можно предположить, что данная задача может быть эффективно реализована при использовании архитектуры графических адаптеров (GPU). В работе описывается реализация представленного выше алгоритма на графическом адаптере с использованием технологии CUDA [2].

Схема распараллеливания и оптимизация. В рассматриваемой задаче необходимые для моделирования действия производятся независимо над каждым фотоном, что позволяет без труда применить схему распараллеливания, при которой каждый поток вычисляет траекторию движения своего фотона. Поскольку число фотонов велико, предложенная схема позволяет загрузить все доступные вычислительные ядра графического адаптера.

Следует отметить, что для повышения производительности каждому потоку имеет смысл проводить вычисления сразу над группой фотонов (в текущей реализации размер группы фотонов равен 1000).

Для обеспечения сходимости метода моделирования распространения света необходимо, во-первых, наличие длинной последовательности различных случайных чисел, во-вторых, организация работы каждого конкретного вычислительного потока со своей подпоследовательностью, причем все эти подпоследовательности не должны пересекаться.

Стандартный генератор случайных чисел языка программирования Си (функция rand) порождает недостаточно длинную последовательность случайных чисел и к тому же не может быть использован при программировании на GPU. Поэтому было принято решение адаптировать алгоритм генератора MCG59 для работы в несколько потоков на графическом адаптере.

В текущей реализации все потоки работают с одной последовательностью случайных чисел, но при этом каждый поток использует свои (*id* + *i* **N*) элементы этой последовательности (здесь *id* – номер потока, *N* – общее число потоков).

Важным условием эффективности программ на GPU является правильное использование имеющихся типов памяти. В частности, для решения данной задачи необходимы следующие наборы данных: информация о среде (хранится в разделяемой памяти), текущие характеристики фотона (хранятся в регистрах) и результирующие данные (имеют большой объем, хранятся в глобальной памяти). Разделяемая память и регистры являются быстрыми типами памяти, но имеют небольшой объем, в связи с чем использовать их для работы с результирующими данными не представляется возможным. Глобальная память, напротив, позволяет хранить большие объемы данных, но является относительно медленной. Однако в нашей задаче обращений к результирующим данным немного, поэтому существенного влияния на производительность использование медленной памяти не оказывает.

Данные, относящиеся к результатам, являются общими для всех фотонов, откуда возникает необходимость обращения к этим данным одновременно из разных потоков, а значит, нужна синхронизация. В текущей реализации для этого используются атомарные операции, работающие с 64-битными целыми числами (для их применения версия compute capability графического адаптера должна быть 1.2 и выше).

Использование механизмов синхронизации обычно сильно уменьшает производительность приложений. Рассматриваемая задача не является исключением, однако здесь все зависит от размерности результирующих массивов, которая является входным параметром алгоритма. При малой размерности (100 элементов в каждом массиве) одновременных обращений к одним и тем же данным много, скорость работы приложения низкая. Но если увеличить число элементов каждого массива до 10 000 (тем самым повысив разрешающую способность задачи), то влияние синхронизации на производительность программы будет минимально.

Дополнительно следует отметить, что для решения требуемой задачи достаточно одинарной точности. А использование операций с одинарной точностью на GPU также существенно ускоряет приложения. К тому же в данном случае возможно использование более быстрых, но менее точных математических функций, предоставляемых встроенной библиотекой CUDA.

Результаты. На рис. 2 приведено время работы программы (в секундах) в зависимости от числа фотонов и вычислительного устройства.

Процессор: Intel Xeon E5520 2.27 ГГц, 4 ядра.
Количество ядер в процессоре: 4.
Количество процессоров: 2.
Объем оперативной памяти: 12 Гб.
Графический адаптер: Nvidia Tesla C1060.

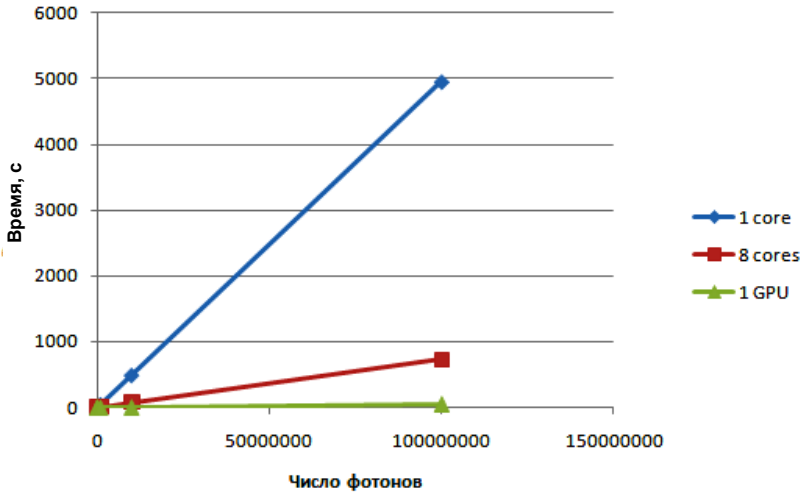


Рис. 2. Время работы алгоритма в зависимости от числа фотонов и вычислительного устройства

Результаты проведенных экспериментов (см. рис. 2) показывают, что использование графического адаптера и технологии CUDA существенно повышает производительность приложения, а потому является вполне обоснованным. Усовершенствование

разработанного программного кода для расчета распространения излучения в средах со сложной геометрией позволит в перспективе эффективно использовать его как для моделирования работы различных систем оптической биомедицинской диагностики, так и для планирования лучевой терапии.

Авторы выражают благодарность Меерову Иосифу Борисовичу и Донченко Роману. Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», проект № 02.740.11.0839.

Список литературы

1. LihongWang, Steven L. Jacques. Monte Carlo Modeling of Light Transport in Multi-layered Tissues in Standard C.1992.

2. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010.

3. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007.

[1]П.И. Балк, [2]А.Г. Деменев, [2]А.С. Долгаль,
[2]О.В. Леденцов, [3]А.В. Мичурин

[1]Германия, г. Берлин
[2]Пермский государственный университет
[3]Горный институт Уральского отделения РАН, г. Пермь

ЭФФЕКТИВНОСТЬ ПРИМЕНЕНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ С ЦЕЛЬЮ ОЦЕНКИ ДОСТОВЕРНОСТИ РЕШЕНИЯ ОБРАТНОЙ ЗАДАЧИ ГРАВИМЕТРИИ

Попытки в 60–70-х годах использовать методы линейного (в том числе и целочисленного) программирования при решении нелинейной обратной задачи гравиметрии (ОЗГ) по оценке геометрии области, занятой массами известной (постоянной) плотности, не увенчались успехом. Причиной тому была плохая со-

вместимость сеточных моделей и классических методов минимизации в вопросах учета разнообразной априорной информации. В 70-х годах отечественными геофизиками проводились масштабные исследования по поиску наиболее эффективных (среди известных) методов решения условно-экстремальных задач для целей минимизации невязки в процессе подбора допустимых решений нелинейной обратной задачи в различных модельных классах, отличных от сеточных. Но отыскать подходящий на все случаи алгоритм (или хотя бы четко очертить классы интерпретационных моделей, для которых оптимальным является тот или иной метод минимизации) так и не удалось [1]. Поэтому обозначилась необходимость разработки проблемно ориентированных методов минимизации, жестко привязанных к специфике обратных задач гравиметрии и магнитометрии.

В работах А.В. Овчаренко [2] и В.Н. Страхова [3] (в соавторстве с М.И. Лапиной) предлагался, по существу, новый принцип структурирования итерационного процесса подбора допустимого решения нелинейной ОЗГ в сеточных классах источников поля, обеспечивающий удобный контроль за соблюдением требования связности и односвязности подобранного приближения к неизвестному носителю аномалиеобразующих масс. В.Н. Страхов предложил называть его монтажным. Конструктивно этот принцип был реализован им в алгоритме регулируемой направленной кристаллизации (РНК). В последующем различные аспекты монтажных алгоритмов затрагивались в работах И.И. Корчагина, У. Шеффера, Т.В. Балк, А.С Долгаля [4, 5]. В цикле работ П.И. Балка метод РНК был обобщен, а сам монтажный подход распространен на постановки обратных задач магнитометрии, случай многосвязной аномалиеобразующей среды с различными значениями плотностей парциальных источников, на задачи совместной интерпретации различных компонент гравитационного и магнитного полей, случай смешанной структурно-рудной задачи.

В этой работе рассматривается метод оценки достоверности решения нелинейной обратной задачи гравиметрии (ОЗГ) на основе гарантированного подхода, позволяющего выделять области геологического пространства, заведомо содержащие ано-

малиеобразующие объекты. Развитие гарантированного подхода к интерпретации поля силы тяжести во многом обусловлено возросшими потребностями геофизической отрасли в оперативной интерпретации больших объемов цифровых данных и тесно связано с возросшими вычислительными возможностями компьютеров. Необходимость применения высокопроизводительных вычислений для решения крупномасштабных задач наблюдается даже в 2D, а тем более в 3D-варианте.

Теория и метод. В самых общих чертах рассматриваемый метод оценки достоверности решения нелинейной обратной задачи гравиметрии (ОЗГ) представляет собой двухуровневый итерационный процесс, во внутреннем цикле которого продуцируются эффективные (с точки зрения решения основной проблемы) допустимые решения ОЗГ, а во внешнем – осуществляется корректировка текущих приближений D_1^* и D_2^* к искомым областям D_1 и D_2, первая из которых содержит аномальные массы, заключенные в неизвестном объеме S^T, а другая целиком заполнена фрагментами аномальных масс: $D_2 \subset S^T \subset D_1$. Если априорная информация содержит (как это всегда и бывает на практике) неопределенность, то итерационный процесс становится трехуровневым, в самом внешнем цикле которого осуществляется переход к все более размытым априорным представлениям о параметрах интерпретационной модели, вносящих указанную неопределенность. Платой за усложнение вычислительного процесса является возможность упорядочения отдельных подобластей из D_1 по вероятности обнаружения в их пределах источников гравитационной аномалии.

Центральное место в технологиях построения пары $< D_1, D_2 >$ занимает собственно алгоритм построения отдельных допустимых решений. От него требуется не только высокая эффективность по быстродействию, но и способность вырабатывать «особые» допустимые решения, небольшое число которых могло бы составить некое репрезентативное подмножество, коллективно обладающее теми же свойствами, что и все множество допустимых решений (являющееся, вообще говоря, бесконеч-

ным). В плане обеспечения приемлемого времени построения отдельно взятого допустимого решения весьма желательно, чтобы алгоритм допускал распараллеливание процесса вычислений, обеспечивающее возможность для использования многопроцессорной вычислительной техники.

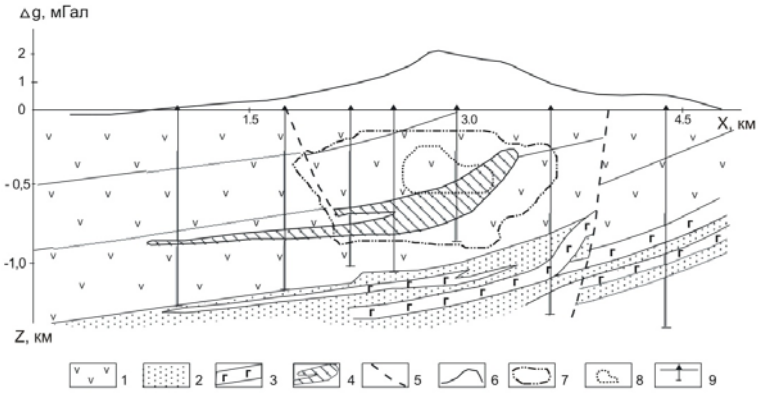


Рис. 1. Результаты интерпретации поля Δg месторождения «Норильск-1»: *1* – породы туфовой толщи; *2* – отложения тунгусской серии; *3* – силлы габро-долеритов; *4* – рудоносная интрузия; *5* – дизъюнктивные нарушения; *6* – локальная составляющая наблюденного поля Δg; *7* – контур области D_1, содержащей все источники локальной аномалии Δg; *8* – контур области D_2, гарантированно содержащей фрагмент аномалиеобразующего объекта; *9* – буровые скважины

Пример применения метода. На рис. 1 приведены результаты решения ОЗГ по данным крупномасштабной гравиметрической съемки, выполненной над месторождением платино-медно-никелевых руд «Норильск-1». В основу модели среды положены следующие допущения: аномалия в основном обусловлена рудоносной интрузией базит-гипербазитового состава, ее избыточная плотность (по отношению к вмещающим породам трапповой формации) составляет 0,2 г/см. С помощью монтажного метода при различных центрах кристаллизации (начальных приближениях) было построено около 400 различных вариантов пространственных распределений масс, удовлетво-

ряющих априорным допущениям. Их синтез позволил выделить область D_2 (рис. 1, контур *8*), с высокой степенью достоверности принадлежащую источнику аномалии при предполагаемом уровне помех $\varepsilon = \pm 0{,}15$ мГал.

Программная реализация и распараллеливание. Представленный подход решения обратной задачи гравиметрии и оценки достоверности результатов ее количественной интерпретации до недавнего времени был программно реализован с использованием системы объектно ориентированного программирования Delphi 7.0 для Windows. Время последовательного счета даже в 2D-варианте с использованием одного процессора типа Intel Core составляет при 100–150 точках задания поля и всего одном возмущающем объекте более часа, а в отдельных случаях (при сложных задачах) – и сутки.

При сжатых сроках выполнения работ требуется применение высокопроизводительных вычислений для решения крупномасштабных задач даже в 2D, а тем более 3D-варианте. На высокопроизводительных многопроцессорных системах Linux установлен значительно чаще, чем Windows. Поэтому важно было обеспечить возможность использования компьютеров с различными операционными системами. Была создана переносимая программная реализация алгоритма оценки достоверности решения ОЗГ в 2D-варианте. Для этой цели использовался кроссплатформенный компилятор FreePascal Compiler и совместимая с ним среда разработки Lazarus.

Решено было использовать распределенные вычисления, которые обычно эффективны для решения вычислительносложных задач, не требующих интенсивного обмена информацией между параллельными подзадачами. Для реализации этих вычислений могут быть использованы средства распределенной вычислительной инфраструктуры программы «Университетский кластер» («УК»)[6]. Участие в этой Программе принимают несколько институтов РАН и многие российские вузы, в том числе и Пермский госуниверситет. Parallel Compute: MPI-сервис уровня инфраструктуры программы «УК», который обеспечивает

разработку и выполнение MPI-программ для вычислительных систем с распределенной памятью. Поэтому предложена схема распараллеливания вышеуказанной программы на основе MPI.

Одной из характеристик эффективности параллельного алгоритма является ускорение – отношение между временем выполнения задания с данными размера *m* асимптотически оптимальным последовательным алгоритмом и временем выполнения того же задания параллельным алгоритмом на машине с *p*-процессорами. Для априорной оценки ускорения применена сетевая формула Амдала [7], которая учитывает потери времени на межпроцессорный обмен данными в параллельном приложении.

$$S_p(m) = \frac{1}{D + f + (1 - f)/p},$$

где $f = f(n)$ – доля последовательных операций, $D = D_{alg} D_{tech}$ – коэффициент сетевой деградации. При этом D_{alg} определяет алгоритмическую составляющую коэффициента деградации, обусловленную свойствами алгоритма, а D_{tech} – техническую составляющую, зависящую от соотношения технического быстродействия процессора и аппаратуры сети. Далее, $D_{alg} = W_{comm} / W_{comp}$, где W_{comm} – количество операций передачи данных, а W_{comp} – количество вычислительных операций. $D_{tech} = t_{comm} / t_{comp}$, где t_{comm} – среднее время выполнения одной операции передачи данных, t_{comp} – среднее время одной вычислительной операции.

Рассмотрена типичная 2D-задача размером $m = 19200$ монтажных элементов. Оценено соответствующее типичное соотношение последовательной и параллельной части алгоритма $f = 2{,}8 \cdot 10^{-5}$. На учебном вычислительном кластере компьютерного центра механико-математического факультета ПГУ оценен соответствующий коэффициент сетевой деградации $D = p \cdot 1{,}9 \cdot 10^{-6}$. Оценка показывает, что применение высокопроизводительных вычислительных кластеров позволяет кардинально

снизить время расчета. В результате распараллеливания программы время счета удается существенно сократить (в примере до 340 раз). Дальнейшее увеличение количества вычислительных ядер (процессоров) сверх оптимального (в примере – 512) лишь замедляет работу программы, так как накладные расходы на передачу информации между процессорами кластера начинают превосходить выигрыш за счет распараллеливания (рис. 2).

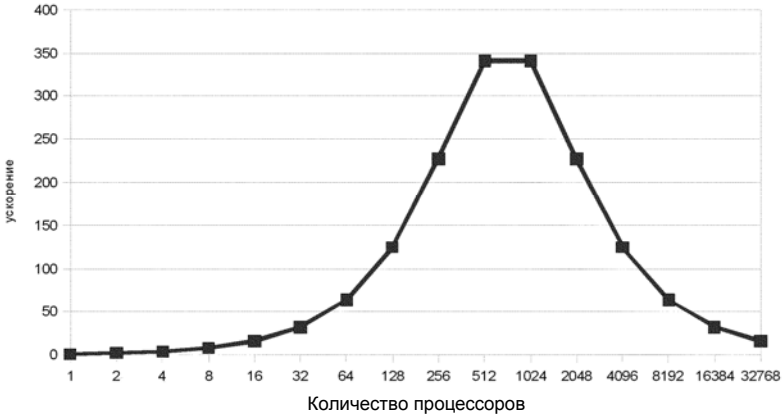


Рис. 2. Зависимость ускорения от числа вычислительных ядер (процессоров)

Создана переносимая программная реализация алгоритма оценки достоверности решения ОЗГ в 2D-варианте. Предложена схема распараллеливания вышеуказанной программы на основе MPI. Априорная оценка ускорения с помощью сетевой формулы Амдала показывает, что применение высокопроизводительных вычислительных кластеров позволяет кардинально снизить время расчета. Оценка достоверности результатов количественной интерпретации в терминах пары множеств D_1, D_2 может существенно повысить вероятность вскрытия искомых аномалиеобразующих объектов в заданных интервалах глубин поисковыми и разведочными скважинами, рекомендованными по гравиметрическим данным. Актуальным в области гарантированного подхода будет проведение исследований по следующим направлениям:

1. Параллельные реализации предложенного способа для 3D-задач.

2. Разработка алгоритма монтажного метода в модификации регулируемой направленной раскристаллизации для 2D- и 3D-задач.

3. Проектирование и реализация в инфраструктуре программы «Университетский кластер» проблемно ориентированного вебсервиса для автоматизации интерпретации результатов гравиметрии.

Работа поддержана в рамках проекта «Развитие центра коллективного пользования высокопроизводительными вычислительными ресурсами» Программы развития национального исследовательского университета ПГУ.

Список литературы

1. Гольдшмидт В.И. Оптимизация процесса количественной интерпретации данных гравиразведки. – М.: Недра, 1984. – 184 с.

2. Овчаренко А.В. Подбор сечения двухмерного тела по гравитационному полю. Вопросы нефтяной и рудной геофизики. – Алма-Ата, 1975. Вып. 2. – С. 71–75.

3. Страхов В.Н., Лапина М.И. Монтажный метод решения обратной задачи гравиметрии // Докл. АН СССР. – 1976. – Т. 227, № 2. – С. 344–347.

4. Балк П.И. Использование априорной информации о топологических особенностях источников поля при решении обратной задачи гравиметрии // Докл. АН СССР. – 1989. – № 5. – С. 1082–1084.

5. Балк П.И. Долгаль А.С., Балк Т.В. Сеточные модели плотностной среды и опыт их применения при прослеживании дифференцированных интрузий по данным гравиразведки // Геология и геофизика. – 1993. – № 5. – С. 127–134.

6. Программа «Университетский кластер». – URL: http://www.unicluster.ru.

7. Деменев А.Г. Анализ параллельных вычислительных алгоритмов: учеб.-метод. пособие; Перм. ун-т. – Пермь, 2007. – 43 с.

[1,2]Н.В. Барановский

[1]Томский политехнический университет
[2]Научно-исследовательский институт прикладной математики и механики
Томского государственного университета

О ПОДГОТОВКЕ СПЕЦИАЛИСТОВ ПО ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ ДЛЯ НУЖД МЧС

Несмотря на интенсивный характер развития науки и техники, катастрофические явления и в настоящее время наносят экономический, экологический ущерб и ведут к гибели людей. Ключевым моментом в решении этой проблемы является разработка мер по прогнозу и предупреждению чрезвычайных ситуаций природного и техногенного характера. Интенсификация человеческой деятельности привела к увеличению антропогенной нагрузки на лесопокрытые территории. Как следствие, лесные пожары из природного регулирующего фактора превратились в катастрофическое явление [1]. В последнее время широкое применение в области охраны лесов от пожаров находят информационно-вычислительные технологии. Большую роль начинают играть технологии математического моделирования процессов лесопожарного созревания, сушки и зажигания лесных горючих материалов (ЛГМ). Стремительное развитие микроэлектроники позволяет создавать системы с параллельной обработкой данных различного масштаба: от многоядерного процессора до многопроцессорного суперкомпьютера. Появилась возможность и необходимость разработки параллельных программных средств в области прогноза и мониторинга лесных пожаров. Например, разработан подход ландшафтного распараллеливания для прогноза лесной пожарной опасности [2], параллельная реализация модели атмосферы MM5 используется для получения прогнозных значений метеорологических полей [3], параллельные вычислительные технологии используются при моделировании процесса распространения лесного пожара [4, 5]. При разработке параллельных программ необходимы специализированные инструменты программирования, поддающиеся параллельной реализации численные ме-

тоды, наличие естественного параллелизма в решаемой задаче и способность разработчика мыслить параллельными категориями [6]. Необходима разработка нового или адаптация существующего подхода для осуществления полного цикла разработки и поддержки программного продукта для высокопроизводительных систем. Следует выделить технологии такого сорта от компании «Майкрософт» [7]. Цель работы – адаптация и внедрение в учебный процесс технологий создания программных продуктов «Майкрософт» для проектирования, разработки, модернизации и технической поддержки информационно-вычислительного параллельного комплекса прогноза лесной пожарной опасности.

Синтез технологии проектирования и разработки ППК. Специалисты «Майкрософт» считают, что при создании приложений и развитии инфраструктуры надо ориентировать на базовые концепции уровня предприятия и принципы разработки приложений Microsoft Solution Framework (MSF) [7], поскольку применение метода MSF позволяет получать комплексные, итерационные, работоспособные решения, приоритеты которых четко определены. Одно из ключевых понятий – производственная архитектура, которая представляет собой скоординированный, единый технологический план. Цель разработки архитектуры формулируется следующим образом [7]: выработать логически связанный, цельный план рутинных работ и скоординированных проектов, необходимых для преобразования сложившейся структуры информационных систем и приложений организации к состоянию, определенному как долгосрочная цель на основе текущих и перспективных задач и процессов. Модель производственной архитектуры предполагает итерационный, поэтапный выпуск серии последовательных версий, по мере выпуска которых производственная архитектура постоянно корректируется. Однако по большей части технологии «Майкрософт» направлены на разработку бизнес-приложений. В рассматриваемом случае речь идет о разработке ППК на базе наукоемкой информационно-вычислительной «начинки». Каждая итерация в рамках коррекции производственной архитектуры должна содержать вложенный итерационный процесс разрешения научной задачи посред-

ством математического моделирования. Шаг 1 – формулировка модели, шаг 2 – формализация модели на языке математических символов, шаг 3 – программная реализация модели и проведение вычислительных экспериментов, шаг 4 – анализ и верификация результатов, шаг 5 – получение новых знаний об исследуемом явлении. Затем переход на шаг 1 и повтор процесса построения и исследования модели [8].

Разработка, поддержка и сопровождение ППК прогноза лесной пожарной опасности. Разработка ППК должна проводиться проектной группой, которая в дополнение к модели «Майкрософт» должна содержать научно-исследовательский компонент: специалист в предметной области (лесной пиролог), специалист-вычислитель, специалист-физик. В данном случае можно ожидать, что в запланированный срок будет разработана стабильная версия ППК с относительно оптимальными характеристиками производительности и программной реализацией математической модели, адекватной физике исследуемого процесса. Менеджер продукта вместе с менеджером программы должен прийти к компромиссному решению относительно функциональных возможностей продукта, сроков его разработки и финансирования. В небольших организациях или при работе над мелким проектом роли можно совмещать. Однако в этом случае существует другая проблема – как не упустить из виду ни одной существенной детали проекта с точки зрения каждой роли. При работе над крупными проектами, наоборот, приходится делить проектную группу на тематические и функциональные подгруппы [7]. Например, в рассматриваемом случае могут быть выделены подгруппы, отвечающие за разработку модулей сушки слоя ЛГМ, зажигания ЛГМ антропогенными и природными источниками. Разделение проектной группы на подгруппы соответствует методологии корпорации Microsoft выделения функциональных групп, которые формируются в рамках одной роли. Выпуск каждой версии ППК должен заканчиваться переходом к уточнению производственной архитектуры и спецификации последующей версии.

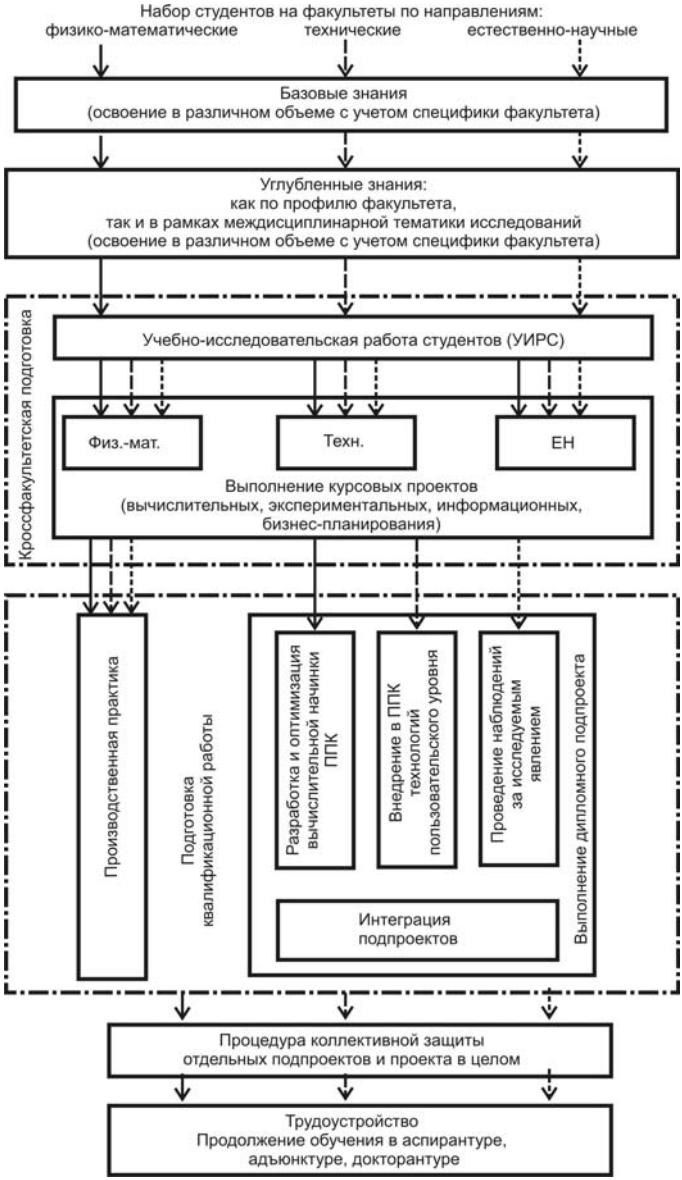


Рис. Схема типичного учебного процесса (сокращение ЕН означает специалист в области естественных наук)

Перспективы подготовки высококвалифицированных кадров. Следует внедрять принципы группового проектирования и разработки программных продуктов в учебный процесс, примерная схема которого представлена на рисунке. Групповая работа над дипломным проектом (возможна как с фиксированным распределением ролей, так и с их перестановкой в ходе работы). Выполнение дипломного проекта предваряет серия курсовых работ по нескольким дисциплинам (в том числе и спецкурсам). Возможен вариант выполнения проекта студентами разных факультетов и разных специализаций. Процедура защиты такого проекта также должна быть коллективной и носить характер семинара, когда каждый из исполнителей защищает свою часть проекта. В начале и конце выступлений ответственный исполнитель озвучивает информацию относительно объединяющей части и роли каждого из подпроектов.

В обязательном порядке на факультетах, имеющих отношение к разработке информационно-вычислительного программного обеспечения, следует внедрить в учебный процесс спецкурс «Основы современных технологий проектирования и разработки программных продуктов». Цель предлагаемого спецкурса заключается в подготовке специалиста, понимающего и способного проектировать и разрабатывать программное обеспечение, отвечающее современным требованиям. Примерная программа такого спецкурса приведена ниже. Автор разработал и использовал программу этого спецкурса при проведении занятий со студентами 3-го курса механико-математического факультета Томского государственного университета.

Примерная программа спецкурса «Основы современных технологий проектирования и разработки программных продуктов»

Введение

(Обзорная информация по технологическому прогрессу аппаратного и программного обеспечения. Типы программного обеспечения.) Объем – 2 часа.

Раздел 1. Современные технологии разработки программных продуктов

(Языки и средства разработки. RAD- и CASE-системы. Машинная графика и разработка 3D-приложений). Объем – 2 часа

Раздел 2. Проектирование и разработка крупных программных продуктов

Производственная архитектура – 2 часа;

Приложения масштаба предприятия – 4 часа;

Проектные группы – 2 часа;

Процесс разработки – 4 часа;

Предпроект – 2 часа;

План проекта – 2 часа.

Объем – 16 часов.

Раздел 3. Современные информационно-вычислительные технологии

(Технологии пользовательского и прикладного уровня. Технологии уровня данных. 2D, 3D-графика и визуализация. Последовательные и параллельные вычислительные технологии. Экспертные системы. Географические информационные системы. Тестирование и производственный цикл. Защита приложения. Стадия «Разработка» и ее результаты.) Объем – 10 часов.

Раздел 4. Выпуск финальной версии продукта

(Стабилизация продукта. Выпуск продукта. Обсуждение продукта) Объем – 4 часа.

Заключение

(Подведение итогов). Объем – 2 часа.

Итого 36 часов. Форма отчетности – зачет.

Примерная структура билета:

Вопрос 1. Современные подходы к проектированию программных продуктов.

Вопрос 2. Средства разработки программных продуктов.

Вопрос 3. Характеристика информационно-вычислительных технологий.

В настоящей работе представлена стратегия обеспечения полного цикла разработки информационно-вычислительного

программного обеспечения для высокопроизводительных систем параллельной архитектуры. В качестве примера приведен алгоритм разработки параллельного программного комплекса прогноза лесной пожарной опасности. Представленный принцип разработки ППК является достаточно общим и может применяться практически в любой предметной области, где требуется обработка больших объемов данных в режиме, опережающем реальное время развития процесса.

Список литературы

1. Кузнецов Г.В., Барановский Н.В. Прогноз возникновения лесных пожаров и их экологических последствий. – Новосибирск: Изд-во СО РАН, 2009. – 301 с.

2. Барановский Н.В. Ландшафтное распараллеливание и прогноз лесной пожарной опасности // Сибирский журнал вычислительной математики. – 2007. – Т. 10, № 2. – С. 141–152.

3. Evaluation of MM5 model resolution when applied to prediction of National Fire Danger Rating indexes / J.L. Hoadley [et. al.] // International Journal of Wildland Fire. – 2006. – Vol. 15. No. 2. – P. 147–154.

4. Барановский Н.В. Основные принципы параллельной реализации общей математической модели лесного пожара // Пожарная безопасность. – 2008. – № 1. – С. 98–102.

5. Вдовенко М.С., Доррер Г.А. Разработка параллельных алгоритмов, моделирующих распространение лесных пожаров // Параллельные вычислительные технологии (ПаВТ2009): тр. междунар. науч. конф. – Челябинск: Изд-во ЮУрГУ, 2009. – С. 420–426.

6. Малышкин В.Э., Корнеев В.Д. Параллельное программирование мультикомпьютеров: учеб. пособие. – Новосибирск: Изд-во Новосиб. гос. техн. ун-та, 2006. – 295 с.

7. Уилсон С.Ф., Мейплс Б., Лэндгрэйв Т. Принципы проектирования и разработки программного обеспечения: учеб. курс MCSD: пер. с англ. – М.: Русская редакция, 2000. – 608 с.

8. Прохоров Ю.В. Математический энциклопедический словарь. – М.: Советская энциклопедия, 1988. – 845 с.

К.А. Баркалов, В.В. Рябов, С.В. Сидоров

Нижегородский государственный университет им.Н.И. Лобачевского

ОЦЕНКИ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОГО ИНДЕКСНОГО МЕТОДА ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ[1]

Данная работа продолжает развитие известного подхода к минимизации многоэкстремальных функций при невыпуклых ограничениях, описанного в работах [1–5] и получившего название *индексного метода* глобальной оптимизации. Подход основан на раздельном учете каждого ограничения задачи и не связан с использованием штрафных функций. При этом решение многомерных задач сводится к решению эквивалентных им одномерных. Соответствующая редукция основана на использовании кривых Пеано (называемых также развертками Пеано), однозначно отображающих единичный отрезок вещественной оси на гиперкуб, а также их обобщений («вращаемые развертки»), которые можно применять при решении задачи на кластерных системах с десятками и сотнями процессоров. Приведены результаты экспериментов, позволяющие оценить эффективность параллельного индексного метода. Эксперименты выполнены на вычислительном кластере ННГУ им. Н.И. Лобачевского, установленном в ходе выполнения нацпроекта «Образование».

Постановка задачи. Рассмотрим задачу глобальной оптимизации вида

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

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

где целевая функция $\varphi(y)$ (в дальнейшем обозначаемая также $g_{m+1}(y)$) и левые части ограничений $g_j(y), 1\le j\le m$, удовлетворяют условию Липшица с соответствующими константами $L_j, 1\le j\le m+1$, а именно

[1] Работа выполнена при поддержке совета по грантам Президента Российской Федерации (грант № МК-1536.2009.9).

$$|g_j(y_1)-g_j(y_2)| \le L_j|y_1-y_2|,\ 1\le j\le m+1,\ y_1,y_2\in D.$$

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

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

исходную задачу можно редуцировать к следующей одномерной задаче:

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

Рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией и липшицевыми ограничениями одномерную задачу, в которой соответствующие функции удовлетворяют равномерному условию Гельдера [1], т.е.

$$|g_j(y(x'))-g_j(y(x''))|\le K_j|x'-x''|^{1/N},\ x',x''\in[0,1],\ 1\le j\le m+1,$$

где N – размерность исходной многомерной задачи, а коэффициенты K_j связаны с константами Липшица L_j исходной задачи соотношениями $K_j\le 4L_j\sqrt{N}$.

Различные варианты индексного алгоритма для решения одномерных задач и соответствующая теория сходимости представлены в работах [3], [5].

Использование множественных отображений. Редукция многомерных задач к одномерным с помощью разверток имеет такие важные свойства, как непрерывность и сохранение равномерной ограниченности разностей функций при ограниченности вариации аргумента. Однако при этом происходит потеря части информации о близости точек в многомерном пространстве, так как точка $x\in[0,1]$ имеет лишь левых и правых соседей, а соответствующая ей точка $y(x)\in R^N$ имеет соседей по 2^N направлениям. А при использовании отображений типа кривой Пеано близких в N-мерном пространстве образам y', y'' могут соответствовать достаточно далекие прообразы x', x'' на отрезке [0,1]. Как результат, единственной точке глобального минимума в многомерной задаче соответствует несколько (не более 2^N) локальных экстремумов в одномерной задаче, что, естественно, ухудшает свойства одномерной задачи.

Сохранить часть информации о близости точек позволяет использование множества отображений

$$Y_L(x)=\{y^1(x),\ldots,\ y^L(x)\}$$

вместо применения единственной кривой Пеано $y(x)$ [2], [4]. Каждая кривая Пеано $y^i(x)$ из $Y_L(x)$ может быть получена в результате некоторого сдвига вдоль главной диагонали гиперинтервала D. Таким образом сконструированное множество кривых Пеано позволяет получить для любых близких образов y', y'', отличающихся только по одной координате, близкие прообразы x', x'' для некоторого отображения $y^i(x)$.

Вращаемые развертки. К числу недостатков ставшей уже классической схемы построения множественных разверток (далее будем называть их сдвиговыми развертками или С-развертками) можно отнести, во-первых, наличие дополнительного ограничения, порождающего сложную допустимую область на одномерных отрезках. А во-вторых, при построении С-разверток число разверток L (а следовательно, и число параллельно решаемых задач) зависело от требуемой точности ε поиска решения задачи. С позиции экономии вычислительных ресурсов было невыгодно использовать число разверток большее, чем $\lceil \log_2(\varepsilon^{-1}) \rceil$. Например, при решении задачи с точностью 10^{-3} по координате целесообразно было выбирать число разверток не больше 10.

Преодолеть эти недостатки, сохранив информацию о близости точек в N-мерном пространстве, позволяет новая схема построения множественных отображений. Отличительной чертой предложенной схемы является построение множества кривых Пеано не с помощью сдвига вдоль главной диагонали гиперкуба, а поворотом развертки вокруг начала координат. При этом найдется отображение $y^i(x)$, которое точкам многомерного пространства y', y'', которым при исходном отображении соответствовали достаточно далекие прообразы на отрезке [0,1], будет сопоставлять более близкие прообразы x', x''. Развертки, порождаемые в соответствии с новой схемой, будем называть вращаемыми развертками или В-развертками.

Максимальное число различных поворотов развертки, отображающей N-мерный гиперкуб на одномерный отрезок, составляет 2^N. Использование всех из них является избыточным, требуется выбрать лишь часть из всех возможных вариантов. В предложенной схеме преобразование развертки осуществляется в виде поворота на угол $\pm\pi/2$ в каждой из координатных плоскостей. Число подобных пар поворотов определяется числом координатных плоскостей пространства $C_N^2 = N(N-1)/2$, а общее число преобразований будет равно $N(N–1)$. Учитывая исходное отображение, приходим к заключению, что данный способ позволяет строить до $N(N–1)+1$ развертки для отображения N-мерной области на соответствующие одномерные отрезки. При этом дополнительное ограничение, которое возникало при построении С-разверток [4], отсутствует. В случае необходимости данный способ построения множества отображений может быть легко «отмасштабирован» для получения большего (вплоть до 2^N) числа разверток.

Параллельный индексный метод. Использование множества отображений $Y_L(x)=\{y^1(x),\ldots,\ y^L(x)\}$ приводит к формированию соответствующего множества одномерных многоэкстремальных задач

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

Каждая задача из данного набора может решаться независимо, при этом любое вычисленное значение $z=g_\nu(y')$, $y'=y^i(x')$ функции $g_\nu(y)$ в i-й задаче может интерпретироваться как вычисление значения $z=g_\nu(y')$, $y'=y^s(x'')$ для любой другой s-й задачи без повторных трудоемких вычислений функции $g_\nu(y)$. Подобное информационное единство позволяет решать исходную задачу (1) путем параллельного решения индексным методом L задач вида (3) на наборе отрезков [0,1]. Каждая одномерная задача решается на отдельном процессоре. Для организации взаимодействия на каждом процессоре создается L очередей, в которые процессоры помещают информацию о выполненных итерациях. Используемая схема не содержит какого-либо единого управляющего процессора, что увеличивает надежность выполняемых вычислений.

Подробное описание решающих правил параллельного индексного алгоритма глобальной оптимизации приведено в работе [7].

Результаты экспериментов. Один из известных подходов к оценке эффективности методов безусловной глобальной оптимизации основан на численном решении этими методами всех задач из некоторой случайно генерируемой выборки большого объема. Генераторы таких выборок можно рассматривать как некоторые классы функций $f(y)$, $y \in D$, с определенной на них вероятностной мерой.

Среди подобных генераторов следует отметить используемый в данной работе GKLS-генератор, описанный в [7] и доступный для свободного скачивания. Отличительной особенностью GKLS-генератора является прозрачный способ задания сложности генерируемых задач: параметрами генератора являются количество локальных минимумов задачи, размеры их областей притяжения, удаленность глобального минимума от локальных и многое другое. Всё это определяет востребованность GKLS-генератора, который используется более чем в 40 странах для тестирования методов глобальной оптимизации. Для сравнения методов использовались описанные в [8] несколько классов тестовых задач по 100 функций каждый. Область поиска D во всех задачах составляет гиперкуб $[-1.0, 1.0]^N$.

Примененная процедура оценки основана на операционных характеристиках [4] и состоит в следующем. Пусть некоторая задача из рассматриваемой выборки решается с помощью алгоритма S. При этом задаче сопоставляется порождаемая алгоритмом последовательность точек испытаний $\{y^k\}$. Указанная последовательность усекается (т.е. процесс решения прекращается) либо в связи с первым попаданием точки очередного испытания в заданную δ-окрестность решения y^*, либо в связи с тем, что в ходе выполнения заданного числа $K_{\max}$ испытаний такое попадание не имело места. В проведенных численных экспериментах использовалось значение $K_{\max}=10\,000$.

Результат решения всех задач выборки с помощью алгоритма S представляется функцией $P_S(k)$, характеризующей до-

лю задач, для которых в ходе k шагов поиска имели место попадания точек испытаний в заданную δ-окрестность решения. Такую функцию будем называть операционной характеристикой алгоритма S. Отметим, что при использовании GKLS-генератора δ-окрестность решения задачи заранее известна.

Эксперименты проводились для двух реализаций алгоритма с вращаемыми развертками: *последовательной* и *параллельной* при плотности $m = 10$, числе разверток $L = 6$. При этом использовались параметры надежности $r = 3{,}8$ (данные параметры являются минимальными для соответствующих методов, при которых достигается решение 100 % задач из выборки); точность попадания в окрестность решения составляла $\delta = 0{,}01$.

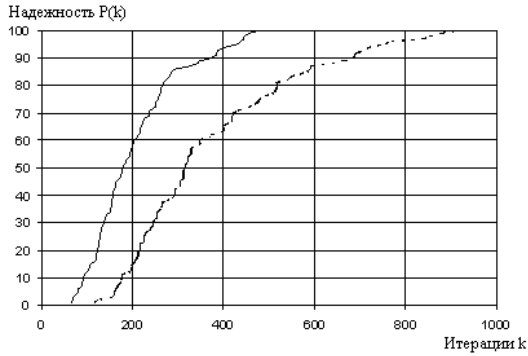


Рис. 1. Операционные характеристики
для параллельного метода

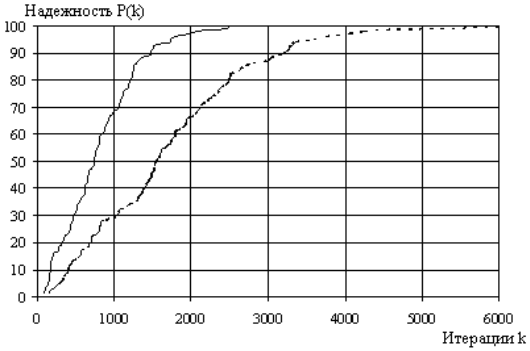


Рис. 2. Операционные характеристики
для последовательного метода

Операционные характеристики для обоих методов, полученные на некоторых классах, порожденных GKLS-генератором, представлены соответственно на рис. 1 и 2. При этом нижняя разрывная кривая характеризует *последовательный* метод, а верхняя непрерывная – *параллельный* метод с вращаемой разверткой. Указанное расположение кривых показывает, что параллельный алгоритм с вращаемыми развертками обеспечивает в среднем значительно более быстрое получение оценок, лежащих в заданной окрестности решения, чем его последовательный прототип.

В проведенных экспериментах доля правильно решенных задач определяет *надежность поиска решения*, а число выполненных испытаний целевой функции – *затратность при поиске решения*. Для последовательного метода число итераций k совпадает с количеством испытаний K, а при применении параллельной версии $K \approx k \cdot L$.

Отметим, что оценка по итерациям косвенно характеризует выигрыш по времени от применения параллельных реализаций. Таким образом, выигрыш по времени на первом классе функций (при условии правильного решения всех задач) примерно равен 1,9 раза, а на втором – он составил 2,4 раза, т.е. коэффициент ускорения возрастает с ростом сложности задач. Аналогичный эффект наблюдается при увеличении размерности (отчасти – за счет возможности увеличения коэффициента распараллеливания, совпадающего с числом используемых разверток L).

Дадим оценки избыточности. Анализируя результаты, приведенные на рис. 1, можно оценить общее количество вычислений функций, выполненных параллельной версией, исходя из общего числа итераций (оно составило 475), необходимого для решения 100 % задач. Общее число измерений функции (в худшем случае) при этом оценивается как $475L = 28500$ на задачу. При этом для полного решения всех задач в худшем случае последовательной версии потребовалось 900 итераций. Таким образом, коэффициент избыточности по количеству испытаний при $L = 6$ за счет параллелелизма может быть оценен

как 28500/900 = 3,17. Аналогичный способ оценивания для второго эксперимента (см. рис. 2) дает оценку избыточности, равную 2500*L*/6000 = 2,5.

Укажем также абсолютное время, затраченное программной реализацией алгоритма на поддержку собственного функционирования в условиях суперкомпьютерных вычислений (без учета времени вычисления функций). Оно составило в проведенных экспериментах (в среднем на одну задачу) 0,6 с на каждый процессор. Общее время вычислений зависит от затрат на одно вычисление функции, т.е. определяется их вычислительной сложностью. Указанные временные характеристики соответствуют следующим показателям использованного оборудования. При проведении экспериментов использовался высокопроизводительный кластер, приобретенный Нижегородским университетом в 2007 году, с двухпроцессорными двухъядерными серверами Intel Xeon 3.2 GHz, число разверток соответствовало числу вычислительных ядер. Размер оперативной памяти, выделяемой каждому ядру составлял 1 Гб.

В заключение можно отметить, что предложенная схема построения множества кривых Пеано (B-развертки) ориентирована на кластерные системы с большим числом вычислительных узлов. Проведены вычислительные эксперименты, результаты которых позволяют оценить эффективность параллельного индексного метода. Эксперименты проведены с помощью параллельной системы глобальной оптимизации GloblExpert, разработаной в ННГУ им. Н.И. Лобачевского.

Список литературы

1. Стронгин Р.Г. Поиск глобального оптимума. – М.: Знание, 1990.

2. Стронгин Р.Г. Параллельная многоэкстремальная оптимизация с использованием множества разверток// Вычисл. матем. и матем. физ. – 1991. – Т. 31, № 8. – С. 1173–1185.

3. Стронгин Р.Г., Баркалов К.А. О сходимости индексного алгоритма в задачах условной оптимизации с ε-резервированн-

ными решениями // Математические вопросы кибернетики. – М.: Наука, 1999. – С. 273–288.

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

5. Баркалов К.А. Ускорение сходимости в задачах условной глобальной оптимизации. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2005.

6. Баркалов К.А., Сидоров С.В., Рябов В.В. Параллельные вычисления в задачах многоэкстремальной оптимизации // Вестн. ННГУ. Математическое моделирование и оптимальное управление. – Н. Новгород: Изд-во Нижегород. гос. ун-та. – 2009. – № 6(1) – С. 171–177.

7. Software for generation of classes of test functions with known local and global minima for global optimizations / M. Gaviano [et al.] – URL: http://si.deis.unical.it/~yaro/GKLS.html // ACM TOMS. – 2003. – Vol. 29, No. 4. – P. 469–480.

8. Lera D., Sergeyev Ya.D. Lipschitz and Hölder global optimization using space-filling curves // Applied Numerical Mathematics. – January 2010. – Vol. 60, No. 1–2, P. 115–129.

А.А. Безгодов, С.В. Иванов, С.С. Косухин

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

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ ЭКСТРЕМАЛЬНОЙ ДИНАМИКИ МОРСКИХ ПЛАВУЧИХ ОБЪЕКТОВ

Моделирование динамики морских плавучих объектов (судов и объектов океанотехники) в экстремальных условиях эксплуатации является актуальной задачей, тесно связанной с обеспечением безопасности мореплавания. Это обусловлено тем, что в сложных погодных условиях статические характери-

стики морских объектов (например, остойчивость, характеризуемая метацентрической высотой) становится невозможным измерить прямыми методами, а их динамические аналоги изменяются нерегулярным образом в силу воздействия на объект внешних сил и моментов стохастической природы, порождаемых ветром, морским волнением и течениями [1]. Это серьезно усложняет задачу исследования динамики корабля и требует применения для ее решения комплексного подхода, включающего последовательное определение характеристик внешних воздействий (ветра, волнения, течений), моделирование вызванных ими возмущающих сил и моментов стохастической природы, а также моделирование нелинейных колебаний объекта в условиях нестационарных изменений внешних условий, неопределенности параметров и пр.

Анализ гибели судов в различных условиях эксплуатации позволяет выделить условия, которые способствуют возникновению экстремальных ситуаций. В частности, при движении судов лагом к волне выделяют такие ситуации, как воздействие на судно ветрового шквала в условиях сильной качки, потеря остойчивости по причине затопления палубного колодца, а также ударное воздействие гребня разрушающейся волны. Напротив, при попутном волнении начинают играть роль совершенно другие критические факторы, а именно: параметрический резонанс бортовых колебаний, потеря управляемости на гребне волны, захват судна волной (брочинг). В свою очередь, развитие каждой из вышеперечисленных ситуаций может усложняться за счет внутренних факторов, определяющих характеристики самого объекта. К ним относятся, например, смещение навалочного груза, затопление отсеков, интенсивное обледенение [2].

Дискретность набора экстремальных ситуаций и разнородность определяющих их факторов приводит к тому, что в настоящее время для компьютерных экспериментов в области динамики корабля не существует единой (универсальной) модели, одинаково удобной во всех случаях. Напротив, на практике используется семейство разнородных численных моделей (и соответствующих им программных решений), предназначенных

для исследования отдельных задач безопасности мореплавания. Как следствие, это порождает проблему интерпретации результатов, полученных по разным моделям для одного и того же явления, а также обоснования адекватности той или иной модели для конкретной практической задачи. При этом разнообразие применяемых моделей косвенно связано с ресурсоемкостью вычислений и возможностью использования высокопроизводительных вычислительных платформ. Так, например, в большинстве случаев для инженерных расчетов используются достаточно простые спектральные модели, или модели, основанные на прямом интегрировании упрощенных уравнений механики твердого тела, поскольку применение гидродинамических моделей часто требует от пользователя не только специальных навыков (формирование геометрической модели объекта и пр.), но и наличия доступа к суперкомпьютерным ресурсам.

В данной статье обсуждается решение, позволяющее объединить существующие и перспективные реализации численных моделей динамики морских объектов как вычислительные сервисы в рамках распределенного высокопроизводительного программного комплекса на основе технологии iPSE. Концепция (и реализующая ее технология) iPSE (Intelligent Problem Solving Environment) определяет принципы построения сред для моделирования сложных систем в форме интеллектуальной оболочки управления параллельными вычислительными процессами в распределенной иерархической среде, включающей в себя вычислительные системы различной архитектуры [3]. Такой подход расширяет достоинства традиционных распределенных проблемно-ориентированных оболочек за счет обеспечения эффективного параллельного исполнения композитных приложений в силу того, что использует для управления параллельными вычислениями симбиотические знания об особенностях предметной области и специфике вычислительного процесса. Концептуальная схема комплекса в целом как интеллектуальной проблемно-ориентированной оболочки (iPSE, Intelligent Problem Solving Environment) приведена на рисунке.

В рамках применяемой концепции комплекс позволяет пользователю формировать варианты композитных приложений для моделирования различных задач динамики корабля. При этом анализ альтернатив и принятие решений по выбору оптимального по производительности варианта осуществляется на основе априорной информации, математических моделей и структурированной базы знаний с использованием различных методов обработки результатов мониторинга вычислительной среды с учетом неполноты и неопределенности данных о ее состоянии.

Рис. Концептуальная схема высокопроизводительного программного комплекса моделирования динамики морских плавучих объектов в экстремальных условиях эксплуатации

Ключевой подсистемой программного комплекса является подсистема сервисов имитационного моделирования внешних воздействий и их влияния на суда и объекты океанотехники, в состав которой входят управляющая оболочка сервиса и вычислительное ядро (по отдельным задачам динамики корабля).

Для имитационного моделирования динамики морских объектов под воздействием экстремальных внешних возмущений требуется предварительно создавать ансамбль реализаций полей ветра, волнения, течений и сопутствующих характеристик в диапазоне мелкомасштабной изменчивости, задавая при этом их синоптические параметры исходя из соответствующей режимной вероятности. Для получения информации о внешних возмущениях в составе комплекса применяются современные стохастические модели, основанные на формализме многомерных динамических систем, параметрически связанных друг с другом в различных диапазонах изменчивости [4]. Эти модели позволяют воспроизводить:

- климатические спектры с заданной повторяемостью (включая спектры, возможные 1 раз в T лет);
- ансамбли пространственно-временных полей морского волнения (с учетом нелинейности профиля волны) на основе спектральных характеристик;
- эффекты локального усиления и обрушения волнения при выходе на мелководную акваторию;
- возникновение волн-убийц (freak waves) в поле обычных волн.

Данные модели (и реализующие их программные сервисы) формируют потоки входных данных для осуществления собственно имитационного моделирования динамики морского объекта, задаваемого в форме сеточной геометрической модели. При этом сервисно-ориентированная архитектура комплекса позволяет предоставлять пользователю доступ к набору различной сложности имитационных моделей динамики судов и объектов океанотехники (спектральные линеаризованные модели, модели в виде уравнений движения с параметрическими связями, двух- и трехмерные гидродинамические модели) [5]. Состыковка моделей различных уровней сложности, включая использование аналитического инструментария исследования упрощенных моделей, позволяет существенно облегчить процесс интерпретации результатов исследования многопараметрических задач динамики судна в экстремальных условиях эксплуа-

тации, что обеспечивает получение более глубоких знаний о процессе взаимодействия морского объекта с внешней средой и особенностях самих моделей. В частности, использование описанных выше имитационных моделей в составе комплекса позволяет:

- изучать существенно нелинейные эффекты, что принципиально важно при моделировании экстремальных явлений;
- детально анализировать всю динамическую картину поведения объекта (явления), в том числе с помощью методов когнитивной научной визуализации;
- анализировать эффекты от влияния различных параметров в ходе модификационного анализа модели в интерактивном режиме.

Для решения двух последних задач необходимо использование технологий динамической трехмерной визуализации. Подсистема визуализации в программном комплексе реализована на основе технологий DirectX и OpenGL, что позволяет эффективно отображать трехмерные динамические сцены с использованием встроенных аппаратных возможностей как графических карт на компьютере пользователя, так и управляющих видеосерверов в полномасштабных системах стереовизуализации.

Технологически программный комплекс спроектирован исходя из специфики применения технологии SciSOA (как адаптации концепции SOA к научным задачам). Это позволяет организовать унифицированный доступ к разнообразным программным модулям, которые могут использовать различные технологии, платформы исполнения, механизмы ввода-вывода и т.п. При этом важнейшей целью является организация эффективной совместной работы расчетных компонентов в процессе решения поставленной перед программным комплексом задачи, что обеспечивает эффективное использование существующих алгоритмических ресурсов в процессе решения новых задач. Унифицированный доступ к данным в ходе функционирования комплекса организован на базе концепции REST. Это позволяет существенно упростить автоматизацию использования отдельных

вычислительных модулей, требующих преобразования входных данных к специфическому формату, а также предоставления данных пользователю в запрашиваемом виде. Кроме того, использование такого подхода предоставляет расширенные возможности анализа данных компьютерных экспериментов и их систематизации.

Работа выполнена при поддержке проектов «Интеллектуальная система навигации и управления морским динамическим объектом в экстремальных условиях эксплуатации», «Интеллектуальные технологии поддержки процессов исследовательского проектирования судов и технических средств освоения океана» и «Высокопроизводительный программный комплекс моделирования динамики корабля в экстремальных условиях эксплуатации» ФЦП «Научные и научно-педагогические кадры инновационной России» на 2007–2013 годы.

Список литературы

1. Интеллектуальные системы в морских исследованиях и технологиях / под ред. Ю.И. Нечаева. – СПб.: ГМТУ, 2001 – 352 с.

2. Нечаев Ю.И., Ярисов В.В. Выбор элементов формы корпуса исходя из требований к остойчивости судна на волнении // Морские интеллектуальные технологии. – 2009. № 2 (4). – С. 45–52.

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

4. Справочные данные по режиму ветра и волнения Баренцева, Охотского и Каспийского морей: справ. изд. / Л.И. Лопатухин [и др.] // Российский морской регистр судоходства. – СПб., 2003. – 214 с.

5. Belenky V.L., Degtyarev A.B., Boukhanovsky A.V. Probabilistic qualities of nonlinear stochastic rolling / // Ocean Engineering. – 1998. – Vol. 25. No. 1. – P. 1–25.

С.М. Белобородов

ОАО «НПО „Искра“», г. Пермь

ПРИКЛАДНАЯ ЗАДАЧА ПАРАЛЛЕЛЬНЫХ РАСЧЕТОВ ЛОКАЛЬНЫХ ДИСБАЛАНСОВ ДИНАМИЧЕСКИ НЕУСТОЙЧИВЫХ РОТОРОВ С ИСПОЛЬЗОВАНИЕМ КЛАСТЕРНЫХ СИСТЕМ

В настоящее время в энергетике постоянно увеличивается количество энергоагрегатов с ротационным рабочим циклом. Использование высокоскоростного вращательного движения в современных машинах высокой удельной мощности требует уравновешенности валопровода, обеспечивающей приемлемую вибрационную нагрузку на опоры и валы роторов, демпфирующие и рабочие элементы.

Сложность и высокая нагруженность рабочих органов машин подразумевает минимизацию «неработающих» нагрузок: деформации валов и элементов роторов, локальных центробежных сил, необходимых усилий демпфирования, а также нагрузок, определяемых неравномерностью изгибной жесткости валов.

Большинство используемых роторов вследствие балансировки на низкооборотном балансировочном оборудовании являются динамически неустойчивыми. Неустойчивость проявляется в резком повышении уровня вибраций как при переходе через «критические» частоты, так и при увеличении вибрационной радиальной нагрузки, обусловленной установкой безопорных элементов валопровода при монтаже.

Ситуация осложнена тем, что в технологическом процессе монтажа валопроводов отсутствуют методики коррекции локальных монтажных дисбалансов роторов, адаптирующих их к условиям эксплуатации. Используемое переносное балансировочное оборудование позволяет выполнять коррекцию только по доступным плоскостям (их в большинстве случаев не более двух).

При этом технологического решения этой усложненной задачи нет: возможности балансировочного оборудования исчерпаны двухплоскостной динамической балансировкой, обес-

печивающей коррекцию моментной неуравновешенности ротора, измеренной относительно балансировочных поверхностей.

Таким образом, складывается достаточно сложное научно-техническое противоречие между необходимостью производства динамически устойчивых валопроводов, имеющейся возможностью этого на основе применения адаптирующих методов балансировки и сборки, и отсутствием технологического программного обеспечения, а также невозможностью их производства без существенного увеличения стоимости.

Одним из направлений решения этого противоречия может стать технологическое программное обеспечение, позволяющее осуществлять информационное сопровождение процесса производства роторов и монтажа валопроводов. Такое сопровождение позволяет разрабатывать меры по коррекции монтажных дисбалансов еще на стадии производства роторов.

Единственным осложнением процедуры расчетов является необходимость ведения параллельных вычислений. Необходимость таких вычислений продиктована взаимным влиянием сборочно-балансировочных операций, оцененных по показателю изменения локальных дисбалансов. При этом одновременный учет изменяющихся и взаимовлияющих параметров существенно удлинял процесс расчетов, а зачастую делал его неприменимым в технологическом процессе. Упрощение расчетов, ограничение факторов приводило к снижению их точности и увеличению неуправляемых погрешностей.

Вместе с тем накопленный опыт при выполнении ряда специфических сборочных и балансировочных операций позволяет не только сформировать ясное представление об этом взаимовлиянии, но и описать алгоритм расчетов, позволяющий обеспечить информационное сопровождение процесса производства.

В основе алгоритма лежит блочная параллельно-последовательная схема очередности решения задач. Основные исходные данные для работы собираются в ходе анализа результатов измерений величин радиальных биений исследуемых поверхностей и расчета величин и направления эксцентриситетов балансировочных и рабочих осей роторов.

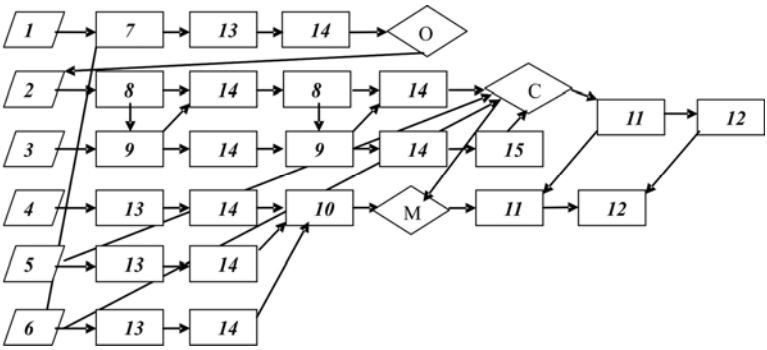


Рис. Алгоритм перспективного программного технологического обеспечения. Обозначения: *1* – ИД для базирования и обработки; *2* – ИД для сборки;*3* – ИД для балансировки; *4* – ИД для монтажа; *5* – ИД для учета динамических изгибов; *6* – ИД для учета неравномерности изгибной жесткости; *7* – процесс обработки; *8* – процесс сборки; *9* – процесс балансировки; *10* – процесс монтажа; *11* – описание локальных дисбалансов; *12* – коррекция монтажных дисбалансов; *13* – процесс измерений; *14* – процесс расчета; *15* – имитация дисбалансов

Основными процессами, оказывающими существенное влияние на изменения локальных дисбалансов, определены: базирование и обработка поверхностей, балансировка и сборка роторов и его элементов, монтаж валопровода, динамический изгиб валов роторов, деформации сложных элементов роторов и демпфирующих элементов. Алгоритм перспективного программного технологического обеспечения приведен на рисунке.

Таким образом, используя возможности высокопроизводительных параллельных вычислений на кластерных системах, предлагаемый алгоритм обеспечивает информационное сопровождение технологических процессов изготовления, балансировки, сборки роторов и монтажа валопроводов, позволяющее вычислительными методами определять меры коррекции монтажных дисбалансов на наиболее целесообразных стадиях производства.

С.А. Бибиков, А.В. Никоноров, В.А. Фурсов, П.Ю. Якимов

Самарский государственный аэрокосмический университет
им. С.П. Королева

РЕКУРРЕНТНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ В РАСПРЕДЕЛЕННОЙ МАССИВНО-МНОГОПОТОЧНОЙ CUDA-СРЕДЕ[1]

Постановка задачи. В последние годы наряду с совершенствованием и ростом производительности мощных суперкомпьютеров наметилась устойчивая тенденция развития технологий параллельных вычислений на персональных компьютерах. Эта тенденция обусловлена существенным технологическим прорывом по крайней мере в трех направлениях.

Во-первых, это появление настольных CPU с возможностью параллельного выполнения программ. В настоящее время для персональных компьютеров доступны процессоры *i*7 с параллельным исполнением 8 потоков. Вторым технологическим прорывом является представленное в 2007 году компанией Nvidia альтернативное направление развития массивно-параллельных вычислений для настольных компьютеров с использованием GPU. Третий фактор, позволяющий всерьез говорить о настольных суперкомпьютерных системах, – это повышение скорости коммуникаций по системной шине для платформы x86 с развитием стандарта PCI Express. Скорость коммуникаций между узлами всегда была узким местом кластерных систем, снижающим их эффективность. Принятие подавляющим большинством производителей стандарта *Infiniband* позволяет строить весьма недорогие вычислительные системы со скоростью обмена данными между узлами до 5 GB/s.

Совокупность указанных факторов стала мотивом возрастающего интереса к созданию компактных, недорогих и вместе с тем высокопроизводительных систем обработки и анализа

[1] Работа выполнена при финансовой поддержке РФФИ (проект № 09-07-00269-а) и программы Президиума РАН.

изображений нового поколения для систем видеоконтроля, видеонаблюдения, подготовки цифровых изображений к печати и др. В работах [1–3] рассматривались примеры создания компактных распределенных систем обработки изображений. В частности, в работе [1] описана реализация алгоритма коррекции цветных цифровых изображений в локальной сети компьютеров посредством специально разработанной GRID-системы, реализованной на Java с native-модулями. В этой работе показана возможность существенного ускорения одного из наиболее затратных по времени процессов подготовки цветных изображений к печати. В работах [4–5] приведены примеры дальнейшего ускорения предпечатной подготовки изображений за счет реализации алгоритмов обработки изображений в CUDA-среде.

Наиболее популярным и широко используемым во многих технологиях обработки изображений является метод скользящего окна. Известны быстрые рекуррентные (например, параллельно-рекурсивные) алгоритмы обработки изображений скользящим окном, реализованные на традиционных аппаратных средствах [6]. Создание таких же эффективных процедур на основе CUDA-технологий, реализуемых с использованием графических ускорителей NVIDIA, к сожалению, пока еще остается проблемной задачей. Связано это с тем, что вычислительная среда CUDA имеет более сложную с точки зрения программирования структуру по сравнению с многопоточной CPU-средой или параллельной средой MPI. Основные структурные особенности среды CUDA следующие.

Во-первых, CUDA-среда имеет двухуровневую многомерную топологию исполнения потоков, которая может описывать десятки тысяч потоков. По аналогии с термином *массивно-параллельная* среда такую среду можно назвать *массивно-многопоточной*.

Во-вторых, потоки могут обмениваться данными только посредством сравнительно небольшого объема общей памяти. Такой обмен возможен в пределах небольшой области локальной связности – блока, включающего (для выпускаемых в настоящее время GPU) не более 512 потоков. Кроме того, когерентность данных в общей памяти достигается только путем

приостановки всех потоков, т.е. упорядоченный обмен данными невозможен. Указанные ограничения создают серьезные проблемы при реализации в среде CUDA рекурсивных алгоритмов (сжатия данных, фильтрации и др.) [7].

Наконец, еще одной важной особенностью является то, что в CUDA-среде можно работать с несколькими видами памяти, имеющими разную область видимости и скорость доступа. Крайне важным при доступе к глобальной памяти является согласование (coalescing) адреса памяти с номером активного потока. Если условие согласования выполняется, то GPU может объединить несколько запросов в одну транзакцию [14]. Несоблюдение этого условия, может привести к падению производительности почти в 8 раз [8, 9].

Многие базовые алгоритмы обработки данных требуют существенной переработки с учетом указанных особенностей. Например, в работе [10] рассмотрен адаптированный для CUDA-среды алгоритм префиксного суммирования. В настоящей статье, являющейся развитием работ [1–5], из всего комплекса рассмотренных в них задач выделены наиболее характерные и сложные для реализации на GPU-процессорах задачи: обработка изображений скользящим окном и организация передачи видеоданных в реальном времени. В частности, рассмотрены основные особенности построения указанных алгоритмов и приводится краткое описание созданного в массивно-многопоточной CUDA-среде программного комплекса.

Рекуррентная CUDA-реализация обработки изображений скользящим окном. Продемонстрируем особенности реализации рекуррентных вычислительных алгоритмов в CUDA-среде на примере простейшего локального оконного фильтра, выполняющего расчет среднего значения. Простота этого алгоритма позволяет наглядно продемонстрировать все основные особенности среды CUDA. Вместе с тем иллюстрируемый на этом простом примере подход может быть легко применен к более сложным локальным оконным фильтрам, применяемым в различных технологиях обработки изображений.

Пусть алгоритм обработки скользящим окном состоит в вычислении для каждой точки изображения суммы значений отсчетов в некоторой окрестности (окне):

$$s_{x,y} = \sum_{i,j=1}^{m} L_{x+i,y+j}, \quad 0 < x, y \le N, \tag{1}$$

где N – размер изображения, m – размер окна (для простоты полагаем, что изображение и окно квадратные). При этом потребуется

$$O_1 = N^2 m^2 \tag{2}$$

арифметических операций.

При рекуррентной реализации этих вычислений в CUDA-среде существенное значение имеет согласование адреса памяти с номером активного потока. Будем считать обращение потока к глобальной памяти согласованным, если оно происходит к словам памяти размером 32 бита с адресами

$$I(t,k) = 32n \cdot k + t, \quad k = 0,1,2..., \quad t = 0,1,2,..., \tag{3}$$

где $I(k,t)$ – адрес ячейки глобальной памяти, к которой производится обращение, n – натуральное число, характеризующее особенности конкретной задачи, связанные с количеством потоков в блоке; t – номер потока внутри блока, k – номер шага некоторой итерации алгоритма, одинаковый для всех потоков внутри блока. Приведенные в спецификации CUDA [8] требования согласованности несколько более сложные и к тому же различаются для различных реализаций стандарта CUDA. Тем не менее легко проверить, что выполнение требования (3) для всех потоков внутри блока позволяют получить согласованный доступ к памяти в смысле спецификации [8].

Рекуррентный алгоритм, удовлетворяющий условию согласованности (3), может быть построен, например, как двухпроходный. При этом сначала выполняется вычисление частичных сумм по столбцам:

$$s^1_{x,y} = s^1_{x,y-1} - L_{x,y-m} + L_{x,y}, \quad s_{x,1} = \sum_{j=1}^{m} L_{x,y+j}, \tag{4}$$

а окончательная сумма получается суммированием по строкам:

$$s_{x,y} = s_{x-1,y} - s^1_{x-m,y} + s^1_{x,y}, \quad s_{1,y} = \sum_{j=1}^{m} s^1_{x+j,y}. \qquad (5)$$

Вычисления по такой схеме требуют

$$O_1 = (2N - m)^2 \qquad (6)$$

арифметических операций.

Ниже приводятся результаты сравнения эффективности предложенных реализаций. Эксперименты проводились на тестовом изображении размером 1024×1024 для окон различных размеров. В первом случае декомпозиция проводилась на 2048 блоков по 512 потоков, часть которых запускалась не одновременно. Во втором и третьем случае запускалось одинаковое количество потоков – 1024, при этом они группировались в 32 блока по 32 потока в каждом. Тестирование проводилось для CPU PIV 3Ghz и GPU Nvidia GF 9500. Результаты представлены в таблице.

Время выполнения различных алгоритмов для разных размеров окна (*m*, мс)

№ п/п		*m* = 5	*m* = 9	*m* = 15
1	CPU, не рекуррентный	283	922	2 531
2	CPU, рекуррентный	142	297	733
3	GPU, не рекуррентный	78	262	473
4	GPU, рекуррентный	23	25	27
5	GPU, рекуррентный с оптимизацией обращений к памяти	12	12	12

Как видно из таблицы, результаты для рекуррентной CUDA-реализации практически не зависят от *m*, а зависят только от стратегии использования памяти.

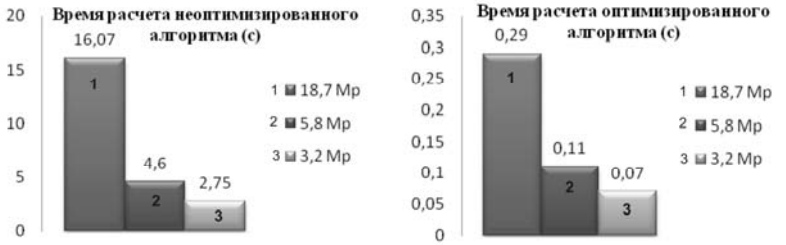
Реализация программного комплекса и результаты экспериментов. Описанная выше схема взаимодействия пото-

ков с памятью при реализации рекуррентного алгоритма использовалась при создании последних версий программного комплекса локализации и коррекции бликов на цифровых изображениях репродукций произведений живописи. Первые версии этого программного комплекса описаны в работах [2–5]. Программный комплекс имеет широкий набор функций для работы с цифровыми изображениями. Приложение работает в режиме *stand alone*, т.е. не требует установки дополнительного программного обеспечения. Благодаря использованию кроссплатформенных библиотек wxWidgets комплекс работает как на платформе Windows, так и на Unix-подобных операционных системах.

Рис. 1. Вид GUI

На рис. 1 представлен вид GUI, разработанного приложения в режиме автоматизированного поиска точечных артефактов (красные точки указывают найденные блики). В последней версии параметры поиска могут задаваться как вручную, так и автоматически путем перехода в режим адаптивного определения параметров. Кроме того, имеется возможность выполнять масштабирование изображения, выбор области поиска, а также модифицировать маску с отмеченными артефактами, сформированную алгоритмом, вручную.

В работах [4] и [5] авторами была показана возможность существенного ускорения алгоритмов обработки изображений в программной системе локализации и устранения бликов за счет использования универсальных графических процессоров NVIDIA и технологии CUDA. В частности, на графических ускорителях NVIDIA GeForce 8400 и 9500 с 8 мультипроцессорами, было продемонстрировано более чем десятикратное ускорение CUDA-реализации по сравнению с реализацией на CPU. В настоящей работе мы демонстрируем возможность дальнейшего повышения ускорения за счет использования в программной системе оптимизированных рекуррентных алгоритмов.



а *б*

Рис. 2. Время работы CUDA-алгоритма: *а* – алгоритм, опубликованный в работе [4]; *б* – результаты оптимизированного рекуррентного алгоритма

Для экспериментальной проверки скорости использовались три изображения размером 3,2, 5,8 и 18,7 мегапикселей. Для расчетов использовалась видеокарта Nvidia GeForce 9500. На рис. 2, *а* приведено время вычислений для алгоритма из работы [4], на рис. 5, *б* приведено время вычислений для алгоритма поиска бликов на изображении, использующего оптимизированную рекуррентную процедуру обработки скользящим окном. Из приведенных диаграмм видно, что в среднем, по трем экспериментам, достигнуто ускорение более чем в 45 раз.

Подчеркнем, что полученное в эксперименте ускорение получено исключительно за счет оптимизации CUDA-алгоритма.

Достигнутые показатели скорости обработки с использованием оптимизированной версии алгоритма поиска бликов открывают возможности для их более широкого применения. В частности, наряду с использованием этого алгоритма в задачах предпечатной обработки изображений, как это было сделано в [4], по-видимому, удастся использовать эту технологию также для обработки видео данных в реальном времени.

Список литературы

1. Никоноров А.В., Фурсов В.А. Предоставление сервиса управления цветовоспроизведением цветных изображений в сети интернет // Телематика–2007: тр. XIV Всерос. науч.-метод. конф. – С.-Петербург, 18–21 июня, 2007. – СПб.,2007. – С. 412–413.

2. Никоноров А.В., Фурсов В.А. Распределенная вычислительная среда коррекции цветных изображений // Телематика–2008: тр. XV Всерос. науч.-метод. конф. – С.-Петербург, 23–26 июня, 2008. – СПб., 2008. – С. 88–89.

3. Фурсов В.А., Никоноров А.В. Распределенный алгоритм коррекции точечных артефактов на цветных изображениях // Параллельные вычисления и задачи управления (PACO–2008): тр. VI Междунар. конф. Москва, 27–29 октября, 2008. – М.,2008.

4. Исследование эффективности технологии CUDA в задаче распределенной предпечатной подготовки цифровых изображений / С.А. Бибиков [и др.] // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: тр. всерос. суперкомпьютерной конф. 21–26 сентября 2009 г., г. Новороссийск. – М.: Изд-во МГУ, 2009. – С. 204–207.

5. CUDA-технология цветовой коррекции теневых искажений на цифровых фотокопиях произведений живописи / С.А. Бибиков [и др.] // Параллельные вычислительные технологии–2010: сб. тр. междунар. науч. конф. Уфим. гос. авиац. техн. ун-та (УГАТУ), г.Уфа, 29 марта – 2 апреля 2010 г. – Уфа,2010. – 656 с.

6. Методы компьютерной обработки изображений / Сойфер В.А. [и др.]. – М.: Физматлит, 2003. – 784 с.

7. Nickolls J., Buck I., Skadron K., Scalable Parallel Programming with CUDA // ACM Queue, Vol. 6. No. 2. March/April 2008. – P. 40–53. – URL: http://mags.acm.org/queue/20080304/?u1=texterity.

8. NVIDIA CUDA Programming Guide, 2010. 130 p. – URL: http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf.

9. NVIDIA CUDA Best Practices Guide, 2009. – 68 p.

10. Harris M., Parallel Prefix Sum (Scan) with CUDA, 2008. – 21p. – URL: http://developer.download.nvidia.com/compute/cuda/sdk/website/projects/scan/doc/scan.pdf.

11. Мурызин С.А., Сергеев В.В., Фролова Л.Г. Исследование эффективности двумерных параллельно-рекурсивных КИХ-фильтров // Компьютерная оптика. – М.: МЦНТИ, 1992. – Вып. 12. – С. 65–71.

12. Smart, J. Cross-platform GUI Programming with wxWidgets. – Prentice Hall, 2000. – 714 p.

13. Munshi A. Kronos OpenCL Working Group, The OpenCL Specification. – 2009. – 314 p.

14. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.

М.Г. Бояршинов, Д.С. Балабанов

Пермский государственный технический университет

ВЫЧИСЛИТЕЛЬНОЕ МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ СЖИМАЕМОЙ СРЕДЫ С ИСПОЛЬЗОВАНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА

Методы вычислительного моделирования движения газовых потоков с использованием высокопроизводительных вычислительных комплексов в настоящее время являются основным инструментом решения краевых задач механики сплошных сред, включающих системы дифференциальных уравнений

движения, неразрывности, энергии, теплопроводности, концентрации, состояния и прочих с соответствующими краевыми условиями. Для построения решений подобных задач большее внимание уделяется конечно-разностным методам, требующим относительно малых вычислительных ресурсов. Значительный интерес вызывают схемы факторизации эволюционных дифференциальных уравнений, позволяющие расщеплять многомерные задачи на последовательности одномерных задач, что приводит к существенному повышению эффективности вычислительных алгоритмов. В ряде исследований для решения задач движения сплошной среды используются методы конечных и граничных элементов с различными видами аппроксимации полей скорости, перемещения, плотности, давления, температуры и других характеристик.

Следует отметить, что вопросам сеточной аппроксимации подобластей, моделирующих источники количества движения, энергии и массы (отверстия, щели, локализованные источники, в том числе со сложной геометрией) при построении численных решений, уделяется недостаточно внимания.

В настоящей работе рассматриваются особенности движения сжимаемой среды из точечного источника, генерирующего поток массы с интенсивностью m и мощностью ε. Система уравнений Эйлера, описывающая движение сплошной среды в трехмерном пространстве, включает уравнения неразрывности

$$\frac{\partial \rho}{\partial t} + \nabla (\rho \mathbf{V}) = m\delta(\mathbf{0});$$

движения $\mathbf{V}$

$$\frac{\partial (\rho \mathbf{V})}{\partial t} + \nabla (\rho \mathbf{V}\mathbf{V}) + \nabla P = \mathbf{0};$$

энергии

$$\frac{\partial (\rho U)}{\partial t} + \nabla (\rho U \mathbf{V}) + \nabla (\mathrm{P}\mathbf{V}) = \varepsilon\delta(\mathbf{0});$$

состояния (адиабатический процесс)

$$\rho(k-\mathbf{1})(U - \mathbf{V}\cdot\mathbf{V}/2) - P = 0.$$

В приведенных уравнениях обозначено: ρ – плотность среды; **V** – вектор скорости потока; P – давление; U – полная удельная энергия; k – показатель адиабаты; δ – дельта-функция Дирака. В силу сферической симметрии движения среды рассматривается 1/8 часть конечной области G (рис. 1); предполагается, что источник находится в одной из вершин рассматриваемого куба. В начальный момент времени в области заданы распределения компонент вектора скорости, плотности и энергии. Для всех искомых функций приняты «мягкие» граничные условия, моделирующие установление течения сплошной среды на «удаленных» от источника границах расчетной области. Это, в известной степени, соответствует традиционным постановкам краевых задач механики жидкостей и газов, в которых заранее не известен характер потоков жидкостей и газов вдали от источников.

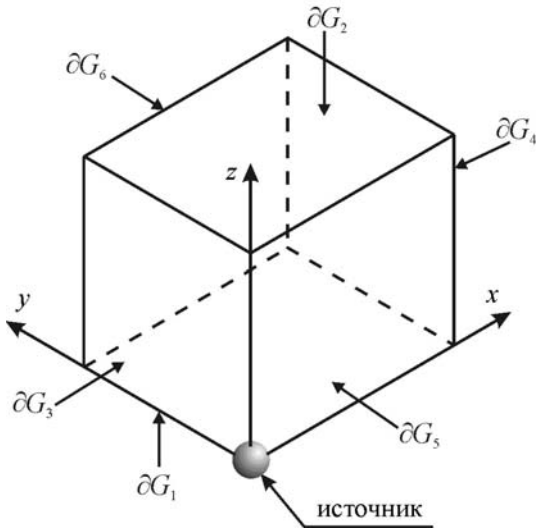


Рис. 1. Схема расчетной области и обозначения ее границ

Для построения численного решения поставленной задачи применяется метод крупных частиц (метод Давыдова), который хорошо зарекомендовал себя при моделировании вихревых

структур, отрывных явлений, фильтрационных и струйных течений, газо- и гидродинамических потоков с большими перемещениями и соударяющимися поверхностями раздела, движения многокомпонентных сред, течения сквозь проницаемые объекты и многих других процессов. С использованием метода крупных частиц на основе технологии ОМР разработаны комплексы программ для численного исследования пространственного движения сжимаемой среды. Для реализации вычислительного эксперимента использовался высокопроизводительный вычислительный комплекс Пермского государственного технического университета (ВВК ПГТУ, 512 ядер).

Верификация результатов вычислительного моделирования выполняется с помощью точного решения стационарной ($\partial\rho/\partial t = 0$, $\partial\rho U/\partial t = 0$, $\partial\rho\mathbf{V}/\partial t = \mathbf{0}$) системы уравнений

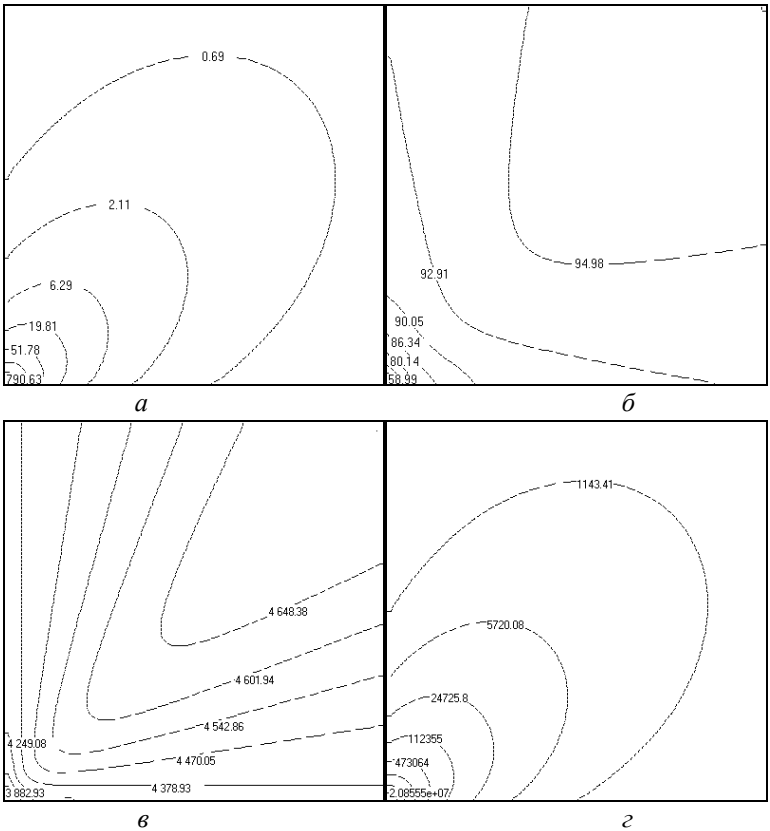
$$\begin{cases} \nabla(\rho\mathbf{V}) = m\,\delta(\mathbf{0}), \\ \nabla(\rho\mathbf{V}\mathbf{V}) + \nabla P = 0, \\ \nabla(\rho U\,\mathbf{V}) + \nabla\cdot(P\,\mathbf{V}) = \varepsilon\delta(\mathbf{0}), \\ \rho(k-1)(\mathrm{U} - \mathbf{V}\cdot\mathbf{V}/2) - P = 0, \end{cases}$$

которое имеет вид

$$V = \sqrt{2\varepsilon/m}; \qquad \rho = \sqrt{m^3}\Big/4\pi r^2\sqrt{2\varepsilon}; \qquad U = \varepsilon/m; \qquad P = 0.$$

Анализ результатов вычислительного моделирования газодинамических характеристик движущейся среды при наличии точечного источника показывает, что распределение значений полной удельной энергии U мало зависит от параметров разностных сеток (в частности, от размеров расчетных ячеек), в отличие от распределений значений скорости $\mathbf{V}$, плотности ρ и давления P, которые весьма существенно зависят от степени дискретизации расчетной области. Следует отметить наличие зависимостей найденных функций плотности скорости, полной удельной энергии и давления движущейся среды как от расстояния до источника, так и от направления практически при всех используемых вариантах дискретизации расчетной области, что противоречит точному решению (рис. 2). Кроме того, распределение плотности движущейся среды в различных сечениях расчетной области ука-

зывает на наличие затопленной струи, генерируемой источником, что также противоречит точному решению рассматриваемой задачи. Анализ результатов вычислительного моделирования позволяет предположить, что отсутствие сферической симметрии в распределении характеристик потока определяется используемыми при проведении расчетов разрешающими соотношениями метода крупных частиц, полученными в предположении о кубической форме расчетных ячеек, аппроксимирующих расчетную область.



а *б*

в *г*

Рис. 2. Распределения плотности (*а*), кг/м^3; модуля скорости (*б*), м/с; полной удельной энергии (*в*), Дж/кг; и давления (*г*), Па, в плоскости $z = 0$ при аппроксимации исследуемой области с использованием 10^6 расчетных ячеек

Следует отметить возрастание погрешностей численных решений (используется чебышёвская норма отклонения численного решения от точного) при повышении степени дискретизации расчетной области. В то же время наибольшие погрешности сосредоточены вблизи точечного источника и существенно уменьшаются по направлению к периферии расчетной области (рис. 3).

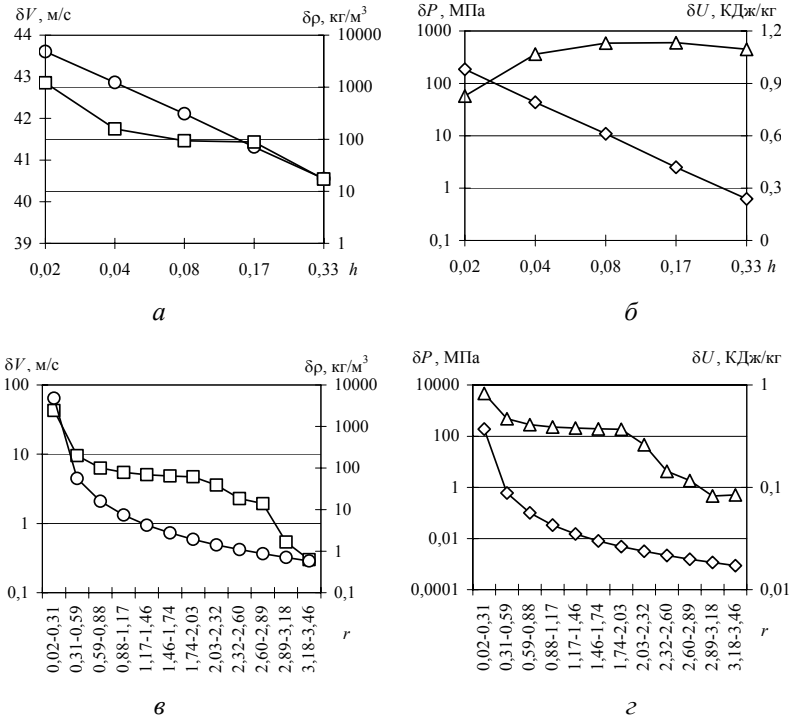


Рис. 3. Погрешности решения в зависимости от аппроксимации расчетной области (*а*, *б*; *h* – размер ячейки, мм) и расстояния до источника (*в*, *г*; *r*, мм): плотность (–О–), скорость (–□–), полная удельная энергия (–Δ–) и давление (–◊–)

Для уточнения численного решения поставленной задачи (в частности, для повышения степени симметрии в распределении численных решений) проведен ряд вычисли-

тельных экспериментов с применением аппроксимации зоны, моделирующей точечный источник, набором расчетных ячеек (рис. 4).

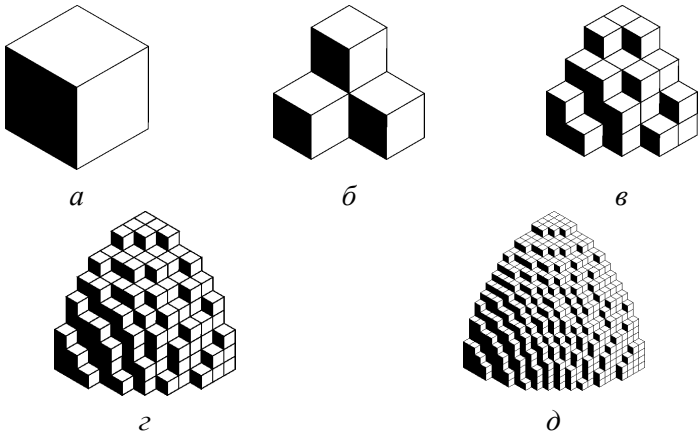
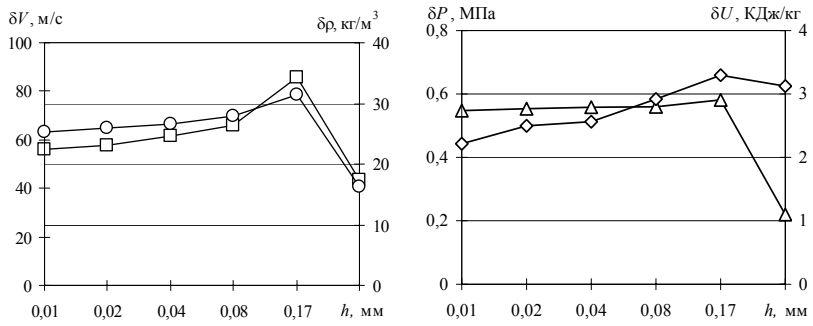


Рис. 4. Аппроксимация 1/8 части области, занятой точечным источником, с использованием 1 (*а*), 4 (*б*), 35 (*в*), 272 (*г*) и 2157 (*д*) расчетных ячеек

В отличие от зависимостей, полученных при аппроксимации источника одной ячейкой, при использовании аппроксимации зоны источника и повышении степени дискретизации расчетной области отмечается снижение погрешностей для всех искомых функций (рис. 5).

Получаемые при этом поля искомых функций близки к сферически симметричным. Распределение плотности в этом случае показывает отсутствие затопленной струи, отмеченное ранее. В то же время следует отметить, что найденные распределения искомых функций имеют более низкие значения по сравнению с точными решениями. Это, по-видимому, объясняется тем, что характеристики точечного источника m и ε распределяются теперь по объему всех ячеек, аппроксимирующих этот источник, снижая тем самым интенсивность его воздействия на искомые решения.



а *б*

Рис. 5. Погрешности решения в зависимости от аппроксимации расчетной области (h – размер ячейки) при использовании аппроксимации точечного источника: плотность (–О–), скорость (–□–), полная удельная энергия (–Δ–) и давление (–◊–)

Комплекс программ, реализующий пространственный вариант метода крупных частиц (метода Давыдова), позволил выполнить сравнение точного решения задачи о движении сжимаемой среды, генерируемой точечным источником, и численного решения, полученного этим методом, с использованием последовательностей разностных сеток с уменьшающимися размерами расчетных ячеек. Погрешности численного решения достигают наибольшего значения вблизи источника и существенно уменьшаются в направлении к периферии расчетной области. Дополнительная аппроксимация источника конечным числом ячеек приводит к распределению искомых величин, близкому к сферической симметрии, и, одновременно, к понижению значений искомых функций по сравнению с точными значениями.

Д.Б. Бражкин

Нижегородский государственный университет им. Н.И. Лобачевского

РЕШЕНИЕ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ

В настоящее время широкое распространение получили вычисления общего назначения на графических картах (General-Purpose Computing on Graphics Processing Units, GPGPU). Статистика показывает, что производительность современных видеокарт растет значительно более высокими темпами, чем производительность центральных процессоров.

NVidia CUDA (Compute Unified Device Architecture) [1] представляет собой программно-аппаратное решение, предназначенное для осуществления вычислений на GPU как на обычном вычислительном устройстве, без необходимости использования графических API. CUDA фактически представляет собой расширение языка *C*. Каждая новая версия технологии позволяет использовать более широкий спектр возможностей: снимаются некоторые ограничения, повышается общая производительность, появляются новые функции.

Высокая пиковая мощность графических карт (в сравнении с CPU) позволяет использовать их при решении многих практических задач, в том числе таких трудоемких, как поиск глобального оптимума многомерных функций.

Постановка задачи. Рассматривается конечномерная задача глобальной оптимизации (задача нелинейного программирования) вида:

$$f(y) \to \min, y \in Y,$$

$$Y = \left\{ y \in D : g_j(y) \le 0, 1 \le j \le m \right\},$$

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

т.е. задача отыскания экстремальных значений целевой (минимизируемой) функции на множестве, задаваемой координатными и функциональными ограничениями на выбор допустимых точек (векторов) $y = (y_1, y_2, ...)$. В данной модели величины a_i, b_i, $1 \le i \le N$ (границы измерения варьируемых параметров задачи – координаты вектора y) являются заранее заданными константами.

Предполагается, что целевая функция $f(y)$, а также левые части ограничений $g_j(y)$ удовлетворяют условию Липшица с соответствующими (возможно, неизвестными) константами: L, $L_j (1 \le j)$, т.е.

$$|f(y_1) - f(y_2)| \le L \|y_1 - y_2\|, y_1, y_2 \in D,$$

$$|g_j(y_1) - g_j(y_2)| \le L_j \|y_1 - y_2\|, 1 \le j \le m, y_1, y_2 \in D.$$

Применение развёрток в задачах с ограничениями. Рассмотрим кривую Пеано $y(x), x \in [0,1]$, отображающую однозначно и непрерывно отрезок вещественной оси на многомерный гиперпараллелепипед,

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

т.е.

$$D = \{y(x) : 0 \le x \le 1\}.$$

Непрерывность целевой функции $f(x)$ исходной многомерной задачи обеспечивает соотношение $\min_{y \in D} f(x) = \min_{x \in [0,1]} f(y(x))$, и возможность вместо многомерной задачи решать одномерную задачу $f(y(x)) \to \min, x \in [0,1]$.

Свойства различных вариантов разверток, а также алгоритмы их построения подробно рассмотрены в [3, 5]. В данной работе реализована кусочно-линейная пеаноподобная развертка. Достоинством данной схемы является ее простота, однако развертка такого типа обладает некоторыми недостатками, для их устранения возможно применение других схем – неинъективной или множественной развертки.

Для решения полученной одномерной задачи используется индексный метод. Подробное описание вычислительной схемы метода рассмотрено в [3].

Программная реализация. По сравнению с предыдущей версией приложения, рассматриваемой в [2], структура претерпела значительные изменения. Теперь графический процессор полностью выполняет всю математическую составляющую алгоритма, а центральный процессор лишь инициализирует данные и отвечает за интерфейс приложения. Общая схема приложения представлена на рис. 1.

Параллельная схема реализации. Распараллеливание алгоритма связано со спецификой устройства графического процессора и представления его как вычислительного устройства при использовании технологии NVidia CUDA. При организации вычислений GPU представляется в виде набора нескольких мультипроцессоров, каждый из которых обладает собственной «быстрой» памятью и способен обрабатывать данные в несколько потоков [1].

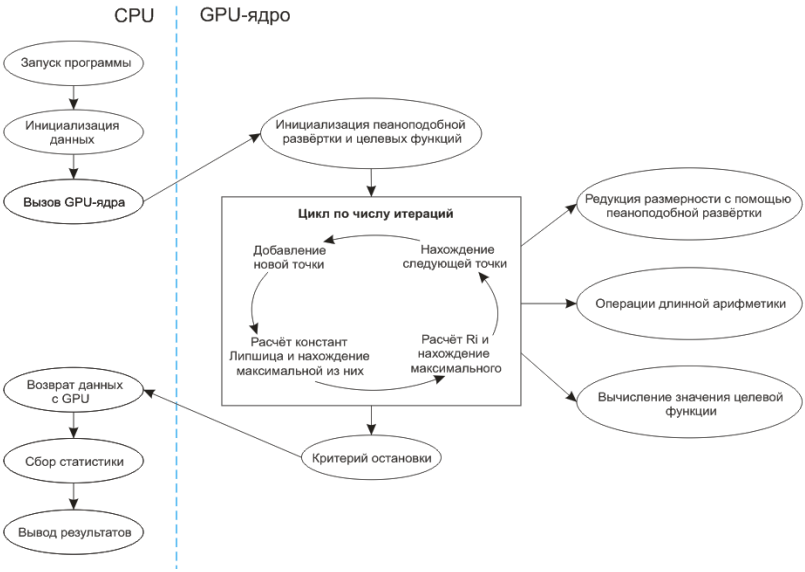


Рис. 1. Общая схема приложения

Для распараллеливания любого алгоритма на GPU удобнее всего разделить обрабатываемые данные между потоками таким образом, чтобы взаимодействие между ними сводилось к минимуму. Это связано с архитектурой GPU, а именно с низкой скоростью обмена данными между вычислительными блоками. В данном случае целесообразно разделить между потоками массив точек предшествующих испытаний. Наиболее трудоёмкими шагами алгоритма являются нахождение констант Липшица, а также вычисление характеристик интервалов. Распараллеливание на этих этапах реализовано схожим образом. Рассмотрим в качестве примера вычисление характеристик интервалов.

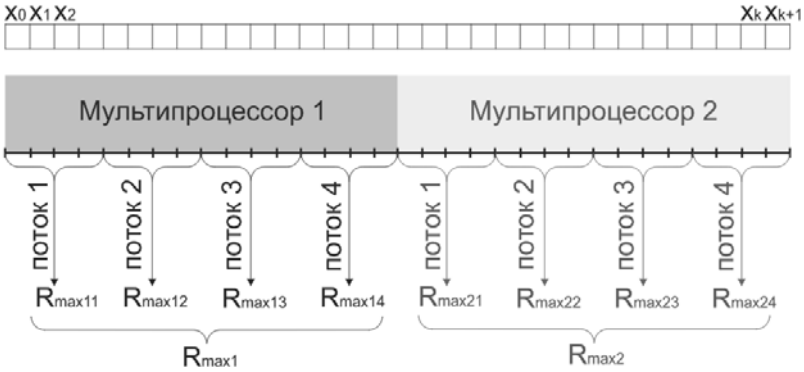


Рис. 2. Схема распараллеливания вычисления характеристик

Примем номер мультипроцессора за *bid*, $0 \le bid < B$, где B – количество мультипроцессоров, а номер потока в нём – за *tid*, $0 \le tid < T$, где T – количество таких потоков. На каждой итерации алгоритма массив точек предшествующих испытаний делится между мультипроцессорами на равные группы. Число точек в массиве равно $k+2$, таким образом, мультипроцессор с номером *bid* получает на вход часть массива с индексами (рис. 2)

$$\left(\frac{bid}{B}(k+1), \frac{bid+1}{B}(k+1) \right).$$

Они, в свою очередь, также поровну разделяются между потоками в мультипроцессоре, на промежутки

$$\left(\frac{bid + \frac{tid}{T}}{B}(k+1), \frac{bid + \frac{tid+1}{T}}{B}(k+1) \right).$$

Каждый поток обрабатывает свою часть отрезка, после чего происходит сбор данных и переход к следующему шагу алгоритма.

Стоит также отметить, что периодически требуется синхронизировать все потоки графического ядра. Технология NVidia CUDA позволяет синхронизировать лишь потоки в пределах одного блока потоков. Однако использование некоторых особенностей технологии позволило реализовать эффективную глобальную синхронизацию потоков.

Использование чисел расширенной точности. При реализации многомерного алгоритма глобальной оптимизации возникает следующая проблема: для достижения точности $\varepsilon = (1/2)^m$ в R^N на отрезке [0, 1] требуется точность $(1/2)^{m \cdot N}$, где m – уровень построения развёртки, а N – размерность пространства. При использовании типа *float* максимально возможная точность ограничена условием $Nm < 23$. Использование типа *double* позволяет расширить этот диапазон до $Nm < 52$, однако в силу особенностей архитектуры GPU значительно снижается производительность. На момент создания приложения никакой информации о сторонних библиотеках для работы с числами расширенной точности для GPU не было. Единственным выходом из сложившейся ситуации являлось создание собственной библиотеки.

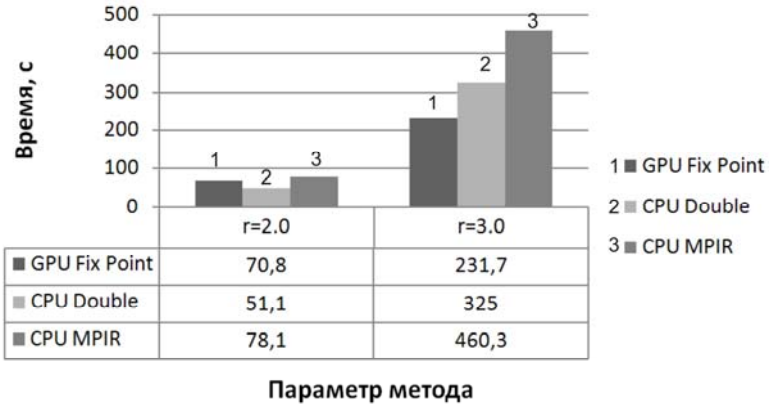
Из операций с числами расширенной точности требуется сложение, вычитание, сравнение, обнуление, битовый сдвиг, а также приведение типов. В качестве базового типа для создания библиотеки использовался 32-битный *int*. Это позволило использовать операции битового сдвига, выполняемые на GPU достаточно быстро [4].

В силу специфики задачи «длинные» числа расположены в интервале (0,1), для их хранения и обработки достаточно реализации чисел с фиксированной точкой, вычисления с которой выполняются быстрее аналогичных вычислений с плавающей точкой.

Результаты вычислительных экспериментов. Для тестирования полученной реализации был использован набор функций Растригина, имеющих вид $f(x) = nA + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i))$; глобальный оптимум этой функции равен 0 при $x_i = 0$.

Эксперименты проводились с тремя реализациями алгоритма – две версии работают на CPU и одна версия на GPU. Тестовая конфигурация: Asus P5Q-E P45, Core 2 Duo E8500@3,8GHz, 2 Gb RAM, NVidia 9800 GT, MS Windows 7, NVidia CUDA 3.0.

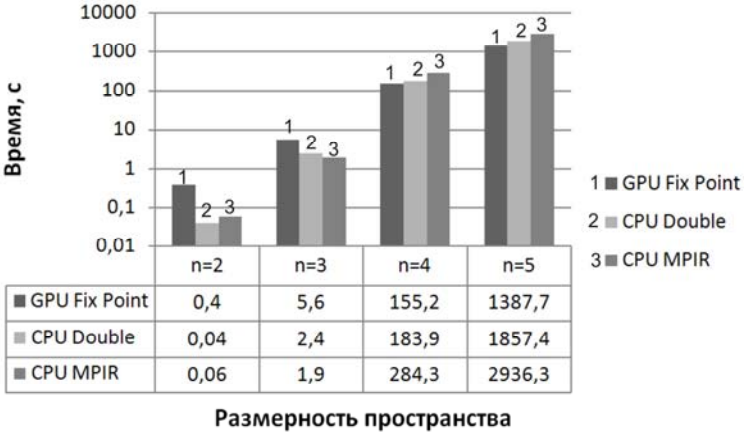
Первый график (рис. 3) показывает время работы различных реализаций алгоритма в зависимости от параметра индексного метода.



	r=2.0	r=3.0
GPU Fix Point	70,8	231,7
CPU Double	51,1	325
CPU MPIR	78,1	460,3

Рис. 3. Время работы реализаций в зависимости от параметра метода

Второй эксперимент (рис. 4) – тестирование алгоритма при различной размерности решаемой задачи. Как видно из первых двух графиков, GPU-версия с ростом сложности задачи показывает лучшие результаты, чем версии на CPU.



	n=2	n=3	n=4	n=5
GPU Fix Point	0,4	5,6	155,2	1387,7
CPU Double	0,04	2,4	183,9	1857,4
CPU MPIR	0,06	1,9	284,3	2936,3

Рис. 4. Время работы реализаций в зависимости от размерности задачи

Третий эксперимент показывает масштабируемость GPU-реализации по числу используемых вычислительных блоков. Вычислительный блок на GPU представляет собой группу из 8 ядер. То есть если видеокарта содержит 112 ядер, то эти ядра сгруппированы в 14 блоков. В связи с этим эксперимент проводился с подключением 1, 2, 4, 8 и всех 14 вычислительных блоков. Результаты представлены на рис. 5. График показывает практически линейный рост производительности. Это объясняется тем, что сложность решаемой задачи высока и, как следствие, основная нагрузка ложится на вычислительные блоки, а не на шину данных.

В данной работе рассмотрена полная GPU-реализация индексного метода глобальной оптимизации. Проведённые эксперименты показали, что на сложных вычислительных задачах графический процессор показывает отличную производительность и его использование в качестве вычислительного устройства себя оправдывает.

Наибольшую трудность вызывает грамотное использование вычислительных возможностей GPU. Всё-таки в первую очередь это устройство для обработки и вывода графической информации, обладающее, однако, огромным вычислительным

потенциалом. С появлением новых средств разработки эта задача значительно упрощается, и если несколько лет назад использование GPU в качестве средства вычисления было лишь исследовательской задачей для энтузиастов, то теперь подобные возможности находят применение даже в коммерческих продуктах.

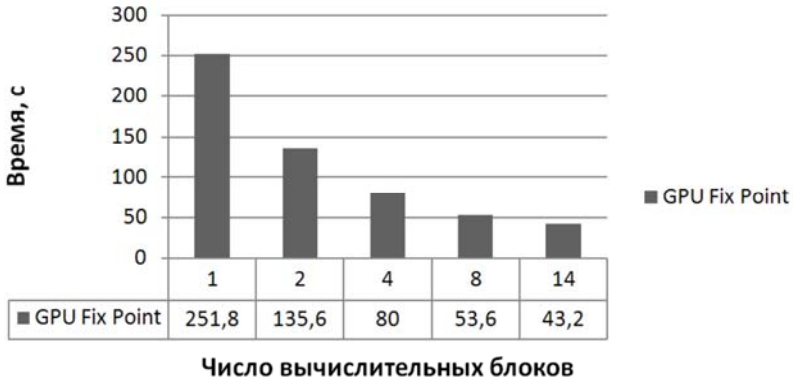


Рис. 5. Масштабируемость GPU-реализации

В перспективе видится несколько возможностей для развития проекта: это и применение других типов развёрток, и дальнейшая оптимизация кода, возможно использование новых методов и технологий.

Список литературы

1. NVidia CUDA C Programming Guide. – URL: http:// developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/ NVIDIA_CUDA_ProgrammingGuide.pdf.

2. Бастраков С.И., Бражкин Д.Б.. Решение задач глобальной оптимизации с использованием смешанных CPU-GPU вычислений // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2009. – С. 23–29.

3. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация: учеб. пособие. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2007.

4. Кнут Д. Искусство программирования. Т. 2. Получисленные алгоритмы. – М.: Вильямс, Addison Wesley Longman, 2000.

5. Стронгин Р.Г. Численные методы многоэкстремальной оптимизации (информационно-статистические алгоритмы). – М.: Наука, 1978.

Л.Н. Бутымова, В.Я. Модорский,
А.Ф. Шмаков, Д.Ф. Гайнутдинова

Пермский государственный технический университет

АНАЛИЗ КОЛЕБАТЕЛЬНЫХ РЕЖИМОВ АКУСТИЧЕСКОГО ТРАНСФОРМАТОРА

В целом ряде машиностроительных конструкций имеются каналы, заполненные жидкостью. Требуется создание методик, позволяющих прогнозировать поведение в динамической системе «жидкость–конструкция» при внешнем колебательном воздействии. Кроме того, авторам неизвестны модели, объясняющие эффект направленного движения жидкости в каналах упругих конструкций при их периодическом изгибе.

В полной постановке требуются моделирование источника колебаний, оптимизация его конструкции, обеспечение эффективности технологического процесса, моделирование самого технического объекта, что предполагает совместное решение гидродинамической задачи и задачи теории упругости (рис. 1).

В Центре высокопроизводительных вычислительных систем имеется необходимое программное обеспечение и оборудование для решения такого класса задач. В частности, Flow Vision, ABAQUS, ANSYS CFD, ANSYS CFX, STAR CD, LS DYNA, ANSYS WorkBench.

В силу сложности поставленной задачи принято решение о ее поэтапной реализации. На первом этапе, рассматриваемом в данной работе, моделируется источник колебаний – акустический трансформатор (АТ).

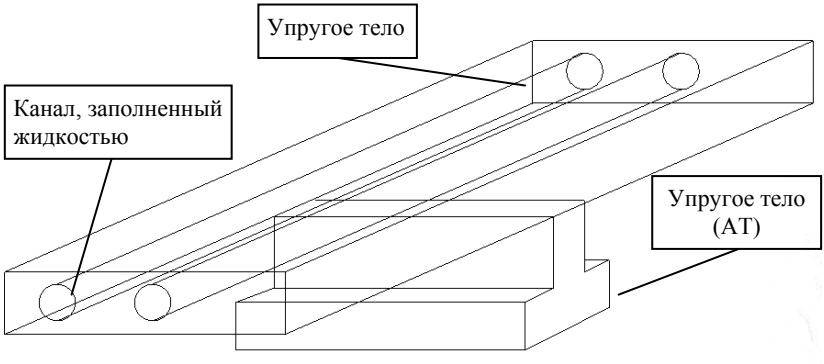


Рис.1. Расчетная схема

Акустические трансформаторы или волноводы применяются для получения ультразвуковых колебаний большой амплитуды. Эти устройства представляют собой стержни переменного сечения из металла, пластика, стекла, керамики и других упругих материалов. В зависимости от характера изменения сечения применяются акустические трансформаторы, ступенчатые, конические и экспоненциальные (рис. 2).

Длина волновода должна иметь вполне определенное значение, чтобы частота собственных колебаний устройства совпадала с частотой возбуждающих колебаний, т.е. чтобы имел место механический резонанс (табл. 1).

Таблица 1

Размеры акустического трансформатора

Габаритный размер	Размерность	Значение
Длина по X	м	0,08
Длина по Y	м	0,05
Длина по Z	м	0,1

Таблица 2

Физико-механические, температурные и электромагнитные характеристики материала конструкции

Характеристика	Размерность	Значение
Модуль Юнга	Па	200000000000
Коэффициент Пуассона	–	0,3
Плотность	кг/м³	7850
Коэффициент теплового расширения	1/°C	0,000012
Предел текучести при растяжении	Па	250000000
Предел текучести при сжатии	Па	250000000
Предел прочности на растяжение	Па	460000000
Предел прочности на сжатие	Па	0
Тепловая проводимость	W/м·°C	60,5
Удельная теплоемкость	Дж/кг·°C	434
Относительная проходимость	–	10000
Удельное сопротивление	Ом·м	0,00000017

Поскольку в волноводах переменного сечения скорость распространения фронта волны является функцией скорости изменения сечения и физико-механических характеристик материала (табл. 2), вычисление резонансной волны волновода представляет определенные трудности. Кроме того, сложно вычислить амплитуды паразитных поперечных колебаний в волноводе, если его ширина или диаметр сравнимы с длиной волны или больше ее. В этом случае рабочая поверхность волновода колеблется несинфазно, что снижает качество генерируемых ультразвуковых колебаний.

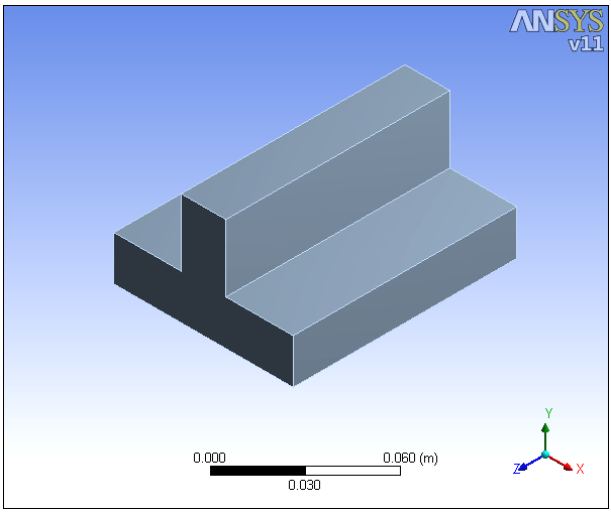


Рис. 2. Общий вид твердотельной модели АТ

Для данной конструкции был проведен модальный анализ. Данный вид расчета используется для определения характеристик вибраций (собственных частот и форм колебаний) конструкции в процессе проектирования. Такой расчет может также являться начальной фазой другого, более подробного расчета, в том числе расчета переходных процессов, исследования вынужденных колебаний или случайных колебаний [Басов К.А. ANSYS: справочник пользователя. – М.: ДМК Пресс, 2005. – 640 с.].

Собственные частоты и формы колебаний являются важными параметрами, учитываемыми при проектировании конструкции в целях учета условий динамического нагружения. Они требуются также в дальнейшем расчете случайных колебаний, расчете вынужденных колебаний при помощи метода наложения форм или в расчетах переходных процессов [Басов К.А. ANSYS: справочник пользователя. – М.: ДМК Пресс, 2005. – 640 с.]. Для анализа собственных частот выбираем первые двадцать мод. Широкую грань закрепляем неподвижно. Материал–конструкционная сталь.

Рассмотрим результаты модального анализа (табл. 3).

Таблица 3

Распределение частот по модам колебаний

Номер моды	Частота, Гц	Номер моды	Частота, Гц	Номер моды	Частота, Гц	Номер моды	Частота, Гц
1	10757	6	28028	11	35786	16	42337
2	12837	7	30335	12	38087	17	44410
3	15778	8	30801	13	38914	18	45250
4	18574	9	31048	14	40987	19	46213
5	26253	10	33385	15	41229	20	46837

Визуально определяем, какой частоте соответствуют колебания в направлении оси *Y* (см. рис. 2).

Переходим к гармоническому анализу. Для этого зададим перемещение широкой грани вдоль оси *Y* с амплитудой 5 мкм. По остальным граням зададим свободное перемещение.

Рассмотрим результаты гармонического анализа. Диапазон частот изменяется от 9,35 до 35 кГц (рис. 4).

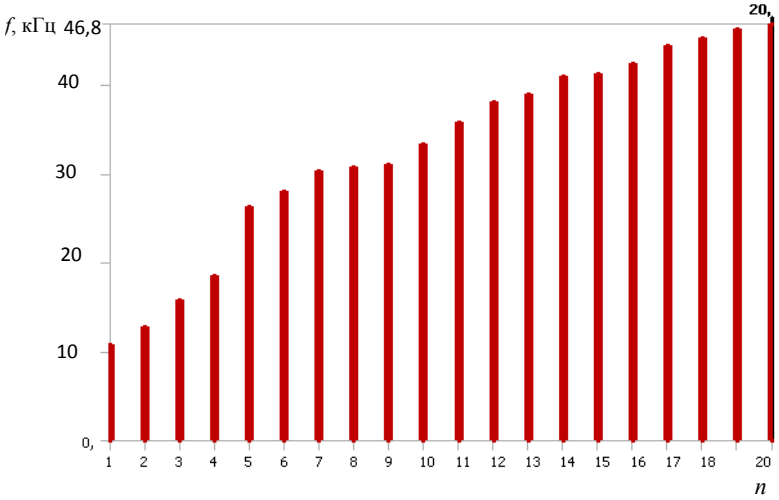


Рис. 3. Зависимость частоты от номера моды колебаний

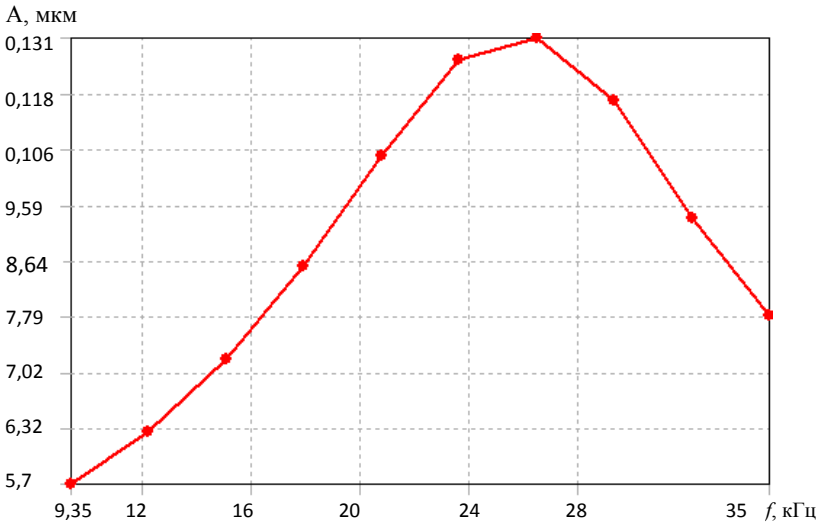


Рис. 4. Зависимость амплитуды перемещения рабочей поверности от частоты

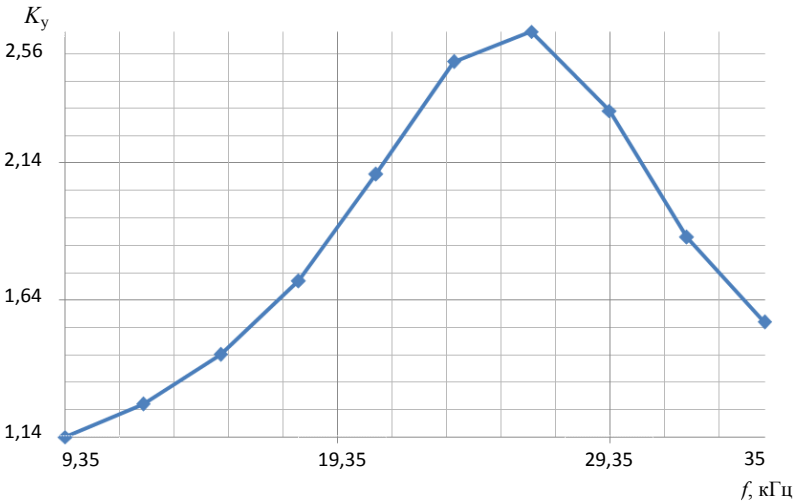


Рис. 5. График зависимости коэффициента усиления от частоты

Модальный анализ показал, что резонансной является частота колебаний вдоль оси *Y*, равная 26,3 кГц. В ходе гармонического анализа был рассчитан коэффициент усиления K_y, равный отношению осредненной амплитуды перемещений узкой грани к отношению осредненной амплитуды перемещений широкой грани. График зависимости коэффициента усиления от частоты приведен на рис. 5.

Расчеты показали, что наибольший коэффициент усиления K_y равен 2,62 при частоте 26,5 кГц.

По результатам проведенных расчетов определены параметры нагружения экспериментальной установки, позволяющие получить максимальный коэффициент усиления K_y для заданных физико-механических и геометрических характеристик акустического трансформатора.

Л.Н. Бутымова, В.Я. Модорский

Пермский государственный технический университет

АНАЛИЗ ДИНАМИЧЕСКОГО НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ АКУСТИЧЕСКОГО ТРАНСФОРМАТОРА

Акустические трансформаторы (АТ), или волноводы применяются для получения высокочастотных колебаний большой амплитуды (рис. 1).

Размеры акустического трансформатора, физико-механические, температурные и электромагнитные характеристики материала конструкции приводятся в статье[1].

[1] Анализ колебательных режимов акустического трансформатора / Л.Н. Бутымова [и др.] // Высокопроизводительные параллельные вычисления на кластерных системах (HPC–2010): материалы X Междунар. конф. – Пермь: Изд-во Перм. гос. техн. ун-та, 2010.

Проведем анализ напряженно-деформированного состояния конструкции при максимальном значении коэффициента усиления K_y^{max}.

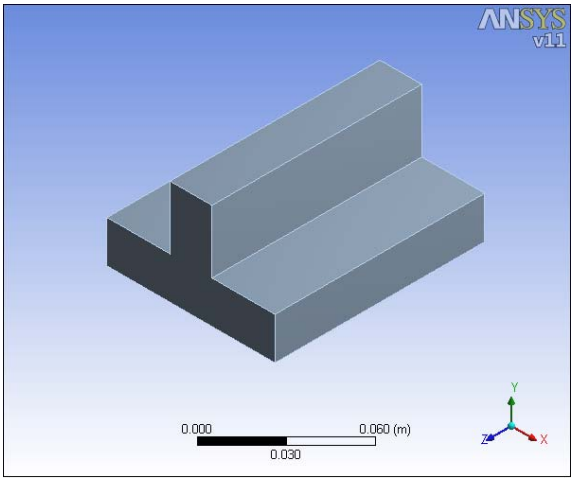
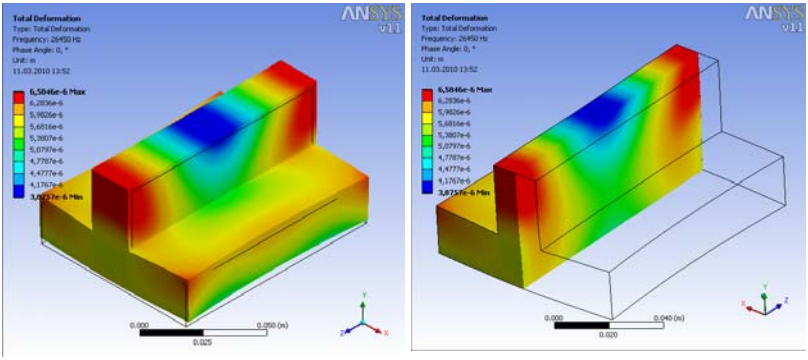


Рис. 1. Общий вид твердотельной модели АТ

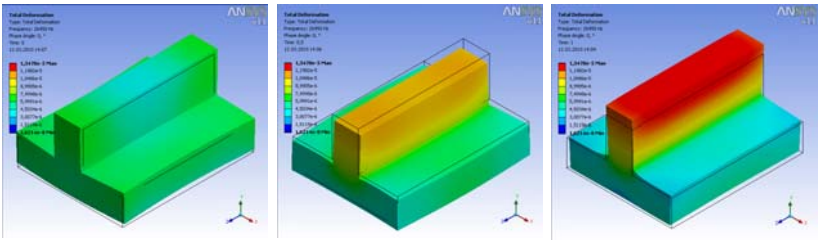
Рассмотрим поле перемещений акустического трансформатора (рис. 2, *а*, *б*). Очевидно, что максимальное значение перемещения величиной 6,6 мкм локализуется на периферии конструкции, в верхней ее части. В центре узкой грани значение перемещения минимально и равно 3,9 мкм.

Наблюдение за кинограммой перемещений (рис. 3, *а*, *б*, *в*) показало, что узкая грань при работе акустического трансформатора не сохраняет плоскости. Это снижает эффективность технологического процесса при использовании волноводов. При нагружении широкой грани синусоидальной нагрузкой с частотой 26,5 кГц наблюдается распространение волны сжатия в направлении узкой грани (оси *Y*). При этом волна сжатия достигает поверхности С раньше, чем поверхности А (рис. 4). Кроме того, наблюдается отражение волны сжатия от грани А. Запаздывание волны, отраженной от грани А, по сравнению с волной, отраженной от грани С, приводит к возникновению изгиба в волноводе в плоскости торцов.



а *б*

Рис. 2. Поле перемещений: *а* – на поверхности акустического трансформатора; *б* – в продольном сечении акустического трансформатора



а *б* *в*

Рис. 3. Кинограмма поля перемещений акустического трансформатора: *а* – $t_1 = 1$с; *б* – $t_2 = 2$с; *в* – $t_3 = 3$с

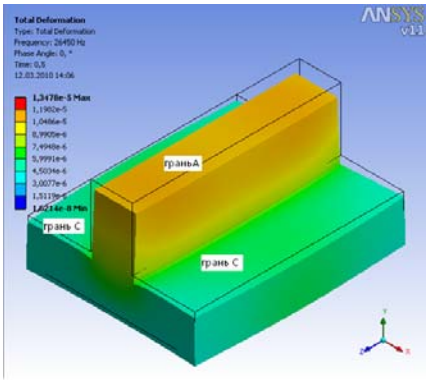
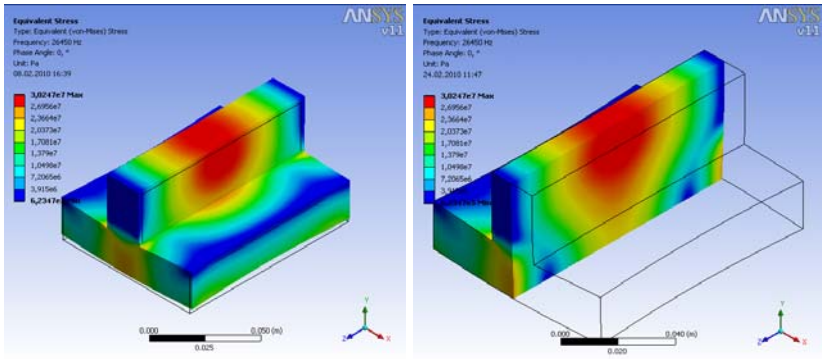


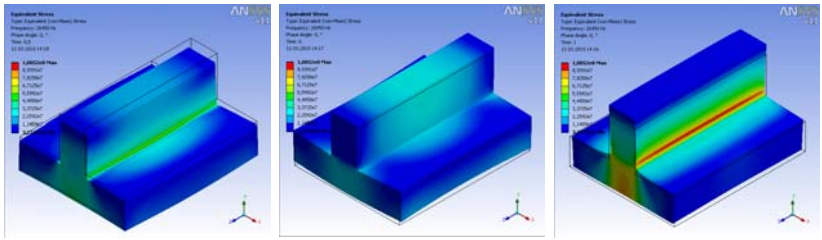
Рис. 4. Механизм перемещения граней акустического трансформатора

Результаты расчетов показали, что необходимая прочность обеспечивается. Сравнение максимальных напряжений, рассчитанных в вычислительном эксперименте ($\sigma_{max}^{ВЭ}$) при заданной частоте 26,5 кГц, равных 3 МПа показало, что они меньше допустимых напряжений ([σ]≈500 МПа) (рис. 5). Максимальные напряжения реализуются в геометрическом центре конструкции (рис.5, *а*, *б*), а также на поверхности узкой грани. По мере удаления от центра напряжения падают до минимального значения равного 0,6М Па. Распределения напряжений напоминают концентрические окружности (рис. 5, *а*, *б*).



а *б*

Рис. 5. Поле напряжений: *а* – на поверхности акустического трансформатора; *б* – в продольном сечении акустического трансформатора



а *б* *в*

Рис. 6. Кинограмма поля напряжений акустического трансформатора: а – t_1 = 1с; б – t_2 = 2с; в – t_3 = 3с

Кинограмма напряжений показывает, как формируется поле напряжений. Начиная из центра широкой грани, волна концентрируется в середине волновода, поднимаясь к узкой грани (рис. 6, *а*, *б*, *в*).

При медленном приложении нагрузки, можно увидеть поле максимальных напряжений, возникающих в конструкции (см. рис. 5, *а*, *б*).

По результатам проведенных расчетов можно сделать следующие выводы:

1. Определены параметры экспериментальной установки, позволяющие получить максимальные K_y для заданных физико-механических характеристик и геометрии акустического трансформатора.

2. В ходе вычислительного эксперимента подтверждена прочность акустического трансформатора при K_y^{max}.

3. Узкая грань при работе изгибается, что приводит к падению эффективности технологического процесса при использовании волноводов.

Вад.В. Воеводин

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

ХАРАКТЕРИСТИКИ РАБОТЫ С ПАМЯТЬЮ И ЭФФЕКТИВНОСТЬ РАБОТЫ ПРОГРАММ

На сегодняшний день существует много средств и технологий для повышения эффективности выполнения программ, однако в большинстве своем они ориентированы либо на работу в рамках конкретной аппаратуры, либо на оптимизацию определенного свойства программы. Представленная работа направлена на создание унифицированного подхода к изучению эффективности всего процесса отображения алгоритма на различные аппаратные платформы. Основной идеей является детальное описание трех этапов процесса отображения – алгоритмов, программ и их реализаций – с помощью набора характеристик, оценивая которые можно оценить и эффективность самого процесса

отображения. Некоторая часть данных характеристик является аппаратно-независимой, что позволяет абстрагироваться от выбранной платформы и исследовать эффективность процесса отображения алгоритма в общих терминах. Принципиальной позицией излагаемого подхода является то, что характеристики необходимо выделять на всех этапах отображения, начиная от свойств самого алгоритма и заканчивая характеристиками среды времени выполнения и аппаратуры. В докладе приводится описание общего процесса отображения, а также примеры характеристик на разных этапах.

На эффективность процесса отображения оказывают влияние многие факторы, и одним из самых важных среди них является эффективность работы с памятью. Организовать эффективную работу с памятью в программах непросто, поскольку на нее оказывает влияние множество различных параметров как самого алгоритма или программы, так и используемой аппаратуры. Память в современных вычислительных платформах имеет сложную структуру и иерархию, поэтому число параметров аппаратуры велико: размер и количество уровней кэш-памяти, объем и скорость чтения/записи в ОП, параметры работы контроллера памяти, размер строки, степень ассоциативности кэш-памяти, размер и строение TLB-буфера и т.д. Учитывать все эти параметры трудно, однако каждый из них может оказывать большое влияние на эффективность работы программы. Например, различие двух данных фрагментов кажется несущественным:

1)
```
for (i = 0; i < length; i+=1040)
        sum = sum + a[i];
```

2)
```
for (i = 0; i < length; i+=1024)
        sum = sum + a[i];
```

В обоих вариантах происходит суммирование элементов вещественного массива a длиной length, а единственное отличие заключено в шаге, с которым происходит выборка элементов массива – в первом случае шаг равен 1040, во втором – 1024. Проведенные на процессоре Intel Clovertown эксперименты по-

казали, что при длине массива в 2^{24} элементов первый вариант эффективнее второго в 8 раз! Причина – в ассоциативности кэш-памяти: во втором варианте требуемые данные постоянно попадают в одни и те же группы строк кэш-памяти, задействуя таким образом только малую ее часть, в то время как в первом варианте используется гораздо больший ее объем.

Рассмотрим второй пример, который отражает иные особенности работы с кэш-памятью:

```
1) for (i = 1; i < length; i++) {
        cache_annil[i] = i;
        helper += cache_annil[i-1];
}
<основной алгоритм>

2) for (i = 1; i < length; i++)
        cache_annil[i] = i;
   for (i = 1; i < length; i++)
        helper += cache_annil[i-1];
   <основной алгоритм>
```

Предположим, нас интересует время работы некоторого основного алгоритма, без учета служебных операций. Перед этим алгоритмом необходимо очистить кэш-память от записанных туда ранее данных для более точного измерения времени выполнения алгоритма. Для этого используется массив cache_annil,причем длина массива length подобрана так, чтобы заполнить всю кэш-память. Отличие двух вариантов только в том, что в первом варианте оба действия (запись нового значения в массив cache_annil и прибавление предыдущего элемента к общей сумме) происходят в рамках одного цикла, а во втором – в разных. Однако это приводит к увеличению времени выполнения, и, как следствие, неверной оценке работы основного алгоритма в первом примере, причем различие

во времени может быть существенным, в зависимости вычислительной сложности самого алгоритма. Причина в том, что в первом варианте весь массив cache_annil просматривается за один проход, и после этого в кэш-памяти остаются модифицированные данные, которые в основной части программы приходится вытеснять, что, естественно, отражается на эффективности этой основной части. Во втором варианте этого не происходит, поскольку модифицированные данные вытесняются в оперативную память до основного алгоритма, во время второго цикла.

Данные примеры демонстрируют разнообразие факторов, которые могут влиять на эффективность процесса отображения. Важной особенностью данных факторов, или характеристик аппаратуры, является то, что чаще всего их нельзя изменять. Программу необходимо составлять таким образом, чтобы в ней учитывались особенности выбранной аппаратуры. А для этого необходимо выделить те характеристики алгоритма и программы, которые могут влиять на работу с памятью, после чего, изменяя их, можно подстраивать программу под аппаратуру. Примерами таких характеристик являются пространственная и временная локальности, общее число обращений в память и т.д. На их исследование и описание и направлена данная работа.

Составим формальное описание работы с памятью. На данный момент будем рассматривать последовательные программы. Работу программы с памятью можно представить в виде потока, который описывается последовательностью обращений к используемым в программе переменным в том порядке, в котором это происходит во время работы программы. Если задействован элемент массива или другой составной переменной, то учитывается именно этот элемент, а не вся составная переменная. Таким образом, в последовательность могут входить как скалярные переменные, так и составные (если в некоторой операции адресуется такая переменная целиком) или их элементы.

Если для работы с элементами составной переменной используется некоторый итератор, то для каждого конкретного зна-

чения итератора в поток обращений записывается отдельное обращение. Например, в случае итерирования массива, если в качестве индекса массива A используется некоторая переменная i, то в потоке обращений необходимо для каждого значения i подставлять это конкретное значение. Например, если в программе есть код

for (i=0;i<5;i++) a[i]=0,

то поток обращений к массиву A будет выглядеть следующим образом: a[0],a[1],a[2],a[3],a[4].

Каждое обращение в потоке A можно поменять на виртуальный адрес соответствующей переменной, получив тем самым поток виртуальных адресов. В этом случае происходит переход на другой уровень абстракции, поскольку работа происходит не с переменными, а с адресами. Далее, при замене виртуальных адресов на физические формируется поток физических адресов. Формулируя и рассматривая характеристики задачи на каждом из уровней (переменные, виртуальные и физические адреса), можно получать описания поведения задачи на разных уровнях абстракции и зависимости от конкретной аппаратуры. Уровень физических адресов позволяет получить наиболее точное описание, поскольку оперирует реальным расположением адресуемых данных в памяти во время выполнения программы, однако его сложнее учитывать, так как расположение данных может изменяться от запуска к запуску.

В докладе вводится более подробное описание данных уровней абстракции и отдельно рассматривается уровень переменных, в рамках которого формулируются некоторые характеристики работы с памятью, определяющие свойства пространственной и временной локальности данных. Далее приводится объяснение, каким образом данные характеристики могут влиять на эффективность работы с памятью, и показывается важность анализа этих характеристик для получения эффективного процесса отображения.

Работа выполнена при поддержке гранта РФФИ № 09-07-00168-а и Федерального агентства по науке и инновациям, государственный контракт № 02.740.11.0388.

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

[1]Институт проблем химической физики РАН, Московская обл., Черноголовка
[2]Институт экспериментальной минералогии РАН, Московская обл., Черноголовка

РЕАЛИЗАЦИЯ НЕСКОЛЬКИХ НЕЗАВИСИМЫХ РЕСУРСНЫХ ГРИД-САЙТОВ НА ЕДИНОМ ФИЗИЧЕСКОМ ПРОСТРАНСТВЕ КЛАСТЕРА

При организации работ в различных распределенных сетях периодически возникает ситуация, когда необходимо провести тестирование настроек и функциональности ресурсного грид-сайта в условиях распределенного вычислительного полигона еще ДО формирования и подключения собственно высокопроизводительного вычислительного ресурса (крупного кластера, «фермы», сети рабочих станций и т.п.). Иногда необходимо проведение различных вычислительных экспериментов в условиях распределенных сред, при том, что использование основных вычислительных ресурсов организации в таких случаях не всегда желательно. Кроме того, достаточно часто необходимо провести тестовые запуски исходящих задач (особенно в случае новых версий сложных прикладных пакетов) на собственных ресурсах (используемых как удаленные) для отладки, выявления ошибок и проблем, связанных с выполнением задач в грид-средах. Это не всегда возможно как из-за трудоемкости/невозможности перенастройки действующего ресурса (включая конфликты приложений), так и из-за отсутствия такового (размещенного в рамках выбранной грид-сети). Для преодоления подобных проблем авторами была разработана методика размещения небольших ресурсных грид-сайтов разных вычислительных полигонов на едином физическом пространстве небольшого кластера.

Постановка задачи. Ранее в ИПХФ РАН реализованы и успешно эксплуатируются несколько ресурсных сайтов различных распределенных грид-полигонов [1–3]:

1) ресурсный узел консорциума EGEE-RDIG (Enable GRID for E-sciencE и Russian Data Intensive GRID, http://www.egee-rdig.ru) на основе распределенной среды gLite (http://glite.web.cern.ch), работы ведутся в рамках виртуальной организации (ВО) RGSTEST;

2) ресурсный сайт категории «А» для работы в рамках созданного в России крупномасштабного вычислительного полигона СКИФ-Полигон (http://skif-grid.botik.ru) на базе промежуточного ПО Unicore (http://www.unicore.eu);

3) сайт Национальной нанотехнологической сети (ГридННС, http://www.ngrid.ru), среда Globus Toolkit 4, http://www.globus.org).

Созданные ресурсные сайты в первую очередь позволяют решать *входящие* задачи как общего характера, так и с использованием прикладных квантово-химических пакетов.

В состав ресурсных сайтов помимо ряда грид-сервисов входит также комплекс клиентских интерфейсов различных уровней для взаимодействия адаптированного квантово-химического прикладного ПО с указанными грид-средами. Они позволяют запускать *исходящие* задачи вычислительной химии на распределенных ресурсах, обеспечивая возможность формирования заданий, запуск на удаленных сайтах (через брокеры ресурсов или в режиме point-to-point -> P2P), мониторинга прохождения заданий, сбора результатов и статистики.

Работа в рамках ВО RGSTEST сети EGEE-RDIG обеспечивает доступ к вычислительным мощностям до 800 процессоров и дисковым массивам порядка 8–15 терабайт в нескольких географических зонах (Москва, Протвино, Харьков, Черноголовка и др.). Разнородность узлов данной ВО позволяет легко варьировать параметры запускаемых задач, ориентируясь на различные типы ресурсов. Использование подобного полигона обеспечивает проведение достаточно масштабных вычислительных экспериментов как научного, так и прикладного характера.

Ресурсный сайт для среды Unicore позволяет выполнять входящие задачи сертифицированных пользователей СКИФ-Полигона, производить мониторинг задач и передавать полу-

ченные результаты пользователям. Обеспечена возможность мониторинга состояния сайта извне. Клиентский интерфейс обеспечивает запуски исходящих задач через среду Unicore на собственном ресурсном сайте ИПХФ (в роли удаленного ресурса – https://unicorgw.icp.ac.ru:8080) и на доступных ресурсных узлах СКИФ-Полигона.

В рамках создаваемой с 2010 года Национальной нанотехнологической сети (ГридННС) предоставлен доступ к ресурсам с общим числом CPU более 8000 (http://mon.ngrid.ru/stats?page=usage) и большим количеством виртуальных организаций, в том числе поддерживающих квантово-химические расчеты (на базе сайта ИПХФ создана и функционирует ВО nanochem).

Исходя из опыта эксплуатации созданных ресурсных сайтов, авторами было принято решение о создании небольшого экспериментального кластера для проведения вычислительных экспериментов и разработку грид-приложений без вовлечения основных ресурсов ИПХФ РАН. Кроме того, данная работа позволила в полной мере оценить степень применимости новейших технологий виртуализации ресурсов для обслуживания грид-сервисов и грид-интерфейсов.

Реализация вычислительного кластера. Был установлен и настроен вычислительный кластер из 4 двухядерных расчетных узлов (с перспективой ближайшего роста до 8 узлов) с общим количеством ядер 16 и трех сопровождающих управляющих машин, опирающихся на сеть Gigabit-Ethernet. На расчетных узлах был установлен дистрибутив Scientific Linux 5.4 Boron, поскольку наличие данного дистрибутива требуется для расчетных узлов ресурсного сайта gLite, а для других распределенных сред выбор ОС (варианты Linux) не принципиален. Для проведения вычислительных экспериментов на расчетные узлы был установлен и настроен ряд прикладных параллельных квантово-химических пакетов (Gamess-US, Gaussian, Dalton-2, NAMD). Для обеспечения параллельных вычислений использовались как сокетные версии пакетов, так MPI-варианты. На управляющих машинах был установлен гипервизор KVM (Kernel-based Virtual Machine, http://www.linux-kvm.org). На ста-

дии изучения потенциальных вариантов гипервизоров для управляющих машин были протестированы несколько свободно распространяемых гипервизоров виртуальных машин – VirtualBox, VMware, Xen [4], однако, исходя из простоты администрирования, минимума накладных расходов ресурсов и устойчивости работы под нагрузкой в качестве гипервизора был выбран KVM. После установки гипервизора на управляющих машинах были установлены следующие сервисные виртуальные машины:

1) для Computing Element среды gLite 3.1 (полигон EGEE-RDIG) – Scientific linux 4,5 и соответствующие сервисы типа lcg-ce, сервисов авторизации и мониторинга;

2) для среды Unicore 6.2 (сайт СКИФ-Полигона) – ОС Ubuntu 9.10, сервисы – шлюз (Gateway); серверный контейнер (Unicore/X), интерфейс к целевой системе (TSI), авторизационный сервис-пользовательская база данных – XUUDB$, пользовательский интерфейс (UI);

3) для среды Globus Toolkit 4 (полигон ГридННС) – ОС CentOS 5.4, сервисы MDS, GRAM, GridFTP, RFT, User Interface.

Выбор соответствующих ОС для управляющих узлов обусловлен либо требованиями дистрибутивов распределенного ПО (как, например, для gLite), либо рекомендациями разработчиков, либо простотой администрирования. На все управляющие машины были установлены серверные компоненты пакета управления заданиями PBS/Torque (http://www.clusterresources.com), тогда как на расчетные узлы была установлена клиентская часть данного пакета. Поскольку все распределенные middleware требуют наличия своих собственных очередей PBS, было принято решение о настройке трех одновременно работающих экземпляров pbs_mom на расчетных узлах с соответствующим набором очередей заданий. В таком варианте каждая управляющая машина связывается с расчетными узлами по уникальному порту и имеет дело только со своими заданиями. Недостатком данного подхода является невозможность (пока) правильного учета ресурсов, используемых расчетным узлом, однако для экспериментальных работ и отладки прикладного ПО это вполне приемлемо. Преимуществами же являются:

1) необходимость ОДНОКРАТНЫХ установки и настройки ПО для решения входящих задач (особенно прикладных, поскольку часто установка прикладных пакетов оборачивается непредвиденными трудозатратами);

2) простота администрирования расчетных узлов;

3) экономия ресурсов (включая прежде всего электроэнергию);

4) повышенный коэффициент загрузки за счет лучшей утилизации CPU.

Преимуществами использования виртуальных машин в качестве управляющих является повышенная надежность ресурсного узла. В случае поломки физического узла копия виртуальной машины (размещенная на другом узле) может быть запущена в считанные минуты. При наличии достаточного объема ресурсов (особенно оперативной памяти) все три виртуальные машины могут быть размещены на одном физическом узле (подобные опыты ставились).

Отметим, что попутно на управляющих машинах можно разместить дополнительные виртуальные машины, отвечающие за различные нересурсоемкие сервисы сети, поскольку их влияние на основные сервисы незначительно.

В ИПХФ разработана методика одновременного размещения нескольких ресурсных сайтов разных распределенных полигонов на едином физическом пространстве одного кластера. Это осуществляется применением технологий виртуализации ресурсов с использованием гипервизора KVM и разделенных экземпляров пакетов управления заданиями PBS/Torque. Подобная методика позволяет проводить обширные вычислительные эксперименты и разрабатывать сложные грид-приложения для различных грид-полигонов без вовлечения основных вычислительных ресурсов организации. Введение подобных вариантов использования кластеров позволяет участвовать в работе нескольких грид-полигонов, значительно повышает надежность управляющих узлов ресурсных сайтов, снижает трудоемкость администрирования сайтов, понижает расходы на поддержку грид-инфраструктуры.

Список литературы

1. Технологии ГРИД в вычислительной химии / В.М. Волохов [и др.] // Вычислительные методы и программирование. – 2010. – Т. 11, № 1. – с. 42–49.

2. Вычислительная химия в грид-средах / В.М. Волохов [и др.] // Computational Chemistry in the Grid Environments // Distributed Computing and Grid-technologies in science and education: 4th Int. Conf. Dubna, JINR, 2010. P.143–144.

3. GRID и вычислительная химия / В.М. Волохов [и др.] // Вычислительные методы и программирование. – 2009. – Т. 10, № 2. – С.78–88.

4. Виртуальные вычислительные среды: использование на GRID полигонах / В.М. Волохов [и др.] // Вестн. ЮУрГУ. Серия «Математическое моделирование и программирование». – 2009. – № 17 (150). Вып. 3. – С. 24–35.

Н.И. Гаврилов, А.А. Белокаменская

Нижегородский государственный университет им. Н.И. Лобачевского

ОРГАНИЗАЦИЯ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ НА GPU В ЗАДАЧЕ СТЕРЕОВИЗУАЛИЗАЦИИ ТОМОГРАММ

В научной визуализации часто приходится иметь дело со скалярными полями данных, заданными в трехмерном пространстве. Данные могут быть получены самыми различными способами – численным экспериментом, сканированием с помощью магнитного резонанса, компьютерной томографией (КТ), сканированием с помощью ультразвука. Например, результатом КТ является множество слоёв, т.е. двумерных массивов данных, которые вместе образуют трехмерное скалярное поле. Объем данных эксперимента значителен. Например, объем одного исследования в медицинской томографии обычно равен 0,25–1 Гб. Обеспечение реального времени визуализации означает вывод на экран порядка 25 и более кадров в секунду.

Метод прямого объёмного рендеринга методом обратной трассировки лучей с 90-х годов прошлого века позиционирует себя как эффективный инструмент для визуального анализа объёмных данных. Разработаны и доведены до широкой научной общественности [1] подходы, позволяющие решать задачи объемного рендеринга в реальном времени на основе параллельных вычислений и высокопроизводительной вычислительной техники. Эффективно решаются в интерактивном режиме задачи сегментации объема [2]. Развитие GPU как вычислительных устройств в последние годы дало этому направлению новый значительный импульс и позволило говорить о многообъемном рендеринге в реальном времени [3–5]. Стереовизуализация научных данных достаточно давно культивируется в Бостонском университете (http://scv.bu.edu), но не вошла пока в широкую практику в медицине.

Методы визуализации. В медицине томограммы хранятся в специальном формате данных DICOM, а для визуализации томограмм используют так называемые DICOM-визуализаторы. Довольно большая часть их общедоступна, в том числе: eFilm, MicroDicom, OsiriX, Onis-Viewer, RadiAnt и отечественный MultiVox DICOM Viewer (www.multivox.ru). Все эти Viewers обеспечивают визуализацию томограммы в виде классических полутоновых изображений 2D-проекций и сечений объёмного скалярного поля. Большинство Viewers обеспечивают также возможность визуализации 3D-моделей томограммы, используя чаще всего простые модификации прямого объемного рендеринга.

Наилучшего качества и информативности объёмного рендеринга позволяет достичь метод обратной трассировки лучей. Лучи обрабатываются независимо друг от друга, в алгоритме преобладают векторные операции, можно не делать большого количества ветвлений в алгоритме. Таким образом, GPU хорошо подходит для задачи объёмного рендеринга с использованием трассировки лучей. С помощью этого метода можно визуализировать данные как набор изоповерхностей или в виде тумана, или и то и другое вместе. У изоповерхностей и тумана можно варьировать прозрачность

и цвет. В научном эксперименте и дорогих профессиональных системах можно встретить примеры мультиобъемного рендеринга, к которому пока не предъявляют требования реального времени.

Высокое качество изображения формирует интерес к использованию стереоизображений, в которых так же, как в прямом рендеринге, используется цвет и прозрачность вокселей.

Реализация трассировки лучей для стереовизуализации объёмов на GPU. Наиболее удобной структурой хранения пространственных массивов данных на GPU является 3D-текстура, аналогичная 3D-массиву действительных чисел. Для CPU удобнее может оказаться иерархическая структура, например, октодерево. Для доступа к элементам дерева процессору необходимо пройти множество ветвлений, что приемлемо для CPU, но убивает производительность для GPU, к тому же в программах для GPU нет рекурсии. Программа, реализующая трассировку, выполнена в среде MS Visual Studio 2008, как шейдерная программа для GPU, в среде OpenGL и GLSL (шейдерный язык для OpenGL).

Предложен метод ускорения вычислений, заключающийся в использовании иерархии текстур: помимо основного массива 512×512×512 в GPU загружается меньший массив – двухканальная текстура 16×16×16. В пространстве каждому элементу этой текстуры соответствует свой блок 32×32×32 исходных данных. Текстура хранит минимальное и максимальное значения данных соответствующего блока.

Обращаясь к маленькой текстуре, луч может пропускать «неинтересные» области пространства, двигаясь с большим шагом. Около 20–30 % лучей так и не найдут «интересные» области, что иногда повышает FPS в три раза.

Различные алгоритмы трассировки луча. Для каждого пикселя искомого изображения генерируется луч, который проходит через объёмные данные с некоторым шагом. На каждом шаге луч может накапливать цвет. Таким образом, алгоритм накопления цвета определяет получаемое изображение. Рассмотрим алгоритм визуализации изоповерхности.

Визуализация изоповерхностей для объёмных данных – это довольно информативный визуальный анализ. Например, для визуализации костной ткани для данных CT-снимка достаточно построить изоповерхность для значения 500 (рентгеновская плотность кости по шкале Хаунсфилда +400 и выше). Цвет в соответствующем алгоритме накапливается лучом только при пересечении изоповерхности. Пересечение имеет место, если на соседних шагах луча выборки значений данных лежат по разные стороны изозначения. Далее численно вычисляем нормаль и определяем цвет, используя локальную модель освещения. Нормаль к изоповерхности определяется как нормированный градиент, который вычисляется численно, по разностной схеме. Помимо изоповерхностей можно также визуализировать контуры, проекции максимальной интенсивности и т.д.

Код программ для GPU. Каждый способ визуализации на практике представляет собой ту или иную шейдерную программу. Размер каждой из них составляет порядка 300 строк, 200 из которых составляют различные функции, одинаковые для всех алгоритмов. Среди таких функций – выборка данных в точке вычисления, в точке нормали, локального освещения, глубины и т.д. Остальные 100 строк занимает функция main, в которой и происходит трассировка луча из пикселя для вычисления цвета и глубины (для буфера глубины). Эта функция своя для каждого метода трассировки, поэтому загружаемая в GPU программа склеивается из заголовочной части, которая содержит функции и различные переменные, и основной части, где содержится функция main. Также во избежание дублирования кода без ущерба производительности происходит предварительная обработка текста шейдерной программы: например, вместо последовательности символов, начинающихся с ‘$’, подставляются конкретные значения. Ниже приведён пример: функция вычисления глубины, которая в зависимости от того, перспективная проекция используется или ортогональная, по-разному вычисляет глубину (pos – позиция наблюдателя, ps – позиция точки, глубину которой мы вычисляем, nav – ориентация камеры).

```
float GetDepth(vec3 ps)
{
      ps -= pos;
#if $IsPerspective==1
      return      clamp(z_far/(z_far-z_near)      -
(z_far*z_near/(z_far-
z_near))/(dot(ps,nav)),0.01,0.99);
#else
      return     clamp((dot(ps,nav)-z_near)/(z_far-
z_near),0.01,0.99);
#endif
}
```

Вычисление глубины изображения нужно для правильного вывода остальных объектов сцены: рамка ограничивающего объёма, секущие плоскости и т.д.

Ниже приведена часть шейдерной программы для визуализации непрозрачной изоповерхности. Здесь луч идёт с небольшим шагом step и на каждом шаге непосредственно обращается к исходным данным через функцию EquF. Выход из цикла происходит, либо если луч вышел за пределы ограничивающего объёма, либо при пересечении изоповерхности с изозначением min_level (эта часть кода для случая, когда изначально луч находился в точке, значение данных в которой было меньше min_level).

```
while(ps == clamp(ps,box1,box2))    //пока    внутри
объёма с данными
{
      e0=e;                         //храним  старое
значение выборки из данных
      e = EquF(ps);                 //выборка     из
объёмных данных

      if(e >= min_level)            //если
пересекли изоповерхность
      {
```

```
            ps    +=    step*((min_level-e)/(e-e0));
      //уточняем точку пересечения
            norm     =     -normalize(GradEqu(ps));
      //вычисляем нормаль
            color                                    =
Phong(ps,norm,vec3(0.7,0.7,0.7));
            gl_FragDepth       =       GetDepth(ps);
      //вычисляем глубину точки ps
            break;
      }
      ps+=step;          //движение луча
}
```

Функции PhongL и Phong вычисляют цвет в точке ps по модели освещения Фонга, однако первая функция к тому же увеличивает либо уменьшает яркость цвета в зависимости от того, заслонена или нет точка ps изоповерхностью. Замена простых условий if на препроцессорные #if нужны для уменьшения размеров скомпилированной шейдерной программы и ускорения самой компиляции. Для режима вывода множества полупрозрачных изоповерхностей используется массив их изозначений и оптических свойств (цвет и прозрачность).

Ещё пример: использование ускоряющей структуры. Перед циклом выше, где идёт накопление цвета, луч проходит через пустые блоки исходных данных. Обращение при этом идёт только к данным ускоряющей структуры, а шаг луча не фиксирован, чтобы пройти все ячейки ускоряющей структуры по пути трассировки.

```
#if $use_accel_struct==1      //если          хотим
использовать ускоряющую структуру
      for(float i=0.0;i<100.0;i++)
      {     //выходим  из  цикла,  если  вышли  за
            пределы     данных,     либо     нашли
            //“интересную” область
```

```
            if(!(ps   ==   clamp(ps,box1,box2))   ||
IEqu(ps)>=min_level)break;
            //шаг   луча   (переход   к   следующему
            пересечению      ячеек      ускоряющей
            //структуры)
            ps=GetCellStep(ps,ray);
      }
#endif
```

В целом использование такой оптимизации ускоряет рендеринг в 1,5–2,5 раза в зависимости от стоимости трассировки луча (визуализация контуров в 5–7 раз дороже визуализации непрозрачной изоповерхности). Пропуск пустых областей происходит только в начале – это оптимальное решение для GPU. Если искать пустые области и дальше, после прохода «интересных», то мы получим только проигрыш в производительности, поскольку постоянное обращение к ускоряющей структуре во время накопления цвета сведёт на нет выигрыш от повторного нахождения пустой области. На рисунке выигрыш от ускоряющей структуры составил 1,5, здесь лучи почти сразу начинают движение в «интересных» областях данных.

Ниже приведены результаты замера производительности (числа кадров в секунду) различных методов рендеринга на различных видеокартах. Информация о центральном процессоре отсутствует, поскольку CPU никак не влияет на производительность рендеринга.

– Размер окна вывода: 1280×1024 (во весь экран).

– Тестовые данные: 512×512×512, 16-битные.

– Длина шага луча: 0,0004 (0,2 от размера вокселя).

Ракурс крупным планом (см. рисунок, *а*), поэтому общее число шагов всех лучей оценивается величиной 1280·1024/0,0004 = = 3,2·10^9. Благодаря раннему прекращению движения луча из-за накопления достаточной непрозрачности это число уменьшается на порядок (таблица).

Результаты замера производительности

	Fog + Isos	Isos	MIP	Shaded Fog	Fog	Iso
NVIDIA GeForce GTS 250	9	10	10	12	12	30
NVIDIA GeForce 9500 GT	4	4	4	5	6	13
NVIDIA Quadro FX 5600	14	16	19	23	25	63
NVIDIA GeForce 8600 GT	3	4	2	4	4	9
ATI RADEON HD 4870	18	21	35	38	40	81
ATI RADEON HD 4890	21	25	39	44	49	108

В первой строке таблицы перечислены различные методы рендеринга:

Fog – туман (закраска пространства в соответствии с передаточной функцией (transfer function));

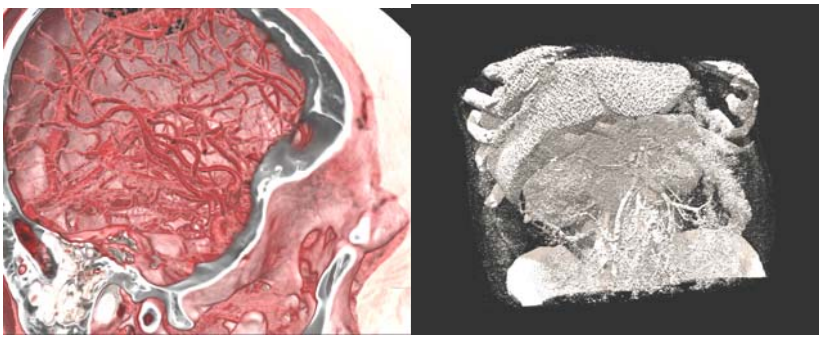
Isos – множество полупрозрачных изоповерхностей (в тесте их 3);

Fog + Isos – комбинация тумана и изоповерхностей;

Iso – непрозрачная изоповерхность;

MIP – проекция максимальной интенсивности;

Shaded Fog – затенённый туман (на цвет в точке влияет не только значение данных, но и градиент в данной точке).



а *б*

Рис. 3D-визуализация в программе SMV. Техники рендеринга: «затенённый туман» (*а*) и непрозрачная изоповерхность с тенью (*б*)

Из таблицы видно, что видеокарты ATI явно превосходят видеокарты NVIDIA в задаче трассировки лучей, что объясняет-

ся наличием большого числа векторных операций в алгоритмах, а видеокарты ATI как раз и ориентированы на выполнение векторных операций, тогда как ядра карт NVIDIA могут поодиночке выполнять скалярные операции.

Для сравнения производительности OpenCL и GLSL-шейдеров написаны две программы, реализующие алгоритм без ускоряющей структуры, рисующий одну изоповерхность для плотности 65 (плотность кости) и освещенный туман для данных всего диапазона плотностей. Протестировано на 32-битных данных размером 256^3. Для проведения эксперимента использовался графический ускоритель NVIDIA GeForce 8600GT и центральный процессор Intel Core2Duo 2,33 GHz. Размер окна визуализации составлял 512×512 точек. Производительность на OpenCL равна 5,6 fps, на GLSL – 10,2 fps. Таким образом, для некоторых задач реализация на шейдерах пока ещё остаётся оптимальным вариантом.

Авторы благодарны проф. В.Е.Турлапову за постановку задачи и участие в обсуждении результатов. Работа выполнена при финансовой поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

Список литературы

1. Klaus Engel, Markus Hadwiger, Joe M. Kniss, Aaron E. Lefohn, Christof Rezk Salama, Daniel Weiskopf. Real-Time Volume Graphics, SIGGRAPH–2004. – URL: http://old.vrvis.at/via/resources/course-volgraphics-2004.

2. Построение 3D модели кровеносных сосудов по серии КТ-изображений печени / А.М. Ятченко [и др.]. Conf. Proc. of the 19th International Conference on Computer Graphics and Vision «GraphiCon'2009», Moscow, 2009.

3. Friedemann Roßler, Eduardo Tejada, Thomas Fangmeier, Thomas Ertl, Markus Knauff. GPU-based Multi-Volume Render-

ing for the Visualization of Functional Brain Images. – URL: http://www.vis.uni-stuttgart.de/ger/research/pub/pub2006/simvis06-roessler.pdf.

4. Flexible GPU-Based Multi-Volume Ray-Casting / R. Brecheisen [et al.] // Technical University of Eindhoven. – URL: http://www.yp. wtb.tue.nl/pdfs/9881.pdf

5. Beyer Johanna. GPU-based Multi-Volume Rendering of Complex Data in Neuroscience and Neurosurgery. – Dissertation. – Vienna University of Technology, okt. 2009. – 131 p.

6. The scientific computing and imagine institute at the university of Utah. – URL: http://www.sci.utah.edu/download/ IV3DData.html.

Р.К. Газизов, И.Р. Фатхулисламов

Уфимский государственный авиационный технический университет

GPU И MATLAB ДЛЯ РЕШЕНИЯ ЗАДАЧИ АВТОМАТИЧЕСКОЙ ГЕОГРАФИЧЕСКОЙ ПРИВЯЗКИ СПУТНИКОВЫХ ИЗОБРАЖЕНИЙ С ПИКСЕЛЬНОЙ ТОЧНОСТЬЮ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

В связи с резким увеличением количества графических карт для персональных компьютеров и рабочих станций графический процессор (GPU) стал платформой для параллельных вычислений, доступной для широкого круга пользователей. Учитывая его высокую пропускную способность памяти и высокую вычислительную производительность, GPU все чаще стали использовать для расчетов общего назначения (GPGPU). Множество удачных примеров использования GPU для реализации различных задач показаны в работе [1].

На сегодняшний день задача автоматической географической привязки спутниковых изображений является актуальной, так как в последнее время наблюдается устойчивая тенденция к увеличению объема получаемой информации из космоса и решение данной задачи требует значительного времени.

На первой стадии процесса получения продуктов обработки спутниковых данных выполняется первичная обработка цифровых изображений. Обязательным этапом обработки полученных данных является географическая привязка спутниковых изображений. От точности решения задачи географической привязки спутниковых изображений зависит решение множества других задач, например, коррекция геометрических искажений, синтез разнородных спутниковых изображений, оценка временной изменчивости ТПО, построение карт морских течений, определение скорости передвижения облачных масс, локализация пожаров и т.д. Постоянный рост числа запускаемых спутников, появление новых тематических задач делает необходимым, чтобы процедура географической привязки выполнялась автоматически и с высокой скоростью.

Решение задачи точной географической привязки спутниковых изображений невозможно без вычисления контрольных точек на изображениях. В данной статье рассмотрена параллельная реализация алгоритма Харриса [2] на языке программирования Matlab с использованием GPU.

Некоторые работы по использованию GPU в Matlab можно найти в [3]. В данной работе был использован коммерческий продукт Jacket [4]. Это набор библиотек для MATLAB, который позволяет использовать стандартный код MATLAB для работы на NVIDIA CUDA при поддержке GPU.

В наших экспериментах мы использовали ПК следующей конфигурации: Intel Core Duo E4600@2.4GHz, 8Gb RAM, NVIDIA GTX 280. Ниже показаны аппаратные особенности NVIDIA GTX 280:

Пиковая вычислительная производительность (TFlops)	0,62
Полоса пропускания памяти (ГБ/с)	141,7
Stream процессоры	240
Частота ядра (МГц)	602
Частота памяти (МГц)	1107
Объем памяти	1G
Общая память	16kB на SM

Во всех наших экспериментах при вычислении процессорного времени использовали соответствующие функции MATLAB. Результаты реализации алгоритмов на GPU сверены с результатами, которые были получены в MATLAB при расчетах на процессоре. Была произведена оптимизация исходного кода под использование данного GPU. Все изображения, которые были использованы в наших тестах, получены со спутников NOAA/AVHRR.

Одной из функций при использовании алгоритма Харриса является функция $conv2$, которая производит свертку исходного изображения и фильтра. Функция принимает в качестве входных параметров исходное изображение A и фильтр B, на выходе – изображение C. Формула для расчета пикселя $C_{m,n}$ выглядит следующим образом: $C[m,n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} B[j,k] A[m-j, n-k]$, где J и K – ширина и высота фильтра соответственно. Если фильтр соизмерим с исходным изображением, то мы можем использовать быстрое преобразование Фурье (БПФ), которое доступно в CUDA SDK. Однако в нашем случае размер фильтра 3×3 пикселя и применять БПФ нецелесообразно.

Учитывая небольшой размер регистровой памяти, мы хранили исходный фильтр в общей памяти, а не выделяли 9 регистров для каждого потока. Для хранения исходного изображения использовали текстурную память. Исходное изображение было разделено на колонки, равные количеству потоков. Каждый поток обрабатывал свою колонку.

Пропускная способность ядра 352 GFLOPS достигается при разрешении изображения 4096×4096 пикселей. Общее ускорение данного метода в 3,2 выше, чем при расчетах на CPU. Данный подход реализации функции $conv2$ на CUDA быстрее, чем поставляемая с коммерческим пакетом Jacket. Даже если не учитывать время на передачу данных между CPU GPU, двумерная свертка с ядром 3×3 пикселя на изображении 4096×4096 пикселей с использованием пакета Jacket занимает 0,114 с, в то время как оптимизированный код на CUDA и MEX занимает 0,091 с (учитывая время на передачу между

CPU и GPU). Как видно из рис. 1, используемый при расчетах CPU дает существенный прирост производительности.

Рис. 1. Зависимость отношения ускорений GPU/CPU от разрешения изображения (функция *conv*2)

Рис. 2. Зависимость отношения ускорений GPU/CPU от разрешения изображения (алгоритм Харриса)

Распараллелив функцию *conv*2, общее ускорение алгоритма Харриса можно видеть на рис. 2. Как видно из полученных результатов, распараллелив функцию *conv*2 можно добиться ускорения на изображениях с разрешением выше, чем 512×512 пикселей.

Общий алгоритм автоматической географической привязки спутниковых изображений с пиксельной точностью был рассмотрен авторами ранее [5]. В данной статье выполнена параллельная реализация алгоритма Харриса с использованием GPU. Алгоритм был реализован с использованием среды программирования Matlab и библиотеки Jacket. Была произведена оптимизация исходного кода под использование данного GPU. Как видно из полученных результатов, оптимизация алгоритма под GPU дает прирост производительности на 20 %. Проведенные вычислительные эксперименты позволяют говорить об эффективности использования GPU для решения задачи поиска контрольных точек на изображениях. В дальнейшем планируется реализовать параллельный алгоритм на GPU других функций алгоритма и проверить алгоритм на различных графических картах.

Список литературы

1. GPU Computing / J.D. Owens [et al.]. Proceedings of the IEEE, 2008.

2. Harris C. and Stephens M. A combined corner and edge detector. In Proceedings, 4th Alvey Vision Conference. – Manchester, 1988. – P. 147–151.

3. http://developer.NVIDIA.com/object/MATLAB_cuda.html

4. http://www.accelereyes.com/products

5. Газизов Р.К., Фатхулисламов И.Р. Автоматическая географическая привязка спутниковых изображений с пиксельной в задаче минимизации объема хранимой информации // Перспективные информационные технологии для авиации и космоса: тр. междунар. конф. с элементами научной школы для молодежи. – Самара, 2010. – С. 748–752.

К.С. Галягин, А.С. Ипанов, М.А. Ошивалов, Ю.А. Селянинов

Пермский государственный технический университет

ОПТИМИЗАЦИЯ ПРОТОЧНОГО ТРАКТА МНОГОФАЗНОГО ЭЖЕКТОРА ДЛЯ ПЕРЕКАЧКИ ПРОДУКЦИИ НЕФТЕДОБЫВАЮЩИХ СКВАЖИН

Использование аппаратов со струйными течениями (струйных насосов) позволяет создавать технологические установки, имеющие ряд преимуществ, обусловленных их предельной простотой и компактностью, отсутствием движущихся частей, высокой надежностью работы, а также возможностью проведения на них одновременно нескольких технологических процессов. Указанные преимущества открывают широкие перспективы создания нового типа многофункциональных малогабаритных устройств для технологических систем нефтегазодобывающей и нефтеперерабатывающей отраслей промышленности.

В настоящее время струйные насосы (инжекторы) нашли применение в составе дожимных насосных станций (ДНС) для перекачки продукции нефтедобывающих скважин с одновременной утилизацией попутного газа и транспортированием газожидкостной смеси (ГЖС) по единому трубопроводу. Проектирование установок требует решения ряда конструкторских, технологических и оптимизационных задач с целью обеспечения максимальной эффективности работы устройств. Необходимо рассчитывать термогазодинамические процессы, происходящие в различных типах струйных течений: свободно истекающих, эжекционных, кавитационных, пульсационных, вихревых, а также анализировать компонентный состав, скорости, температуры, давления и другие значения термодинамических и физических параметров смеси.

Существующие методики расчета струйных насосов дают возможность лишь приближенно оценить их интегральные ха-

рактеристики и не позволяют исследовать гидродинамику течения в условиях перекачки ГЖС. Детальное описание исследуемых физических явлений приводит к сложной математической модели тепломассообмена, реализация которой возможна лишь с использованием современных численных пакетов программ. Одним из наиболее эффективных для решения подобных задач является программный продукт STAR-CD компании CD-ADAPCO. Пакет ориентирован на решение задач механики жидкости и газа для многокомпонентных и многофазных сред с учетом их возможного химического взаимодействия.

Приведенные в данной статье численные расчеты выполнены по заказу ООО «ЛУКОЙЛ-Пермь» в рамках научно-исследовательской работы по проектированию автоматизированной многофазной гидроструйной установки (УГС) на основе эжектора с изменяемыми геометрическими параметрами. Основные цели исследования – обеспечение устойчивой работы установки в различных режимах перекачки ГЖС и снижение энергозатрат за счет рационального выбора геометрических и расходных характеристик проточного тракта.

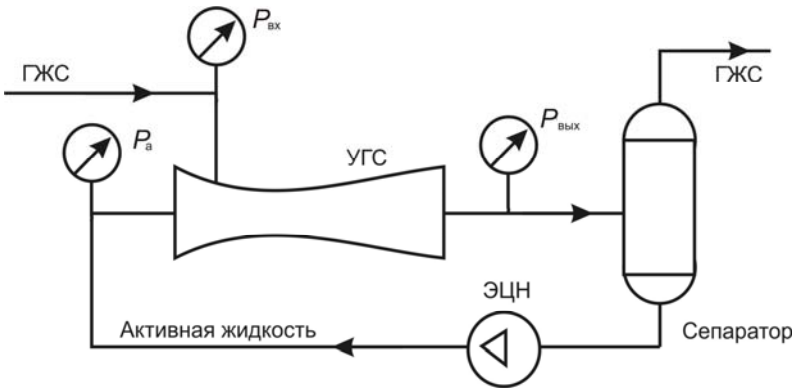


Рис. 1. Технологическая схема ДНС

Фрагмент технологической схемы дожимной насосной станции, в состав которой входит УГС, приведен на рис. 1. Схема включает собственно гидроструйную установку, шурфовой электроцентробежный насос (ЭЦН) с регулируемым электроприводом для подачи в УГС рабочей жидкости и сепаратор, обеспечивающий отделение ГЖС от рабочей жидкости. Нефтегазовая смесь (пассивная среда) подается в УГС при низком давлении $P_{вх}$, эжектируется рабочей жидкостью (активной средой) с повышением давления до $P_{вых}$ и после сепарации поступает в транспортный трубопровод. Шурфовой насос повышает давление активной жидкости до $P_а$.

Отличие проектируемой установки от традиционных струйных насосов заключается в схеме подачи активной и пассивной сред. В УГС используется кольцевое активное сопло с изменяемыми геометрическими параметрами. На рис. 2 выделена рабочая часть проточного тракта УГС, включающая кольцевое сопло, каналы для подачи активной и пассивной сред в камеру смешения. Активная жидкость подается через кольцевой канал *1*, образованный газовой трубкой *3* и подвижным сопловым наконечником *5*. Площадь проходного сечения сопла регулируется смещением соплового наконечника под действием пружины *6* в зависимости от активного давления, создаваемого шурфовым ЭЦН. С увеличением давления активной жидкости пружина сжимается и сопловой наконечник смещается вправо, что приводит к увеличению сечения сопла и расхода рабочей жидкости. На рисунке сопловой наконечник изображен в крайнем положении, соответствующем минимальному сечению. Газ поступает по неподвижной внутренней трубке *3*, жидкость – по внешнему каналу *2*. Причем такое распределение ГЖС по каналам достаточно условно, так как в рабочих режимах УГС по обоим каналам может подаваться газ, жидкость или ГЖС.

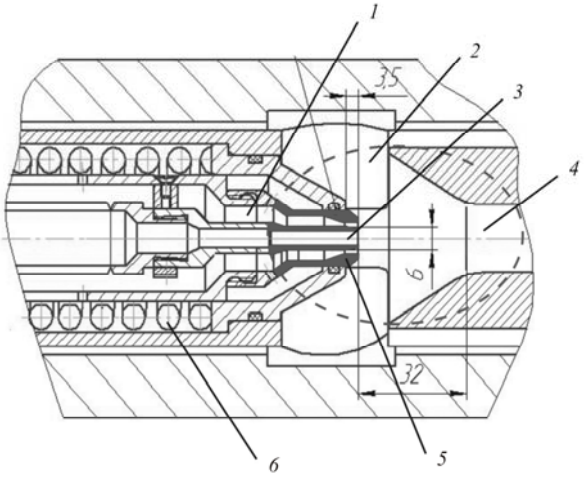


Рис. 2. Конструкция соплового узла УГС:
1 – канал подачи активной жидкости; *2* – канал подачи пассивной жидкости; *3* – канал подачи пассивного газа; *4* – камера смешения; *5* – подвижный сопловой наконечник; *6* – пружина

Расчетная схема задачи, используемая при моделировании течений в проточной части УГС, приведена на рис. 3. Она включает кольцевое сопло, каналы подачи пассивной и активной сред, камеру смешения и диффузор.

Рис. 3. Расчетная схема задачи

Поставленная задача реализована в рамках инженерного приложения CCM+ пакета STAR-CD (лицензия № 4901-10). Расчеты проводятся на основе прямого численного моделирова-

ния гидродинамики и теплообмена в реальной конструкции проточного тракта установки. Течение ГЖС в пределах расчетной области рассматривается как турбулентное, газовая фаза подчиняется законам идеального газа. Для оценки достоверности получаемых численных решений были проведены калибровочные расчеты. Сопоставление результатов с данными натурных испытаний УГС, проведенных на стенде ЗАО «НОВОМЕТ-Пермь» (активная жидкость – вода, пассивные среды – вода и воздух), убедительно свидетельствует об адекватности используемой компьютерной модели.

При расчетах в качестве граничных условий задавались значения давлений сред на входе в УГС и газожидкостной смеси на выходе из диффузора. Давление активной жидкости на входе в сопло варьировалось в соответствии с напорной характеристикой шурфового насоса при его работе на разных частотах регулирования. При решении задачи определялись поля скоростей, давлений, температур, а также расходы активной жидкости и компонентов пассивной среды. Проведено параметрическое исследование режимов работы УГС при варьировании профиля проходного сечения кольцевого канала подачи активной жидкости, геометрии камеры смешения и взаимной ориентации камеры смешения и активного сопла.

В качестве иллюстрации на рис. 4 представлены результаты расчета течений для различных режимов работы УГС при следующих исходных данных: $P_а = 100$ кг/см^2, $P_{вх} = 8$ кг/см^2, $P_{вых} = 20$ кг/см^2, диаметр камеры смешения 19 мм. По внутреннему каналу подается пассивный газ, а по наружному – жидкость. На рис. 4, *а*, *б*, *в* изображены траектории течения ГЖС в зоне инжекции и на входе в камеру смешения, а также указаны расчетные расходы активной жидкости и компонентов пассивной среды.

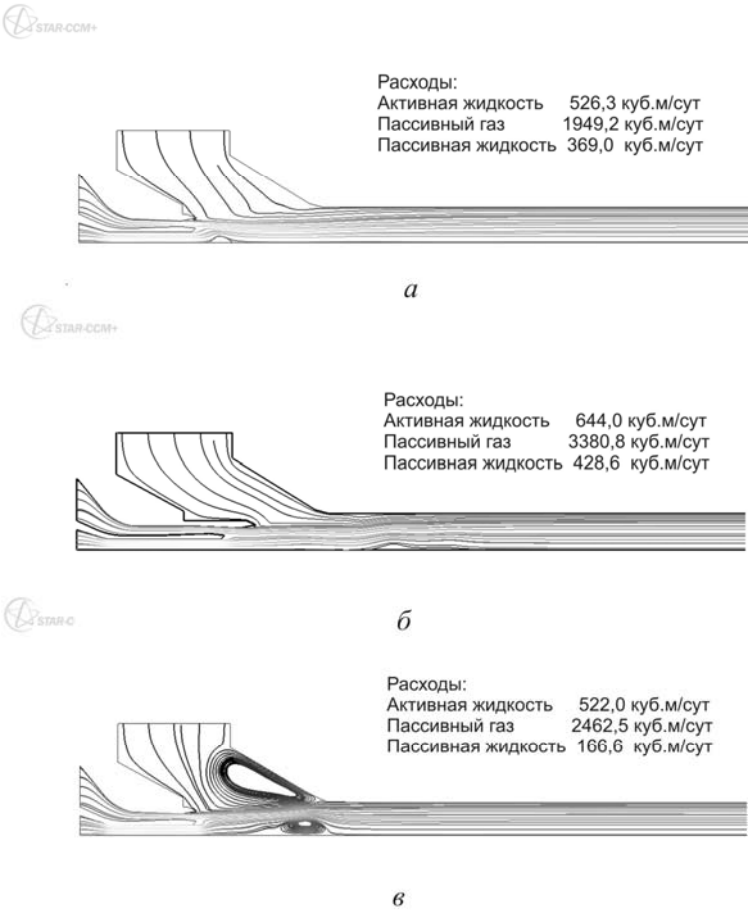


Рис. 4. Траектории течения ГЖС

Рис. 4, *а* и рис. 4, *в* соответствуют режиму, когда сопловой наконечник находится в крайнем левом по схеме (см. рис. 2) положении (симметричное сопло), а площадь проходного сечения активного сопла минимальна. В расчетном варианте на рис. 4, *а* установка обеспечивает перекачку ГЖС в номинальном режиме работы шурфового насоса. При увеличении давления $P_{вх}$ на вхо-

де пассивной среды шурфовой насос по заданному алгоритму поднимает давление $P_а$ и сопловой наконечник смещается вправо. На рис. 4, *б* сопловой наконечник находится в крайнем правом положении, а площадь сечения кольцевого сопла максимальна. В этом режиме возрастает объем перекачиваемой ГЖС, что приводит к снижению давления $P_{вх}$. Расчетами показано, что расходные характеристики проточного тракта УГС чрезвычайно чувствительны к изменению его геометрических параметров. Так, уменьшение диаметра камеры смешения всего на 1 мм существенно изменяет картину течения (см. рис. 4, *в*). На входе в камеру образуется устойчивый вихрь, который частично «запирает» канал пассивной жидкости, в результате чего ее расход уменьшается более чем в два раза. В то же время уменьшение притока жидкости через внешний канал способствует более интенсивной инжекции пассивного газа.

В реальных промысловых условиях перекачки ГЖС давление смеси $P_{вых}$ на выходе из диффузора изменяется в зависимости от различных факторов и может достигать больших значений. С целью проверки работоспособности инжектора на этих режимах были проведены расчеты при различных давлениях на выходе. На рис. 5 изображены кривые расходов пассивных сред, полученные при следующих условиях: шурфовой насос обеспечивает один и тот же перепад давлений, давление пассивной ГЖС неизменно и составляет 8 кг/см^2, работает симметричное активное сопло (как наиболее проблемный вариант). Как и следовало ожидать, с повышением давления на срезе диффузора расходы газа и жидкости уменьшаются, причем наиболее быстро падает расход жидкости. При давлении смеси на выходе порядка 30 бар инжектор еще работоспособен. Очевидно, это значение и является предельно достижимым в выбранных условиях. Дальнейшее повышение давления приводит к расчетной неустойчивости течения вплоть до образования возвратных течений в диффузоре и во входном сечении пассивной жидкости.

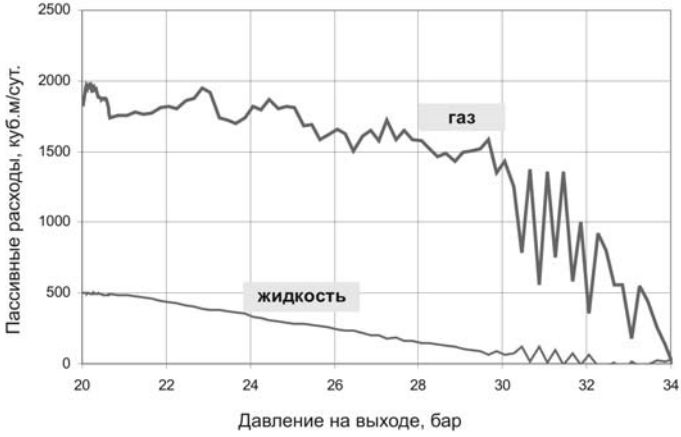


Рис. 5. Зависимость расходов пассивных сред от давления $P_{вых}$

На основании результатов численного моделирования выполнена оптимизация параметров проточного тракта гидроструйной установки, проведена оценка эффективности ее работы в различных условиях. Определены диапазон и алгоритм автоматического регулирования площади проходного сечения активного кольцевого сопла, обеспечивающие работу УГС в области максимальных КПД шурфового насоса.

К.С. Галягин, М.А. Ошивалов, Ю.А. Селянинов, Е.И. Вахрамеев

Пермский государственный технический университет

МОДЕЛИРОВАНИЕ ПРОЦЕССА ПАРОФАЗНОГО ОСАЖДЕНИЯ

Процесс парофазного осаждения примесей по технологии MCVD является одним из ключевых звеньев в технологической цепи производства высококачественных кварцевых световодов. На этапе подготовки преформ для вытяжки световолокна на внутренней поверхности опорной кварцевой трубки создаются условия для осаждения оксидов кремния и легирующих элементов (Ge, P), доставляемых в зону реакции в виде летучих хлоридных соединений.

При производстве активных световодов, способных усиливать проходящие по ним сигналы, поверхность преформы подвергается дополнительному легированию редкоземельными металлами (Yb, Er и т.д.). Соединения редкоземельных металлов не обладают летучестью, поэтому для транспортировки их в зону поверхностного осаждения используют органометаллические соединения – хелаты, содержащие ионы соответствующих легирующих компонентов. Пары хелатов имеют достаточно узкий температурный диапазон «живучести»: при низких температурах они конденсируются, при высоких – распадаются. Поэтому их подача должна производиться в строго регламентированном диапазоне температур, несовместимом с условиями транспорта легирующих хлоридов.

Процесс поверхностного осаждения представляет собой весьма ответственный этап производства световода, так как толщина и компонентный состав осажденного на преформе слоя практически полностью определяют параметры светопроводящей жилы будущего световолокна. Это в полной мере обуславливает актуальность численного анализа процесса осаждения, который в рамках данной работы реализован средствами инженерного приложения CCM+ программного продукта STAR-CD компании CD-ADAPCO (лицензия № 4901-10).

На рис. 1 приведены принципиальная схема технологического процесса хлоридно-хелатного легирования (*а*) и принятая для численного моделирования схема расчетной области (*б*). Опорная кварцевая труба наружным радиусом R_0 и внутренним R_1 вращается вокруг оси с постоянной угловой скоростью Ω. Кольцевой подвод теплоты по наружной поверхности трубы моделирует ее локальный нагрев водородной горелкой, перемещающейся вдоль оси со скоростью $W_{\text{гор}}$. В левой части трубы расположен коаксиальный канал транспорта хелатов, перемещение которого синхронизировано с горелкой, т.е. расстояние z_x остается постоянным в ходе всего процесса. Канал (питатель) представляет собой термостатированную тонкостенную стеклянную трубку с внутренним радиусом R_2 и толщиной δ. Подача

исходных газообразных хлоридов в зону реакции производится по кольцевому зазору, образованному внутренней поверхностью опорной трубы и питателем для транспорта хелатов.

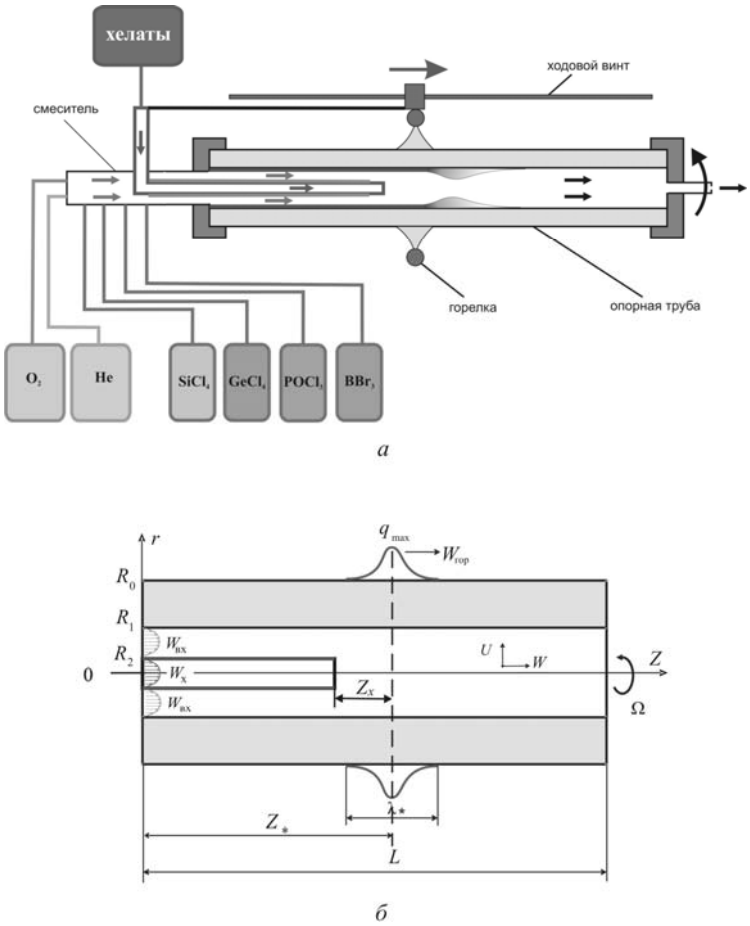


Рис. 1. Схема технологического процесса (*а*) и расчетная схема задачи тепломассообмена (*б*)

Суть процесса осаждения легированного слоя кварца на внутренней поверхности преформы заключается в следующем.

При движении газового потока, содержащего парообразные хлориды Si, Ge, P, его температура повышается, достигая максимального значения в зоне горелки, где происходит реакция окисления с образованием оксидов соответствующих элементов. При попадании хелатов в высокотемпературную зону органика выгорает с образованием оксидов редкоземельных металлов. Далее происходит коагуляция образовавшихся оксидов и осаждение их на холодной поверхности опорной трубки перед горелкой под действием сил термофореза.

Численное моделирование процесса поверхностного легирования преформы путем парофазного осаждения осуществляется в два этапа: на первом этапе для сечения опорной трубы и потока газовой среды внутри неё определяются поля температур и скоростей движения газового потока, на втором – исследуется процесс осаждения путем регистрации траекторий частиц заданного размера, инжектированных в высокотемпературной зоне газового потока в поле действия термофоретических сил. Вопросы протекания химических преобразований при окислении хлоридов, выгорания хелатов и коагуляция легирующих частиц в рамках данной работы не рассматривались.

При решении задачи тепломассопереноса в расчетной области ставятся граничные условия, адекватно описывающие натурный объект моделирования с учетом зависимости теплофизических характеристик от температуры. На входе газовых потоков задаются профили скоростей. Тепловое воздействие газовой горелки моделируется движущимся кольцевым поверхностно распределенным источником тепла. Учитываются как диффузионный, так и радиационно-конвективный механизмы переноса теплоты.

Детальное описание изучаемых явлений приводит к сложной и громоздкой схеме дискретизации расчетной области (сетка порядка $3 \cdot 10^6$ ячеек), что в рамках решения связанной нестационарной задачи тепломассообмена требует значительных вычислительных ресурсов ЭВМ. В частности, моделирование на

двухпроцессорной системе одного прохода горелки в пределах расчетной области занимает порядка двухсот часов машинного времени. Поэтому была проведена серия исследований с целью оптимизации ресурсных затрат.

Исходная объемная постановка задачи с вращением сведена к плоской осесимметричной схеме с равномерно распределенным по окружности потоком тепла от горелки.

Были проведены предварительные калибровочные расчеты для выбора рациональных параметров пространственной и временной дискретизации. В результате число ячеек сетки удалось снизить до 20000, а шаг по времени увеличить до 0,05 с. Размер ячейки принятой базовой сетки составляет 0,3×1,8 мм. Кроме этого, для повышения точности численных решений выбраны адаптивные сетки со сгущением в явно выделенном пограничном призматическом слое, позволяющие получать достоверные решения в области максимального значения градиентов искомых функций.

Для дискретизации применялась итерационная неявная схема второго порядка точности с нижней релаксацией по схеме Гаусса-Зейделя. Коэффициенты релаксации составляют: 0,7 для скорости, 0,3 для давления и 0,9 для энергии. При таких параметрах время счета одного прохода составляет 12 часов на однопроцессорной вычислительной системе. Таким образом, удалось достигнуть вполне разумного компромисса между точностью решения и потребляемыми вычислительными ресурсами.

Ниже приведены некоторые основные результаты расчетного анализа. К сожалению, формат черно-белого представления не позволяет наглядно проиллюстрировать тепловое и аэродинамическое состояние расчетной области в виде полевых изокартин, поэтому ограничимся графиками распределения температур граничных поверхностей по длине опорной трубки, приведенными на рис. 2.

Видно, что наибольший разогрев опорной трубки наблюдается в зоне непосредственного действия газовой горелки (на графике она показана вертикальным пунктиром в центральной части). Учитывая, что скорость движения газового потока много

больше скорости перемещения газовой горелки, отметим принципиальное изменение направления температурных градиентов в газовом потоке, определяющих силы термофореза. Если после горелки (левая часть графика) частицы, двигающиеся в потоке газа отталкиваются более горячей поверхностью трубки к оси потока и испытывают торможение в осевом направлении, то перед горелкой (правая часть области) создаются условия для их осаждения на более холодной поверхности кварцевой трубки и ускорения движения вдоль канала.

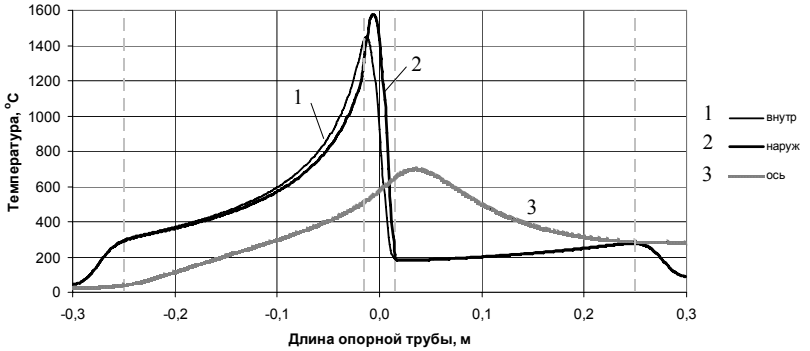


Рис. 2. Профили температурного состояния системы

Более детально картины распределения компонентов вектора градиента температуры газового потока в зоне горелки иллюстрирует рис. 3.

Неоднородность распределения температурных градиентов порождает специфику поля термофоретических сил, которые кроме величины вектора градиента температуры определяются еще и размерами самих частиц. В качестве первого приближения величину силы термофореза можно оценить на основании термодинамических представлений с использованием уравнения состояния идеального газа в следующем виде:

$$F_T = \Delta P \cdot \delta^2 = -\rho R\, \delta^3 \cdot grad\, T\,,$$

где R – газовая постоянная, ρ – плотность газа, δ – размер частиц, T – поле температур газового потока.

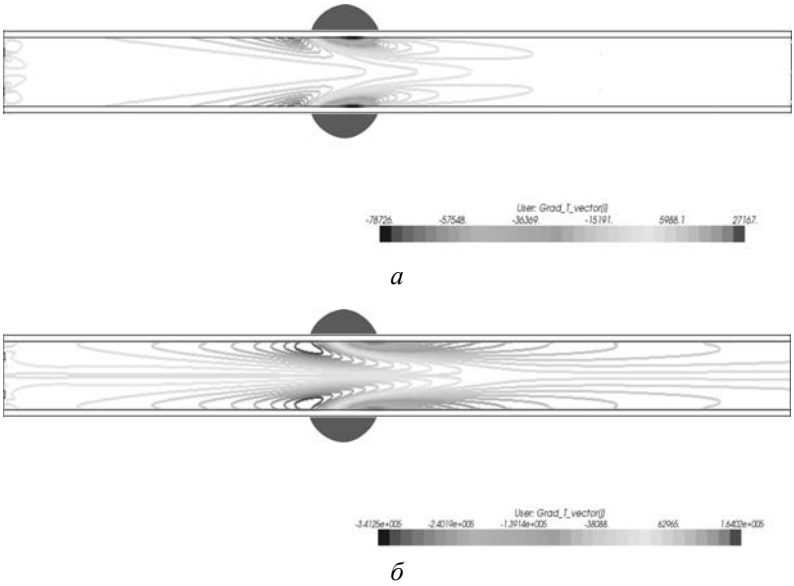


Рис. 3. Поля осевых (*а*) и радиальных (*б*) градиентов температуры

Для анализа условий осаждения легирующих примесей был применен прием инжектирования частиц, позволяющий визуализировать процесс осаждения. В интересующей области газового потока размещается регулярная система точечно-кольцевых инжекторов, впрыскивающих частицы одинакового фиксированного размера, траектории которых регистрируются в агранжевых координатах вплоть до осаждения на поверхности трубки (рис. 4). Часть частиц при этом, избежав осаждения, покидают технологическую зону вместе с потоком газа.

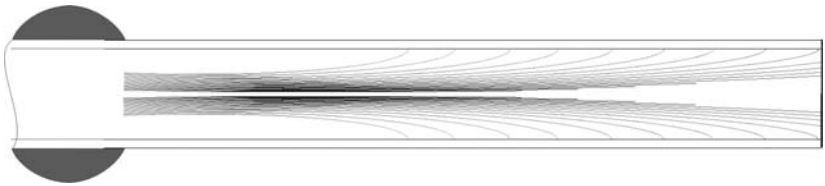


Рис. 4. Треки осаждаемых частиц

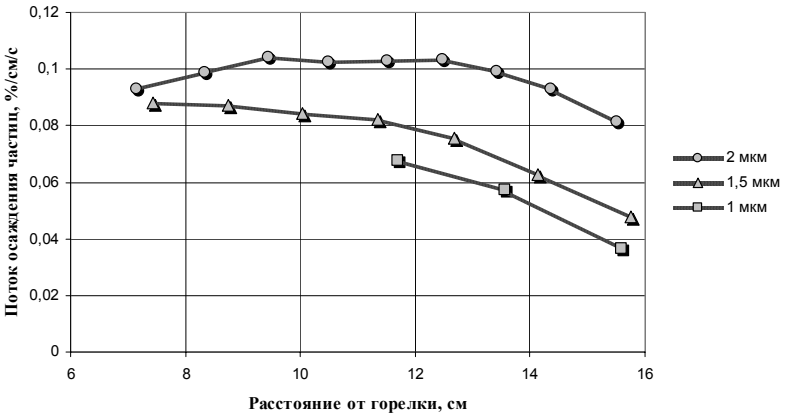


Рис. 5. Распределение частиц по длине зоны осаждения

В результате применения специально разработанной процедуры обработки картины траекторий осаждающихся частиц различных размеров определяются расходные характеристики процесса в виде погонной плотности массового потока осаждения в долях от концентрации частиц в зоне размещения инжекторов (рис. 5).

Применение этой процедуры для частиц каждого оксида, участвующего в легировании, в сочетании с интегрированием по спектру размеров частиц позволяет оценить общую толщину и покомпонентный состав осажденного на поверхности опорной трубки слоя.

А.В. Гергель

Нижегородский государственный университет им Н.И. Лобачевского

ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ С АДАПТИВНЫМИ РЕШАЮЩИМИ ПРАВИЛАМИ

Многоэкстремальные задачи, методы решения которых рассматриваются в данной статье, широко встречаются в приложениях. Усложнение математических моделей для более адекватного описания изучаемых объектов, явлений и систем приводит

к резкому увеличению трудоемкости анализа их математических моделей. Это повышение сложности приводит к необходимости целенаправленного выбора вариантов в процессе поиска оптимального решения и построения неравномерных покрытий области поиска, являющихся плотными только в окрестности искомого глобального оптимума [1–6]. Возможность построения неравномерных покрытий обычно обеспечивается предположением о выполнимости условия Липшица. При этом поведение функционалов задачи многоэкстремальной оптимизации часто является неоднородным в разных подобластях областях области поиска. В некоторых подобластях значения функционалов модели может изменяться достаточно быстро (что будет соответствовать большим значениям константы Липшица в таких подобластях), в других подобластях – например, в окрестности точек экстремумов функций – значения функционалов могут изменяться более плавно. Как результат, построение интегральных (единых) оценок характеристик оптимизируемых функций для всей области поиска может не соответствовать полностью параметрам функционалов в разных подобластях области поиска.

Задачи многоэкстремальной оптимизации и адаптивная оценка констант Липшица. Рассматривается одномерная безусловная задача многоэкстремальной оптимизации

$$\varphi(x^*) = \min\{\varphi(x): x \in [a, b]\}, \qquad (1)$$

для которой минимизируемая функция $\varphi(x)$ и ее первая производная удовлетворяют условию Липшица с константами L и $L1$ соответственно. Одним из способов учета разнородного поведения оптимизируемых функций в разных областях поиска состоит в построении отдельных оценок констант Липшица оптимизируемых функций для разных подобластей области поиска. Целесообразность такого направления работ достаточно очевидна – так, в окрестности точек экстремумов гладкой оптимизируемой функции значения производных должны быть близки к нулю, определяя тем самым невысокое значение константы Липшица в таких подобластях.

Проведение исследований в данном направлении были инициированы в работе [1]; предложенный в работе подход рассматривается в п. 2. В пп. 3–5 предлагаются новые способы построения отдельных оценок констант Липшица оптимизируемых функций для разных подобластей области поиска.

Локальная настройка оценок константы Липшица. Суть предложенного в [1] подхода состоит в следующем. Для построения решающих правил алгоритмов глобального поиска предлагается использовать не единую для всей области поиска оценку константы Липшица, а строить такие оценки для каждого поискового интервала в отдельности. Поскольку получение таких оценок может происходить с достаточно большими погрешностями, то при нахождении локальных (интервальных) оценок должна учитываться и общая (глобальная) оценка константы Липшица.

Приведем в соответствии с [2] точное алгоритмическое описание предлагаемого подхода.

Вычислительная схема оценки локальных констант Липшица. Предположим, что алгоритм глобального поиска, в котором используются оценки константы Липшица, выполнил $k \geq 1$ итераций поиска и получена поисковая информация

$$a = x_0 < x_1 < \ldots < x_i < \ldots < x_k = b$$

(нижний индекс соответствует упорядоченному представлению точек испытаний).

Оценка константы Липшица M_i интервала (x_{i-1}, x_i), $1 \leq i \leq k$, вычисляется в соответствии со следующими выражениями:

$$m_i = rM_i,\ M_i = \max\{\mu'_i,\ \mu''_i,\ \mu_0\}, \tag{2}$$

где

$$\mu'_i = \max\{|z_j - z_{j-1}|/(x_j - x_{j-1}) : j = i-1, i, i+1\}, \tag{3}$$

$$\mu''_i = M(x_i - x_{i-1})/d\max, \tag{4}$$

$$M = \max_{1 \le i \le k} \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}}, \tag{5}$$

$$d\max = \max_{1 \le i \le k} (x_i - x_{i-1}). \tag{6}$$

Когда $i = 1$ или $i = k$, для вычисления μ'_i из (3) используются только $j = i$, $i+1$ или $j = i-1$, i соответственно.

Поясним приведенные соотношения. Величина M является оценкой глобальной константы Липшица L на всем интервале поиска $[a,b]$. Величины M_i, $1 \le i \le k$, являются оценками локальных констант Липшица L_i на интервалах (x_{i-1},x_i), $1 \le i \le k$. Каждая такая оценка строится на основе трех величин: μ'_i, которая отвечает за локальные свойства функции $\varphi(x)$ на интервале $[x_{i-1},x_i]$; μ''_i, которая следит за глобальными свойствами функции $\varphi(x)$ на всем интервале поиска $[a,b]$; и параметра μ_0. Когда интервал $[x_{i-1},x_i]$ велик, роль глобальной информации повышается, так как в этом случае локальная информация может быть ненадежной. Когда интервал $[x_{i-1},x_i]$ мал, роль глобальной информации понижается, так как в этом случае большее значение имеет локальная информация, а влияние глобальной информации ослабляется. Наличие параметра μ_0 отражает предположение о том, что функция $\varphi(x)$ не является константой на интервале $[a,b]$. Параметр μ_0 также контролирует чувствительность алгоритма. В случае, когда $L \le \mu_0$, алгоритм теряет чувствительность к изменениям локальной информации и функционирует как метод полного перебора.

Представленная схема локальной настройки сохраняет свойства сходимости алгоритма глобального поиска, к которому она применяется [2].

Для демонстрации эффективности рассмотренного подхода приведем результаты вычислительного эксперимента [2], в котором в качестве контрольного множества оптимизационных задач использован широко применяемый в научной литературе набор тестовых многоэкстремальных функций. При прове-

дении эксперимента использовались следующие алгоритмы: Галперина (АГ), алгоритм Пиявского (АП) и алгоритм глобального поиска Стронгина (АГП). Сравнение выполнялось двумя методами, использующими локальную настройку – это метод ломаных (МЛ-ЛН), построенный на основе метода Пиявского, алгоритм глобального поиска Стронгина с локальной настройкой (АГМ-ЛН). Алгоритм глобального поиска Стронгина с локальной настройкой в соответствии с новой предложенной аддитивной схемой свертки обозначен в таблице как АГП-ЛНА

Результаты вычислительных экспериментов для одномерных методов глобальной оптимизации, использующих схему локальной настройки (МЛ-ЛН и АГП-ЛН, АГП-ЛНА)

Задача	Методы глобального поиска					
	АГ	АП	АГП	МЛ-ЛН	АГП-ЛН	АГП-ЛНА
1	377	149	127	37	35	29
2	308	155	135	36	36	27
3	581	195	224	145	136	42
4	923	413	379	45	41	37
5	326	151	126	46	45	31
6	263	129	112	84	54	36
7	383	153	115	41	39	27
8	530	185	188	126	132	42
9	314	119	125	44	42	33
10	416	203	157	43	40	29
11	779	373	405	74	72	56
12	746	327	271	71	66	51
13	1829	993	472	73	45	40
14	290	145	108	43	50	27
15	1613	629	471	62	63	40
16	992	497	557	79	53	45
17	1412	549	470	100	101	81
18	620	303	243	44	41	34
19	302	131	117	39	34	26
20	1412	493	81	70	42	29
В среднем	720,8	314,6	244,15	65,1	58,35	38,05

Результаты экспериментов сведены в таблицу. По каждому методу в таблице приведено количество испытаний, выполняемое алгоритмом до выполнения условия остановки. В последней строке таблицы содержится среднее число испытаний по всему набору тестовых задач для каждого метода в отдельности.

Точность поиска в расчетов принималась $\varepsilon=10^{-6}(b-a)$. Для алгоритмов Галперина и Пиявского использовалась точная оценка константы Липшица. Параметры остальных алгоритмов были равны 2 (для АГП и АГП-ЛН) и 1,1 для МЛ-ЛН. Для методов МЛ-ЛН и АГП-ЛН параметр μ_0 устанавливался равным 10^{-6}.

Следует подчеркнуть, что схема локальной настройки может быть применена для построения самых разных одномерных и многомерных алгоритмов решения задач липшицевой глобальной оптимизации. Она может быть также обобщена, например, на случай задачи с целевой функцией, удовлетворяющей условию Гельдера или на случай решения задачи глобальной оптимизации с многоэкстремальными липшицевыми ограничениями [2].

Локальная настройка с аддитивной сверткой оценок константы Липшица. Внимательно проанализировав схему локальной настройки, можно сделать важных замечания.

Замечание 1. Характер зависимости (4) влияния оценки глобальной константы Липшица для интервала (x_{i-1},x_i), $1\le i\le k$, позволяет сделать вывод, что при построении данного соотношения явно или неявно делалось предположение о квадратичном поведении оптимизируемой функции в окрестности глобального минимума. На самом деле, производная для квадратичной функции имеет вид линейной зависимости и для определения значения производной на одном интервале при известном ее значении на другом интервале необходимо разделить известное значение производной на длину интервала, на котором задано значение, и умножить на длину искомого интервала. Или аналитически – пусть минимизируемая функция имеет вид

$$\varphi(x)=ax^2.$$

Тогда производная этой функции имеет вид

$$\varphi'(x)=2ax.$$

Оценка константы Липшица для интервалов $(x_1=0, x_2)$ и $(x'=0, x'')$, $x''< x_2$ имеет вид

$$M_1=\frac{|z_2-z_1|}{x_2-x_1}=ax_2,\quad M'=\frac{|z''-z'|}{x''-x'}=ax''$$

соответственно. Но тогда оценку M' можно получить через значение оценки M_1, т.е.

$$M'=[M_1/(x_2-x_1)](x''-x')=(M_1/x_2)\,x'',$$

что и соответствует правилу (4), схеме локальной настройки. Отсюда следует два возможных вывода:

1) схема локальной настройки наиболее эффективно будет действовать для многоэкстремальных оптимизационных задач, в которых поведение минимизируемых функций в окрестности глобального минимума является квадратичным;

2) схема локальной настройки может быть построена и для многоэкстремальных оптимизационных задач, в которых поведение минимизируемых функций в окрестности глобального минимума не является квадратичным, заменив для этого только соотношение (4) – так, для кубического поведения функций в окрестности глобального минимума соотношение должно иметь вид

$$\mu_i''=M(x_i-x_{i-1})^2/(d\max)^2.$$

Замечание 2. Локальная (3) и глобальная (4) составляющие интервальной оценки константы Липшица могут рассматриваться как два разных противоречивых критерия при оценке некоторого компромиссного значения константы Липшица. Предпочтение одного критерия (например, локальной составляющей) приводит к более быстрому завершению работы алгоритма глобального поиска, однако здесь возможна потеря решения из-за оценки константы Липшица с недостатком. Предпочтение другого критерия (глобальной составляющей) может привести к более продолжительной работе алгоритма глобаль-

ного поиска из-за оценки константы Липшица с избытком. Как результат, для получения некоторого компромиссного (промежуточного) значения константы Липшица необходима та или иная свертка данных критериев. Так, в схеме локальной настройки данная свертка близка к широко известной минимаксной свертке критериев. Возможны и иные способы сверток.

Локальная настройка с аддитивной сверткой оценок константы Липшица. В предлагаемой новой схеме локальной настройки оценка интервальных оценок константы Липшица осуществляется в соответствии с аддитивной сверткой

$$m_i = rM_i,\ M_i = \max\ \{\ \mu'_i + \mu''_i,\ \mu_0\ \}, \qquad (7)$$

где величины μ'_i, μ''_i, μ_0 определяются соотношениями (3)–(6).

Результаты вычислительных экспериментов с использованием локальной настройки с аддитивной сверткой оценок константы Липшица представлены в таблице (столбец АГП-ЛНА).

Кроме аддитивной свертки (7) разработаны еще несколько дополнительных схем построения оценок констант Липшица оптимизируемых функций для разных подобластей области поиска:

локальная настройка с интервальной схемой построения оценок константы Липшица;

локальная настройка с выделением подобластей с близкими значениями константы Липшица.

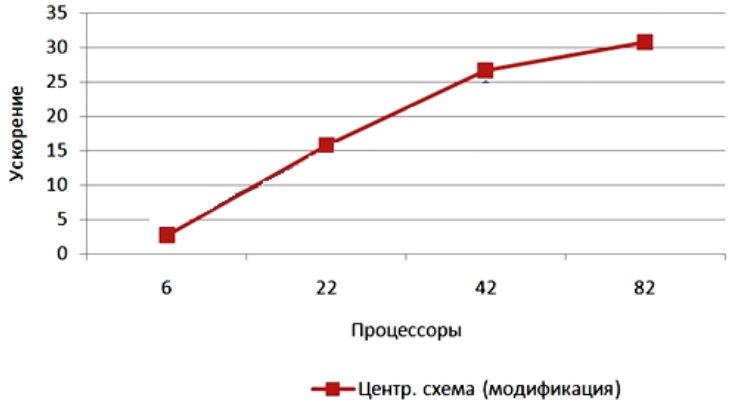


Рис. Результаты вычислительного эксперимента

Для организации параллельных вычислений использовалась централизованная схема. На рисунке представлены результаты вычислительного эксперимента параллельного метода многоэкстремальной оптимизации с адаптивными решающими правилами.

Список литературы

1. Сергеев Я.Д. Одномерный детерминированный алгоритм глобального поиска // Вычисл. матем. и матем. физ. – 1995. – Т. 35, № 5. – С. 705–717.

2. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. – М.: Физматлит, 2008.

3. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. – М.: Наука, 1978.

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

5. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. – Н. Новгород: Изд-во ННГУ, 2007.

6. Гергель В.П. Об одном способе учета значений производных при минимизации многоэкстремальных функций // Вычисл. матем. и матем. физ. – 1996. – Т. 36, № 6. – С. 51–67.

А.В. Гергель, Д.В. Гнатюк

Нижегородский государственный университет им Н.И. Лобачевского

СРАВНЕНИЕ АДАПТИВНЫХ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ МНОГОМЕРНОЙ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ

Сложность решения многомерных многоэкстремальных задач оптимизации обусловлена несколькими причинами. В первую очередь стоит отметить экспоненциальный рост объема вычислений при увеличении размерности (числа варьируе-

мых параметров). Во-вторых, вычисление значений функционалов оптимизационной задачи требует большого количества вычислений. Именно на такие вычислительно-трудоемкие задачи ориентирована разработка параллельных методов многомерной многоэкстремальной оптимизации в данной работе.

Задачи многомерной многоэкстремальной оптимизации и адаптивная многошаговая схема редукции размерности. Задача многомерной многоэкстремальной оптимизации может быть определена как проблема поиска наименьшего значения действительной функции $\varphi(y)$

$$\varphi(y^*) = \min\{\varphi(y): y \in D\}, \quad (1)$$

где D – область поиска, представляющая собой некоторый гиперпараллелепипед N-мерного евклидова пространства.

Многие регулярные поисковые методы решения многомерных оптимизационных задач сводят многомерную задачу (явно или неявно) к системе одномерных подзадач. Один из наиболее общих методов редукции размерности состоит в применении *многошаговой схемы редукции размерности*, согласно которой решение многомерной задачи оптимизации может быть получено посредством решения последовательности «вложенных» одномерных задач [1–3]:

$$\min_{y \in D} \varphi(y) = \min_{y_1 \in [a_1, b_1]} \ldots \min_{y_N \in [a_N, b_N]} \varphi(y_1, \ldots y_N). \quad (2)$$

Использование этой схемы позволяет получить общий способ для применения одномерных алгоритмов оптимизации к решению многомерных оптимизационных задач. Согласно (2) решение многомерной задачи (1) сводится к решению одномерной задачи

$$\varphi^* = \min_{y \in D} \varphi(y) = \min_{y_1 \in [a_1, b_1]} \tilde{\varphi}_1(y_1), \quad (3)$$

где

$$\tilde{\varphi}_i(y_i) = \varphi_i(y_1, \ldots, y_i) = \min_{y_{i+1} \in [a_{i+1}, b_{i+1}]} \varphi_{i+1}(y_1, \ldots, y_i, y_{i+1}) \; 1 \le i < N, \quad (4)$$

$$\varphi_N(y_1,...,y_N) = \varphi(y_1,...,y_N). \quad (5)$$

Правила (3)–(5) определяют множество задач

$$F_l = \{f_i(y_i, x),\ 1 \le i \le l\}, \quad (6)$$

порождаемых в соответствии с многошаговой схемой редукции. Количество задач в множестве F_l в процессе поиска может изменяться: увеличиваться при переходе к следующей переменной и уменьшаться при завершении решения какой-либо из задач (можно отметить, что при этом количество задач не превышает размерность решаемой задачи N). При этом активной – решаемой – в множестве F_l является только одна задача – это задача с максимальным номером варьируемой переменной.

В данной статье рассматривается обобщенная (адаптивная) многошаговая схема редукции размерности, в которой предлагается осуществлять одновременное решение всех задач множества F_l.

Такое решение может выполняться последовательно (при наличии единственного процессора), тогда для выполнения очередной итерации глобального поиска необходимо выбирать для решения одну из задач множества F_l в соответствии с тем или иным правилом выбора задач. Но решение задач может выполняться и параллельно, если используемый вычислитель является многопроцессорным или многоядерным. Именно параллельный – более общий вариант – будет рассматриваться далее в работе.

Важно отметить, что выполнение итерации глобального поиска для любой задачи множества F_l с номером переменной меньшим, чем N (N есть размерность решаемой задачи), будет приводить к порождению новой одномерной задачи вида (4) и, таким образом, количество задач в семействе F_l может оказаться значительным (десятки и сотни тысяч для сложных задач оптимизации).

Централизованная схема параллельного глобального поиска. Характеристики интервалов, вычисляемые алгоритмами глобального поиска, рассматриваются как некоторая мера важности интервалов на предмет содержания в них искомого решения оптимизационной задачи. Следуя данному пониманию,

в параллельных методах после выбора точки проведения испытания для первого процессора в точном соответствии с последовательным алгоритмом для второго процессора точку испытания выбирают из следующего по важности интервала (т.е. из интервала со следующей по порядку максимальной характеристикой) и т.д.

Рассмотренная выше идея послужила основой для централизованной схемы параллельного глобального поиска. Пусть $p > 1$ есть число процессоров, используемых для решения задачи оптимизации. Каждый из процессоров, получив точку $y \hat{I} D$, осуществляет независимо и параллельно с другими процессорами вычисление значения функции $j(y)$ из (1). В процессе вычисления значения функции испытание считается незавершенным, таким образом, при выполнении алгоритма будут существовать точки завершенных и незавершенных испытаний. Центральный процессор осуществляет выбор очередной итерации, который заключается в поиске интервала с максимальной характеристикой из списка допустимых интервалов, и вычисление по некоторому правилу точки нового испытания.

Результаты выполненных численных экспериментов для оценки эффективности централизованной схемы подтвердили следующий факт: при превышении определенного числа процессоров-исполнителей центральный процессор не обеспечивает нагрузку всех имеющихся процессоров-исполнителей. Таким образом, дальнейшее повышение числа процессоров выше некоторого порога не приводит к желаемому росту ускорения.

Модификация централизованной схемы параллельного глобального поиска. Идея модификации централизованной схемы основана на устранении «узкого» места оригинальной схемы. Это достигается за счет увеличения числа процессоров, выполняющих роль центрального. В роли менеджера, выдающего задания процессорам-исполнителям, в модифицированной схеме выступает не один, а k процессоров. При условии сохранения единства поисковой информации между этими k процессорами выбор очередной итерации каждым из них может осуществляться следующим образом:

• обозначим множество центральных процессоров $P = \{ p_1, p_2, \ldots, p_k \}$;

• выберем k интервалов с максимальными характеристиками $L = \{ l_1, l_2, \ldots, l_k \}$;

• тогда точка очередного испытания для процесса p_i выбирается в интервале l_i.

Для обеспечения единства поисковой информации для всех центральных процессоров:

• перед каждой новой итерацией производится синхронизация очередей характеристик каждого из процессов;

• процессор-исполнитель высылает результаты своих вычислений каждому из центральных процессоров.

Работу модифицированной централизованной схемы можно схематично представить в следующем виде (рис. 1).

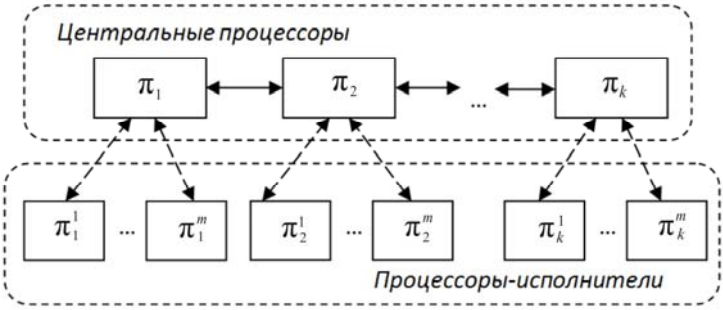


Рис. 1. Схема работы модифицированной централизованной схемы

Таким образом, данная модификация централизованной схемы позволяет достаточно хорошо масштабировать алгоритм параллельного глобального поиска на большое число процессоров.

Результаты вычислительных экспериментов. На рис. 2, 3 приведены результаты выполненных численных экспериментов для оценки эффективности предложенной модификации централизованной схемы параллельного глобального поиска для адаптивной многошаговой схемы редукции размерности.

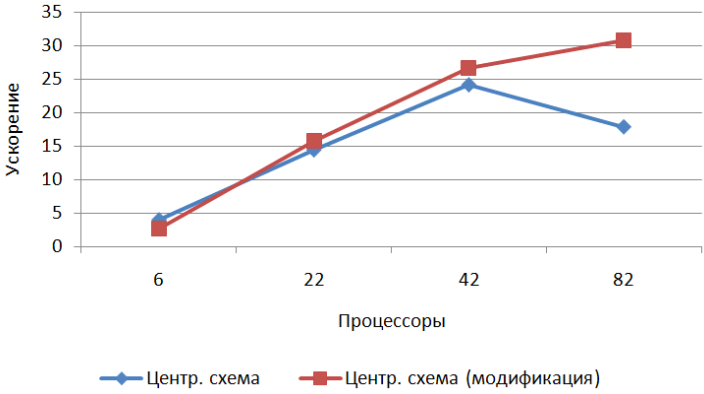


Рис. 2 Оценка эффективности модификаций централизованной схемы для адаптивной многошаговой схемы редукции размерности на классе задач Растригина

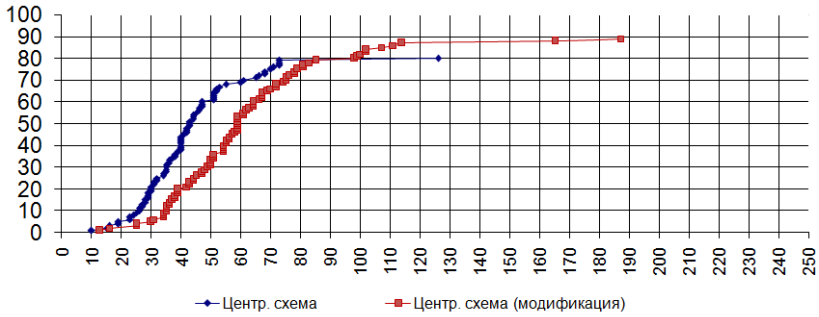


Рис. 3 Оценка эффективности модификаций централизованной схемы для адаптивной многошаговой схемы редукции размерности на классе задач Гришагина

В первом эксперименте (см. рис. 2) был использован класс тестовых задач многоэкстремальной оптимизации Гришагина 6-мерной размерности, во втором эксперименте (см. рис. 3) были построены операционные характеристики для класса тестовых задач Гришагина. Результаты экспериментов подтверждают хорошую масштабируемость модификации централизованной схемы, однако для задач малых размерностей оригинальная схема показывает лучшие

результаты. Это обусловлено тем, что для таких задач эффект «узкого» места оригинальной схемы не достигается.

Список литературы

1. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. – М.: Наука, 1978.

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

3. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2007.

4. Sergeyev Y.D., Grishagin V.A. Parallel asynchronous global search and the nested optimization scheme. // J. Comput. Anal. Appl. – 2001. – Vol. 3, No. 2. – P. 123–145.

5. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007.

6. Гергель А.В., Гнатюк Д.В. Адаптивные параллельные алгоритмы для многомерной многоэкстремальной оптимизации // Высокопроизводительные параллельные вычисления на кластерных системах: материалы конф. – Н. Новогород, 2009. С. 92–96.

В.П. Гергель, Д.В. Ахматнуров, А.В. Калачев, М.С. Царев

Нижегородский государственный университет им. Н.И. Лобачевского

УЧЕБНО-ИССЛЕДОВАТЕЛЬСКАЯ БИБЛИОТЕКА ПАРАЛЛЕЛЬНЫХ МЕТОДОВ ДЛЯ СИСТЕМ С РАЗДЕЛЕННОЙ ОБЩЕЙ ПАМЯТЬЮ

На протяжении многих лет одной из наиболее острых проблем эффективного использования потенциала компьютерной техники является трудоемкость разработки программного обеспечения. Сложность разработки программного обеспечения

связана с большим разнообразием компьютерных систем и их непрерывным совершенствованием. При появлении новых вычислительных систем часто приходится разрабатывать заново большое количество алгоритмов. К сожалению, в отличие от компьютерного оборудования технологии разработки программ совершенствуются существенно медленнее. При разработке параллельных программ следует учитывать особые типы возможных ошибок, такие как тупики и гонки данных. Данные типы ошибок часто требуют больших временных затрат на выявление и отладку. Следовательно, для параллельных вычислительных систем проблема трудоемкости разработки программ особенно критична. Одним из выходов из столь сложной ситуации может быть создание языков параллельного программирования.

В настоящее время для разработки параллельных программ наиболее распространено использование технологий OpenMP и MPI. Позволяя в целом создавать высокоэффективные параллельные программы, данные технологии, являются средствами низкоуровневой разработки – фактически, эти технологии являются «ассемблерами» параллельного программирования. В связи с этим многими исследователями предпринимаются попытки создания новых – параллельных – языков программирования, которые помогли бы снизить трудоемкость разработки параллельных программ. В этом направлении представляет интерес модель вычислительных систем с разделенной глобальной адресуемой памятью (*partitionedglobaladdress*space, PGAS). В рамках данной модели память многопроцессорной системы образуется из памяти отдельных процессоров и является глобально-адресуемой.

В работе представлен ряд языков параллельного программирования, созданных на основе модели PGAS. К данной группе языков относятся Chapel и X10, разработанные в рамках программы HPCS (High Productivity Computing Systems), и язык Co-Array Fortran (CAF). Далее дается краткая характеристика этих языков, рассматривается разработанная библиотека параллельных методов ParaLibPGAS и приводятся результаты экспериментов.

Общая характеристика исследуемых языков программирования. Chapel – это новый язык параллельного программирования, разрабатываемый компанией Cray (http://chapel.cray.com). Основу модели вычислительной системы в языке представляет понятие *locale*, являющееся абстракцией вычислительного элемента системы. Locale содержит обрабатывающее устройство и память. Задачи, исполняемые на locale, имеют однородный доступ к локальной памяти, доступ к памяти других locale происходит более медленно.

X10 разрабатывается при поддержке компании IBM и находится сейчас в стадии активного развития (http://x10-lang.org). Синтаксис языка X10 создан на основе языка программирования Java. Для представления отдельного вычислительного узла высокопроизводительной системы в X10 используется понятие *place*. Place может рассматриваться как абстракция мультипроцессора с общей памятью (SMP-узла). Выполняемая последовательность действий в X10-программе определяется как активность (*activity*), выполняемая исполнителем. На одном исполнителе могут выполняться одновременно несколько активностей.

Co-Array Fotran(CAF) – расширение языка Fortran 95/2003 для параллельного программирования. CAF разработан Robert Numrich и John Reid в 1990-х годах (http://www.co-array.org). В качестве основной задачи, поставленной при разработке языка, было внесение минимально возможных расширений Fortran, достаточных для организации параллельных вычислений. В основе подхода схема SPMD (Single-Program-Multiple-Data). В результате разрабатывается один экземпляр CAF-программы, который далее копируется необходимое количество раз. Копии CAF-программы могут выполняться параллельно и обрабатывают свои «локальные» данные. Данные, к которым требуется доступ из разных копий CAF-программы, должны быть представлены в виде *co-arrays*-массивов. Передача данных между копиями программы осуществляется только при помощи явно указываемых действий.

Библиотека параллельных методов. Учебно-исследовательская библиотека параллельных методов ParaLibPGAS разработана для исследования эффективности языков про-

граммирования на основе модели PGAS (Chapel, X10, CAF). Библиотека содержит параллельные алгоритмы для решения широкого класса задач вычислительной математики (матричная алгебра, решение дифференциальных уравнений в частных производных и др.).Архитектура библиотеки представлена на рис. 1.

Для реализации интерфейса и служебных модулей библиотеки был использован язык Java. Каждый исполняемый файл на уровне библиотеке «обёртывается» в специально созданный класс, который реализует интерфейс для запуска алгоритма на исполнения. Преимущество данного подхода заключается в том, что класс-обёртка знает все ограничения, которые накладываются на метод и на входные данные и может проверить корректность этих данных ещё до запуска алгоритма на исполнение.

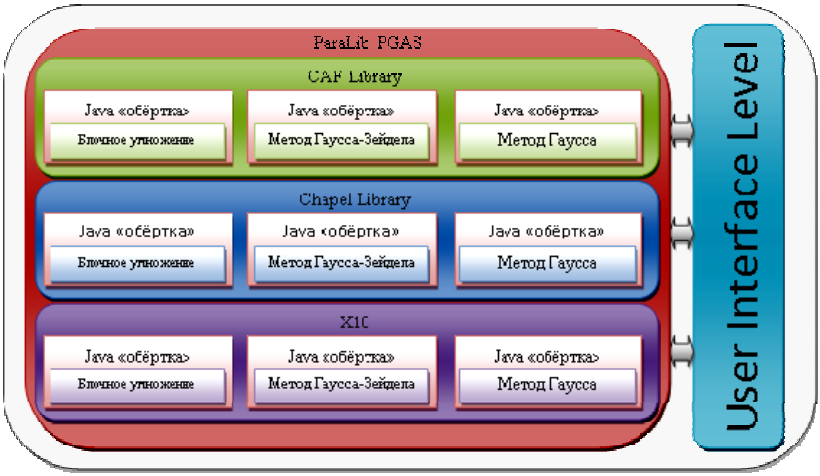


Рис. 1. Архитектура библиотеки ParaLibPGAS

Все методы для одного языка объединяются в классы библиотеки, которые предоставляют набор функций для работы с методами. Данный подход позволяет обеспечить расширяемость данной системы. Для добавления нового метода (или даже языка) необходимо добавить один новый класс в исходный код, всё остальное система будет делать автоматически (рис. 2).

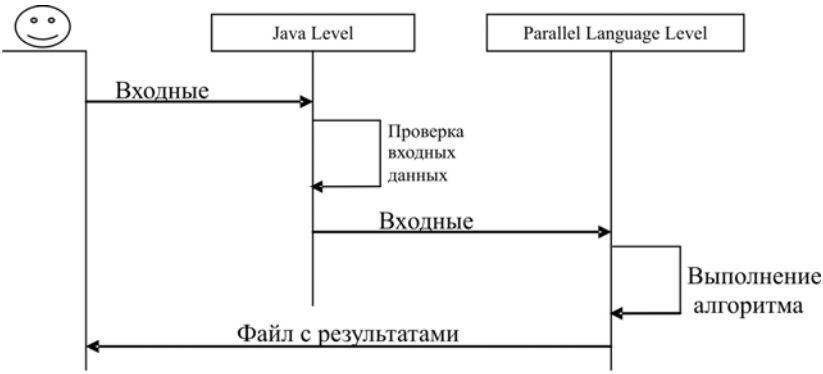


Рис. 2. Общая схема выполнения метода библиотеки

Результаты вычислительных экспериментов. Вычислительные эксперименты проводились на компьютере с процессором Intel Core 2 Quad CPU Q6850 2,40 GHz и оперативной памятью объемом 4 гигабайта под управлением ОС OpenSuse 11.4. Для компиляции программ используются компиляторы Chapel 1.1 для программ на Chapel и компилятор g95 для программ на CAF.

Вычислительные эксперименты заключались в сравнении последовательных версий алгоритмов матричного перемножения и решении систем дифференциальных уравнений с параллельными версиями этих же алгоритмов, написанных с помощью новых языков.

На рис. 3 представлены результаты вычислительного эксперимента матричного перемножения. Программы, разработанные на C++ и распараллеленные с помощью OpenMP, сравниваются с программами, разработанными на Chapel. Сравнение производится при проведении вычислений с использованием 2, 3 и 4 потоков.

На рис. 4 показаны результаты выполнения блочного параллельного алгоритма Гаусса-Зейделя для языка CAF (4 и 16 блоков).

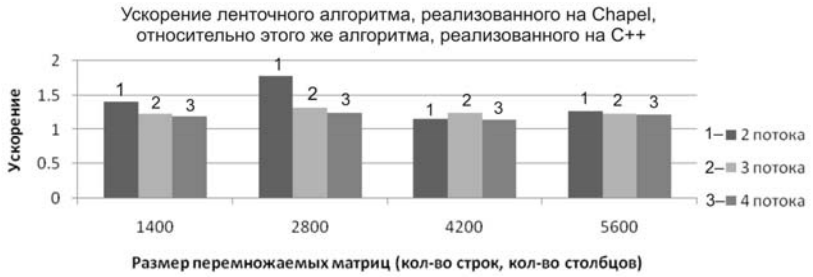


Рис. 3. Результаты экспериментов для языка Chapel

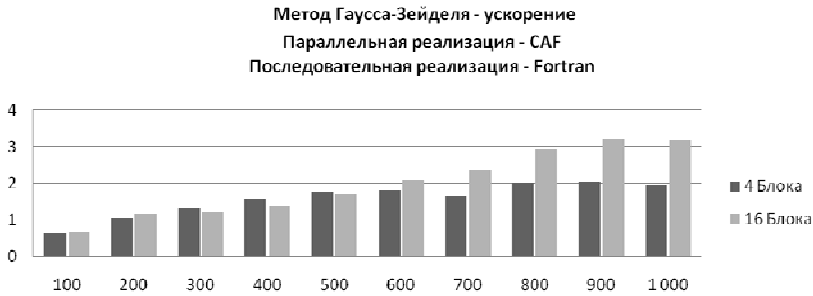


Рис. 4. Результаты экспериментов для языка CAF

В ходе выполнения работы были изучены новые языки параллельного программирования, разработана библиотека параллельных методов ParaLibPGAS, проведены вычислительные эксперименты. Результаты экспериментов показывают достаточную эффективность параллельных программ, разработанных с использовнием новых языков.

Планируемые в дальнейшем работы связаны с расширением состава библиотеки ParaLibPGAS, проведением более расширенного набора вычислительных экспериментов, сопоставлением трудоемкости разработки параллельных программ с использованием разных языков параллельного программирования.

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

Список литературы

1. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. – 424 с.
2. Сайт алгоритмического языка Chapel – http://chapel.cray.com.
3. Сайт алгоритмического языкаX10 – http://x10-lang.org.
4. Сайт алгоритмического языкаCAF – http://www.co-array. org.

В.П. Гергель, А.А. Сиднев

Нижегородский государственный университет им. Н.И. Лобачевского

О ПРИМЕНЕНИИ МАКРОМОДУЛЬНОЙ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММ

Разработка программного обеспечения является трудоемкой профессиональной деятельностью. Для упрощения разработки программного обеспечения разработчики выполняют декомпозицию задачи на более мелкие и простые подзадачи, организуя их в отдельные вычислительные блоки (модули). Разработанные модули могут быть использованы в дальнейшем повторно, оформляясь в виде отдельных библиотек. Модульный подход разработки программного обеспечения является на данный момент общепризнанным подходом.

Одним из недостатков такого подхода является отсутствие стандартов на интерфейсы модулей. Разработчик библиотеки сам определяет удобные ему структуры хранения данных и интерфейсы функций, которые их обрабатывают. В результате каждая библиотека получается уникальной и возникает сложность, связанная с заменой используемых библиотек.

Другая проблема при использовании библиотек – проблема выбора наиболее оптимальной библиотеки под текущие задачи проекта. Для решения широкого круга задач существует достаточно много библиотек. Каждая библиотека может иметь

свою сложность внедрения, эффективность реализации, удобство использования структур данных и поддержку различных программно-аппаратных платформ. Задача выбора лучшей библиотеки может не иметь однозначного решения, поэтому разработчикам приходится идти на компромисс при выборе библиотеки, теряя в эффективности реализации, удобстве использования или количестве поддерживаемых программно-аппаратных платформ. Со временем возможностей текущей библиотеки под решение очередной задачи может не хватать, поэтому разработчики могут столкнуться с задачей перехода на другую библиотеку в связи с такими причинами, как:

- переход на другую программно-аппаратную платформу;
- требуование большой эффективности;
- отказ от использования текущей версии библиотеки в связи с прекращением её поддержки (она устарела).

При переходе на новую библиотеку разработчику придётся столкнуться с необходимостью выполнить модификацию используемых структур данных и функций под те, которые используются в библиотеке. Такой переход может быть очень трудоёмким.

Рассматриваемая макромодульная технология (ММТ) разработки программ позволят решить указанные проблемы.

Идея технологии. В основе технологии макромодульной разработки программ лежит идея макроописания участков кода программы. Такое описание содержит информацию о том, какие действия выполняются в указанном участке кода и какие структуры данных при этом используются. Действием может быть решение системы линейных уравнений, перемножение матриц, выполнение быстрого преобразования Фурье и др.

На основании описания макросреда, реализующая поддержку макромодульной технологии, может выполнять замену пользовательского кода на вызовы библиотечных функций. Каждое макроописание соответствует абстрактной функции, а для каждой абстрактной функции может существовать несколько реализаций из одной или нескольких библиотек (рис. 1). Таким образом, разработчик получает возможность использовать под-

ходящую ему библиотеку без внесения дополнительных изменений в код. Макросреда берёт на себя работы, связанные с сопряжением интерфейсов, используемых в пользовательской программе и целевой библиотеке.

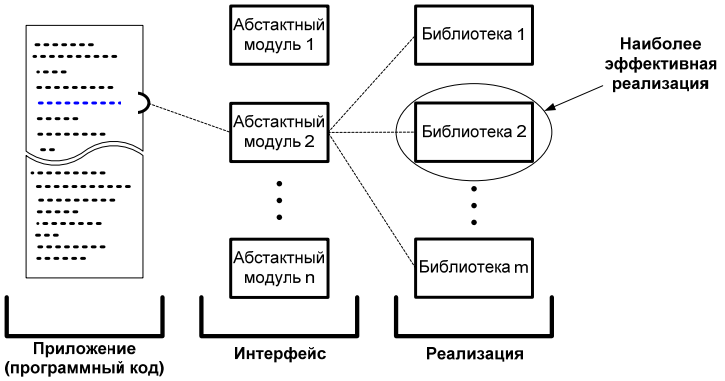


Рис. 1. Общая схема макромодульной технологии

Реализация технологии. Технология макромодульной разработки программ может применяться к любому языку программирования. Далее будет рассматриваться язык C/C++.

Наиболее удобным средством для выполнения макроописания в языке C/C++ являются директивы препроцессора [1]. Мы будем использовать директивы pragma, которые предназначены для реализации возможностей компиляторов и наилучшим образом подходят для выполнения макроописания. В том случае, если среда разработки не поддерживает макроописания, соответствующие директивы будут проигнорированы и будет собрано приложение, написанное пользователем.

Каждое макроописание соответствует абстрактной функции с теми типами данных, которые используются в программе пользователя – метафункции. Абстрактная функция может иметь несколько реализаций из одной или нескольких библиотек. Выбор наиболее подходящей реализации может быть выполнен пользователем.

Для упрощения модификации, использования и расширения возможностей макромодульной технологии все необходимые данные хранятся в текстовых файлах формата расширяемого языка разметки XML. Для описания макросреды используется набор из 5 файлов (рис. 2):

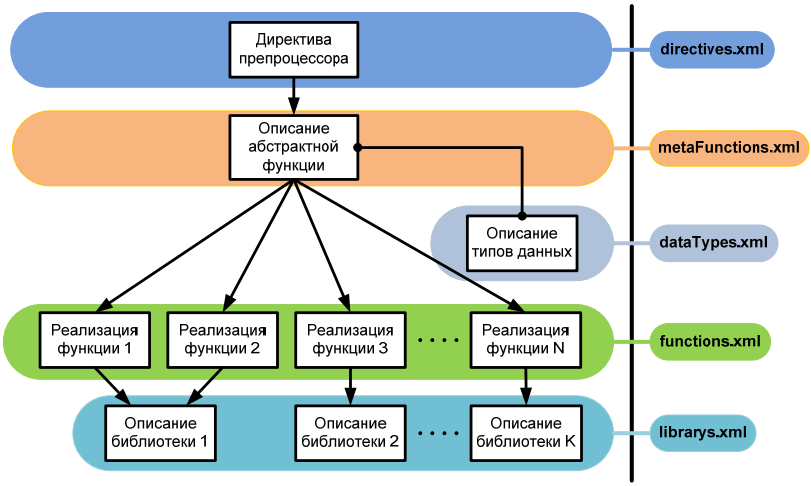


Рис. 2. Реализация макромодульной технологии

- описание поддерживаемых директив (directives.xml);
- описание метафункций (metaFunctions.xml);
- описание структур данных (dataTypes.xml);
- реализация функций (functions.xml);
- описание библиотек (librarys.xml).

Для каждой директивы создаётся запись в файле directives.xml с указанием абстрактной функции, которая выполняет указанный с помощью директивы код. Интерфейс функции описывается с помощью типов данных, представленных в dataTypes.xml. Доступные абстрактные функции представлены в файле metaFunctions.xml. Каждой абстрактной функции соответствует одна или несколько реализаций, использующих библиотеки. Описание таких реализаций представлено в functions.xml. Для каждой реализации указы-

ваются требуемые заголовочные файлы и статические библиотеки. Описание используемых библиотек содержится в librarys.xml. В описании указаны пути к заголовочным файлам, статическим библиотекам и динамическим библиотекам (эти пути могут задаваться при установке соответствующей библиотеки).

Использование макромодульной технологии происходит перед компиляцией исходных кодов программы на этапе препроцессирования. Для каждой директивы макромодульной технологии выполняется замена указанного блока в программном коде на соответствующую реализацию из библиотеки. Если таких реализаций несколько, то выбор функции осуществляется в соответствии с конфигурацией, указанной разработчиком.

Пример использования. Для удобства использования макромодульной технологии было разработано расширение к среде разработки Microsoft Visual Studio 2008, которое позволяет автоматизировать применении технологии [5]. Расширение позволяет выполнять предварительное препроцессирование исходных кодов программы и задавать целевую конфигурацию для сборки.

В качестве примера использования технологии была выбрана задача матричного умножения [2]. Программа содержит макроописание и простейшую реализацию матричного умножения. В качестве библиотечных реализаций были выбраны библиотека MKL [3] и технология для вычислений на графических картах CUDA [4].

Ниже представлены результаты вычислительных экспериментов на тестовой системе с видеокартой GeForce 8800 GTS и двухъядерным процессором Intel Core 2 E6650.

Из результатов эксперимента видно, что реализация пользователя неэффективна и выполнение матричного умножения на графической плате эффективнее, чем на процессоре на данной тестовой системе. На системе с большим количеством вычислительных ядер ситуация может измениться. Таким образом, пользователь получает три реализации программы с минимальными

затратами на разработку. Эффективность программы и используемой библиотеки определяется целевой платформой.

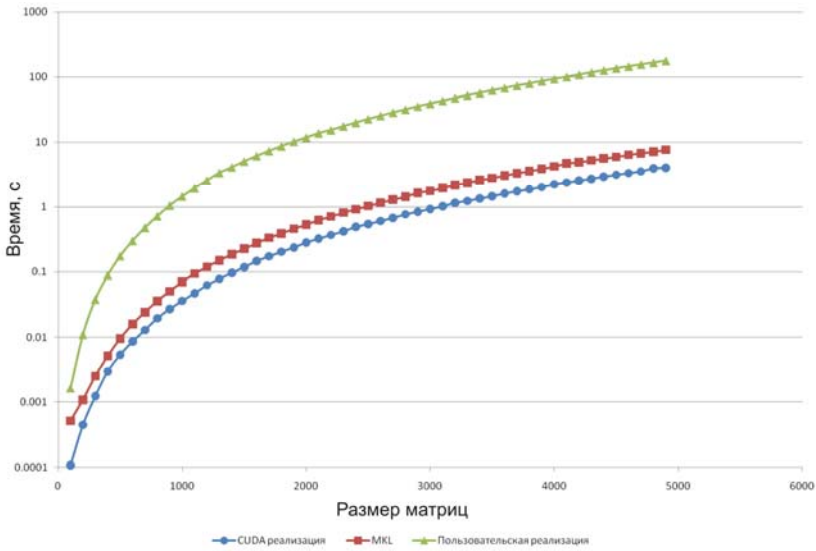


Рис. 3. Результаты экспериментов

Рассмотренная технология макромодульной разработки программ позволяет упростить разработку программного обеспечения за счёт сокращения затрат на внедрение библиотек. Пользователю достаточно сделать макроописание участков кода, и сборка приложения будет осуществляться автоматически с использованием наиболее подходящей библиотеки. В результате решается проблема с выбором наиболее оптимальной библиотеки, а переход на использование новой библиотеки значительно упрощается.

Для поддержки макромодульной технологии используется механизм директив препроцессора. В том случае если среда разработки не поддерживает макроописания, то соответствующие директивы будут игнорироваться и будет собрано приложение, написанное разработчиком.

Список литературы

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с.

2. Knuth D.E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms. Addison-Wesley Professional; 3 edition. November 14, 1997. – P. 501.

3. Intel® Math Kernel Library. Reference Manual. September 2007.

4. NVidia CUDA. Reference Manual. Version 2.3. July 2009.

5. Craig S., Marc Y., Brian J. Working with Microsoft® Visual Studio® 2005. Microsoft Press.

В.П. Гергель, К.А. Чабан

Нижегородский государственный университет им. Н.И. Лобачевского

СИСТЕМА РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ DCS

В настоящее время многие организации имеют в своем распоряжении довольно крупную компьютерную сеть, объединяющую десятки и даже сотни компьютеров. С ростом такой сети все актуальнее становится вопрос об эффективности использования вычислительных ресурсов, которые она объединяет. Зачастую имеющиеся компьютеры загружены не полностью. Поэтому вполне естественно задействовать простаивающие ресурсы для выполнения какой-либо полезной работы – например, использовать их для решения некоторой трудоемкой задачи. Развитие данного направления привело к возникновению концепции *utility computing*.

Основная идея данной концепции состоит в использовании независимых компьютеров, объединенных в сеть, в качестве единой параллельной машины или виртуального суперкомпьютера. Эта идея имеет ряд преимуществ, например низкая стоимость суммарных вычислительных ресурсов.

Быстрый рост сети Интернет позволяет применить эти концепции в намного большем масштабе. К тому же настольные

ПК в корпоративных и домашних условиях нагружены далеко не полностью – обычно используется только одна десятая часть мощности. Поэтому возникает интерес к использованию вычислительных мощностей, которые доступны в виде свободных циклов работы центрального процессора.

В сфере программного обеспечения грид на сегодняшний день доступно множество различных инструментариев для организации распределенных вычислений. В качестве примеров можно привести инструментарии Globus Toolkit [http://www.globus.org], Alchemi .NET Framework [http://www.alchemi.net], X-Com [http://x-com.parallel.ru]. В настоящей работе рассматривается система Distributed Calculation System (DCS), с помощью которой можно организовать распределенную вычислительную среду из имеющихся компьютерных ресурсов. Согласно распространенной классификации данную систему можно отнести к классу *инструментариев*[1], так как она включает в себя базовые средства разработки распределенных приложений и поддерживает необходимые для их работы операции.

Сегодня наблюдается быстро растущий интерес к технологиям грид со стороны коммерческих предприятий и исследовательских организаций. Операционная система Microsoft Windows широко используется большинством фирм. И для массового распространения грид и распределённых вычислений необходимо разработать инструментарии, поддерживающие данную операционную систему. Это позволило бы эффективно использовать свободную вычислительную мощность настольных компьютеров и рабочих станций посредством создания виртуального вычислительного ресурса, стоимость которого значительно ниже стоимости традиционных суперкомпьютеров.

---

[1] Существующие программные средства грид можно разделить на следующие категории: *программные средства*, *базовые средства грид (инструментарии)* и *платформы*. Такая классификация отражает полноту решения задачи создания и обслуживания грид, поддержки работы пользователей и разработки грид-приложений. Более подробную информацию можно найти на ресурсе http://www.gridclub.ru.

Описание системы. Система DCS предназначена для создания распределённой вычислительной сети в соответствии с принципами клиент-серверной архитектуры. Distributed Calculation System состоит из 3 основных компонентов:

1. ***DCS Server*** – центральная часть системы, которая представляет собой ASP.NET Web-сервис, связанный с экземпляром Microsoft SQL Server 2008. Основные функции сервера можно разделить на 2 группы:

Методы для работы клиента администратора:

- добавление вычислительных заданий;
- получение результатов расчёта;
- оперативное получение информации о прогрессе расчётов;
- сбор сведений о предоставленных сервисом вычислительных ресурсах, их активности;
- on-line настройка параметров сервиса;
- управление ресурсами сервиса.

Методы для работы вычислительного клиента:

- регистрация клиента на сервере,;
- получение части (порции) задания для расчёта;
- передача частичного результата расчёта на сервис для хранения.

2. ***DCS Administration Client*** – клиент администратора системы. Данное приложение является основным инструментом пользователя при работе с системой.

Можно выделить 3 основных логических модуля данного компонента вычислительной сети. ***Менеджер задач***, предназначенный для формирования пакетов вычислительных заданий, отправки их на сервер и последующего получения результатов вычислений. С помощью менеджера производится также удаление заданий и приостановка расчета. ***Монитор активности вычислительных клиентов.*** Данный модуль предоставляет сводную информацию об основных параметрах клиентов, подключённых к сервису. С его помощью можно оценить мощность вычислительного ресурса, сформированного с помощью инструментария DCS. ***Мастер настроек сервиса*** – служебная утилита, помогающая удалённо изменять настройки сер-

виса. С её помощью можно осуществлять перезапуск сервиса, удаление информации о клиентах, задачах, а также чтение журнала работы.

3. ***DCS Calculation Client*** – компонент системы, который организует расчет отдельных порций входных данных с помощью прикладной программы. Клиенты системы DCS отвечают за выполнение следующих действий:

- выполнение вычислений для имеющегося блока (порции) входных данных прикладной задачи;
- запрос новых блоков данных для вычислений от сервера;
- передача результатов вычислений на сервер.

Система DCS подразумевает использование имеющихся вычислительных ресурсов для обработки клиентским приложением, запущенным на компьютере, блоков некоторой задачи, предоставляемых приложению сервером. Следовательно, задача должна допускать декомпозицию входного набора данных на независимые части или, иными словами, должен иметь место параллелизм по данным. После полной обработки данных блоков задача должна считаться полностью решенной. Это требование вызвано необходимостью организации процесса решения задачи в наиболее универсальном виде.

Требование независимости порций данных вызвано идеологией организации любой распределенной вычислительной системы, при которой каждый конкретный блок задачи будет обрабатываться на каком-либо доступном системе компьютере. Причем другие порции данных, возможно, будут обрабатываться на других компьютерах, не связанных с данным. Хотя потенциальный доступ к данным возможен, он существенно замедлит процесс выполнения задачи ввиду низких скоростей передачи данных по сети по сравнению с локальным доступом к данным.

Вычислительное задание в рамках системы DCS является совокупностью следующих элементов. *Вычислительный модуль* – исполняемый файл со всеми зависимостями (динамическими библиотеками, файлами конфигурации и т.д.), реализующий алгоритм расчета. Данное приложение используется клиентами для проведения вычислений. Система DCS не накладывает ограничений на инструменты реализации данного компонента.

Тем не менее пользователь должен учитывать возможные проблемы запуска вычислительного модуля на предоставленных ему вычислительных ресурсах, так как особенностью применения инструментариев данного класса является неоднородность распределённой среды. Другими словами пользователь системы должен обеспечить возможность запуска своего приложения на компьютерах, составляющих вычислительную сеть DCS.

Generator.exe – генератор аргументов задачи. С помощью данной служебной утилиты реализуется алгоритм декомпозиции входного набора данных на независимые друг от друга части. Данное приложение предназначено для генерации входных параметров вычислительного модуля в зависимости от номера текущей порции задания. Generator.exe реализуется прикладным программистом системы.

Служебные данные задачи – набор параметров и ограничений, используемых компонентами системы во время вычисления, включающий:

- имя задачи;
- число порций разбиения;
- краткое описание задачи;
- максимальный срок выполнения, после которого задание считается отменённым.

Использование системы. Для демонстрации возможностей системы рассмотрим задачу визуализации четырёхмерных фракталов Julia методом трассировки лучей.

Фракталы Julia генерируются в 4-мерном пространстве кватернионов с помощью итерационной формулы $z_{n+1} = z_n^2 + c$, где c – какая-либо константа.

Для визуализации в привычное для человека трехмерное пространство осуществляется проекция 4-мерного пространства на трехмерное.

Трассировка лучей – это метод машинной графики, позволяющий создавать фотореалистичные изображения любых трехмерных сцен. Трассировка лучей моделирует прохождение лучей света через изображаемую сцену. Фотореализм достигается путем математического моделирования оптических свойств света и его взаимодействия с объектами. Сначала отдельные объекты распо-

лагаются в трехмерном пространстве-сцене, а также задаются физические и оптические свойства их поверхностей и их цвет. Затем определяется, где будут расположены источники света и наблюдатель. И, наконец, с помощью программы трассировки лучей создается математическая модель сцены, света и наблюдателя, с помощью которой вычисляется цвет каждого пикселя графического изображения, получаемого на экране дисплея.

Данная задача была решена с помощью системы DCS. Ниже приведено время эксперимента в зависимости от числа используемых клиентов:

Число клиентов	Время расчёта
1 клиент	23:35
2 клиента	12:45
3 клиента	08:14
4 клиента	06:08

Для экспериментов использовались компьютеры: Intel Core 2 Quad Q9550 2.83 Ghz, 2.5 Gb RAM, Windows XP SP3.

М.В. Гиркин, Д.И. Крыжановский, М.А. Фёдоров

Волгоградский государственный технический университет

МОДЕЛИРОВАНИЕ РАБОТЫ РЕЗИСТИВНОЙ ПАМЯТИ RRAM НА КЛАСТЕРНЫХ СИСТЕМАХ МЕТОДОМ МОНТЕ-КАРЛО

Как известно, суммарное время вычислений, осуществляемых одним последовательным вычислительным устройством, складывается из времени, непосредственно затрачиваемого на выполнение арифметических операций, и времени, затрачиваемого на операции с памятью. Развитие вычислительной техники, наблюдающееся в последние годы и сопровождающееся широким внедрением многоядерных процессоров и других параллельных систем, позволило значительно увеличить скорость вычислений, снизив как среднее время выполнения отдельных

элементарных операций, так и время выполнения арифметических вычислений в целом. Однако скорость обмена данными с памятью по-прежнему остаётся относительно низкой по сравнению со скоростью собственно вычислений, что зачастую снижает эффективность работы высокопроизводительных систем. Кроме того, традиционные типы памяти, основанные на сохранении заряда, практически исчерпали свои технологические ресурсы и достигли физического предела масштабирования и скорости обмена данными [7, 8]. В связи с этим разработка новых типов памяти, основанных на иных физических принципах, становится всё более актуальной.

По мнению ряда исследователей наиболее перспективными являются типы памяти, основанные на явлении резистивного переключения: CBRAM (память с мостом проводимости – *Conductive Bridge* – между двумя электродами), PCRAM (память, основанная на изменении сопротивления между двумя фазами – *Phase Change* – халькогенида) и RRAM (резистивная – *Resistive* – память). Эти типы памяти обладают наиболее простой структурой среди всех предложенных на настоящий момент вариантов энергонезависимой памяти (проводник – диэлектрик – проводник) и, кроме того, демонстрируют хорошую масштабируемость. Кроме того, RRAM характеризуется низким рабочим напряжением (< 2 В), малым временем переключения (< 10 нс), высокой плотностью и большим числом циклов перезаписи.

В недавних исследованиях был предложено несколько механизмов физических процессов, объясняющих работу резистивной памяти:

- модель, основанная на захвате носителей заряда [3];
- электрохимическая миграция вакансий кислорода [15];
- электрохимическая миграция ионов кислорода [10];
- ряд других моделей [4, 5, 6, 13, 14].

Однако, несмотря на значительные усилия, удовлетворительное фундаментальное понимание механизма резистивного переключения в RRAM по-прежнему отсутствует, что задерживает дальнейшее развитие данного типа памяти.

Ниже в работе представлена стохастическая модель механизма резистивного переключения, основанная на переходах электронов между вакансиями кислорода вдоль канала проводимости в слое оксида. Модель была предложена и протестирована в последовательном исполнении сотрудниками Института микроэлектроники Венского технического университета. Далее описаны численные улучшения, введённые в существующую реализацию модели, а также результаты её распараллеливания, выполненные сотрудниками факультета электроники и вычислительной техники Волгоградского государственного технического университета.

Описание модели. В предложенной модели [9] возникновение канала проводимости объясняется за счёт переходов электронов и вакансий кислорода. Рассматривается пять различных элементарных событий:

1) формирование вакансий кислорода за счёт смещения ионов из узлов кристаллической решётки;

2) аннигиляция вакансий в результате возвращения ионов кислорода в узлы кристаллической решётки;

3) переход электрона между двумя вакансиями;

4) переход электрона от вакансии к электроду;

5) переход электрона от электрода к вакансии.

В текущей реализации используется 41 входной параметр модели, задающие условия протекания процесса и задающиеся пользователем. В рамках настоящего проекта данная модель была сначала повторена, а затем распараллелена с использованием технологии MPI и протестирована на кластере Волгоградского технического университета. Чтобы избежать больших накладных расходов, было решено отказаться от распараллеливания вычислений внутри одного эксперимента. В данном случае различные параллельные потоки соответствуют отдельным перезапускам одного и того же эксперимента и разным наборам значений входных параметров модели. После отладки параллельной реализации модели были получены результаты, аналогичные представленным в работе [9].

Параллельная генерация случайных чисел. Для поддержки моделирования по методу Монте-Карло был реализован алгоритм генерации случайных чисел, предложенный Лекюйе [12], с периодом $\approx 2{,}3 \cdot 10^{18}$; для генерации нормально распределённых величин было использовано преобразование Бокса-Мюллера [12]. Чтобы обеспечить более высокую масштабируемость программного кода, генераторы случайных чисел были реализованы в рамках объектно ориентированной парадигмы. Структура объектной модели генераторов и преобразователей для заданных распределений приведена на рисунке.

Для параллельной генерации случайных чисел была использована комбинация репликационной (*replicated*) и распределённой древовидной (*distributed random tree*) схем. В главном потоке запускается последовательный генератор, выходные значения которого являются начальными значениями (*seed*) для остальных генераторов, функционирующих уже в параллельно исполняемых потоках. Именно выходные значения этих генераторов используются для дальнейшего розыгрыша случайных событий по методу Монте-Карло.

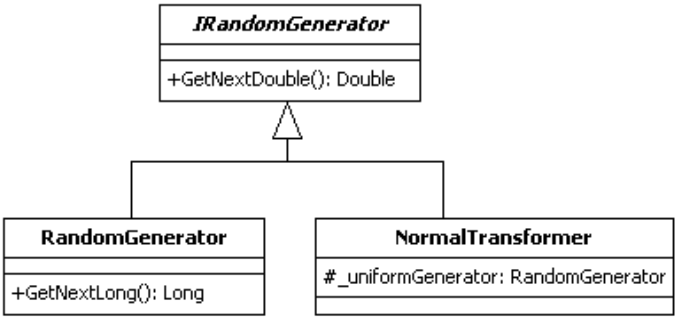


Рис. Структура объектной модели генераторов случайных чисел

Одно из требований, которому должны удовлетворять случайные величины, генерируемые в параллельных потоках, отсутствие корреляции между ними. Была поставлена серия экспериментов, в ходе которых генерировалось 50 последовательностей (каж-

дая в своём потоке) из 100 000 случайных чисел. По результатам экспериментов была составлена корреляционная матрица, элементы которой рассчитывались по формуле [1]

$$r_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}}, \quad (1)$$

где N – длина последовательностей случайных чисел, x_i и y_i – элементы двух последовательностей, проверяемых на наличие корреляции, $\bar{x}$ и $\bar{y}$ – средние значения проверяемых последовательностей (так как генерировались равномерно распределённые величины на полуинтервале [0;1), средние значения были приблизительно равны 0,5). Во всех экспериментах элементы корреляционной матрицы принимали значения на отрезке [0; 0,02] по абсолютному значению, что свидетельствовало об отсутствии статистически значимой корреляции между отдельными генераторами случайных чисел.

Чтобы получить более достоверное подтверждение этого факта, сгенерированные последовательности были также проверены и по критерию Фишера. Для всех пар последовательностей по формулам регрессионного анализа были построены линейные модели вида $y = a_0 + a_1x$, и во всех случаях они оказались неадекватны [2]. Таким образом, было получено дополнительное подтверждение отсутствия статистически значимой корреляции между последовательностями случайных чисел и пригодности.

План дальнейшего исследования. В настоящий момент описанное выше исследование находится на стадии усложнения модели и удаления из неё искусственных упрощений и допущений. Именно на данном этапе в полной мере скажутся преимущества параллельной реализации модели. Предлагаются следующие усовершенствования в модели:

- заменить 2D-моделирование на 3D;
- вместо одинакового расстояния между узлами кристаллической решётки ввести распределение расстояний;

• вместо жёстко заданного направления движения электронов ввести вероятностное распределение возможных направлений;

• ряд других усложнений.

По замыслу авторов это позволит получить более адекватную модель и добиться более глубокого понимания физической природы процессов, управляющих работой RRAM.

Список литературы

1. Львовский Е.Н. Статистические методы построения эмпирических формул: учеб. пособие для втузов. – 2-е изд., перераб. и доп. – М.: Высш. шк., 1988. – 239 с.

2. Петрович М.Л. Регрессионный анализ и его математическое обеспечение на ЕС ЭВМ: практ. руководство. – М.: Финансы и статистика, 1982. – 199 с.

3. Fujii T., Kawasaki M., Sawa A., Akoh H., Kawazoe Y., Tokura Y. Appl. Phys. Lett. Vol. 86. No. 1, art. No. 012107, 2005.

4. Gao B., Sun B., Zhang H., Liu L., Liu X., Han R., Kang J., Yu B. IEEE Electron Device Lett. Vol. 30. No. 12. P. 1326–1328, 2009.

5. Kim S., Choi Y.K. IEEE Trans. Electron Devices. Vol. 56. No. 12. P. 3049–3054, 2009.

6. Kinoshita K., Tamura T., Aso H., Noshiro H., Yoshida C., Aoki M., Sugiyama Y., Tanaka H. IEEE Non-Volatile Semicond. Memory Workshop, p. 84,85, 2006.

7. Kryder M.H., Kim C.S. IEEE Trans. Magn. Vol. 45. No. 10. P. 3406–3413, 2009.

8. Lee B.C., Zhou P., Yang J., Zhang Y.T., Zhao B., Ipek E., Mutlu O., Burger D. IEEE Micro. Vol. 30. No. 1. P. 131–141, 2010.

9. Makarov A., Sverdlov V., Selberherr S. First Principles Modeling of Bipolar Resistive Switching in Metal-Oxide Based Memory.

10. Nishi Y., Jameson J.R. Dev. Res. Conf. P. 271–274, 2008.

11. Numerical Recipes in C. The Art of Scientific Computing. − Second Edition, Cambridge University Press.

12. Robert C.P., Casella G. Introducing Monte Carlo Methods with R, Use R. − Springer Science + Business Media.

13. Rozenberg M.J., Inoue I.H., Sanchez M.J. Phys. Rev. Lett. Vol. 92. No. 17. P. 178302-1, 2004.

14. Russo U., Ielmini D., Cagli C., Lacaita A.L., Spiga S., Wiemer C., Perego M., Fanciulli M. IEDM Tech. Dig. P. 775–778, 2007.

15. Wu S.X., Xu L.M., Xing X.J. Appl. Phys. Lett. Vol. 93. No. 4. P. 043502/1-3, 2008.

В.В. Горбик, А.В. Панюков

Южно-Уральский государственный университет, г. Челябинск

ТЕХНИКА РЕАЛИЗАЦИИ СИМПЛЕКС-МЕТОДА НА КЛАСТЕРНЫХ СИСТЕМАХ

Применение симплекс-метода для практических задач линейного программирования остается вне конкуренции, несмотря на появившиеся полиномиальные алгоритмы [1]. В настоящее время используются две техники реализации симплекс-метода: 1)~метод симплекс-таблиц; 2)~метод обратной матрицы (модифицированный симплекс-метод).

Далее особенности данных методов будем рассматривать на примере задачи линейного программирования

$$\max\left\{c^T x : Ax = b \geq 0,\ x \geq 0;\ c, x \in \mathbf{R}^n;\ b \in \mathbf{R}^m\right\}.$$

Метод симплекс-таблиц итерации \$k\$ пересчитывает симплекс-таблицу

$$S^{(k)} = \frac{Z^{(k)} = c_{B(k)}^T B(k)^{-1} b \;\Big|\; z^{(k)} = -c^T + c_{B(k)}^T B(k)^{-1} A}{X_{B(k)} = B(k)^{-1} b \;\Big|\; B(k)^{-1} A},$$

где $B(k)$ − базисная матрица, содержащая все относящиеся к базисным переменным k-й итерации столбцы матрицы A (базисные столбцы); $c_{B(k)}$ − вектор коэффициентов целевой функ-

ции, относящихся к базисным переменным *k*-й итерации. При этом правый столбец симплекс-таблицы содержит вектор $X_{B(k)} = B(k)^{-1} b$ значений базисных переменных *k*-й итерации и значение $Z^{(k)}$ целевой функции на этом решении. Столбцы матрицы S, соответствующие базисным переменным, являются ортами (т.е. из них может быть составлена единичная матрица). Верхняя строка содержит вектор $z^{(k)} = -c^T + c_{B(k)}^T B(k)^{-1} A$ невязок двойственной задачи. Значения элементов строки $z^{(k)}$, относящиеся к базисным столбцам, являются нулевыми.

Критерием оптимальности текущего базисного решения является неотрицательность вектора z. Переход от таблицы *k*-й итерации к таблице $(k+1)$-й итерации осуществляется применением процедуры исключения Жордана-Гаусса для столбца i (ведущего столбца, вводимого в базис), с использованием в качестве ведущей строки l^* (определяющей выводимый из базиса столбец).

В методе симплекс-таблиц выполнение итерации, включая пересчет симплекс-таблицы, потребует не более $(m+n)$ операций деления и сравнения, а также не менее $m(n+1)$ операций сложения и умножения. Алгебраическая пространственная сложность табличного симплекс-метода в основном определяется числом операндов в симплекс-таблице, т.е. равна $mn + O(m)$.

Метод обратной матрицы – в отличие от табличного, на каждой итерации вместо пересчета симплекс-таблицы пересчитывается матрица $B(k)^{-1}$. Наличие обратной матрицы позволяет для текущего базиса легко находить $y(k)^T = c_{B(k)}^T B(k)^{-1}$ – соответствующее решение двойственной задачи.

Текущий базис является оптимальным, если соответствующее ему двойственное решение допустимо: $y(k)^T A \geq c^T$. Если же в матрице A найдется столбец

$$A_{i(k)} : y(k)^T A_{i(k)} < c_{i(k)},$$

то введение его в число базисных приведет к увеличению целевой функции.

Образом столбца $A_{i(k)}$ в симплекс-таблице $S(k)$ будет вектор $g = B(k)^{-1} A_{i(k)}$ \$. Поэтому ведущей строкой, определяющей выводимый из базиса столбец, будет

$$r = \arg\min_{l:\, g_l > 0} \frac{X_{B(k)l}}{g_l}.$$

Оценим алгебраическую вычислительную сложность рассматриваемого метода. Очевидно, что в общем случае на заключительной итерации потребуется проверка всех n ограничений двойственной задачи, для чего потребуется не более mn операций умножения $m(n-1)$ операций сложения. На промежуточных итерациях для установления недопустимости двойственного решения эта величина, как правило, является существенно меньшей.

Пересчет обратной матрицы потребует не более m операций деления, не более m^2 операций умножения и не более (m^2-1) операций сложения/вычитания. Поскольку $m < n$,то затраты вычислительных ресурсов на пересчет обратной матрицы могут быть существенно ниже затрат на пересчет симплекс-таблицы. Алгебраическая пространственная сложность метода обратной матрицы определяется числом элементов в исходных данных и в обратной матрице, т.е. равна $mn + m^2 + O(m)$, т.е. меньше, чем в табличном симплекс-методе.

Отметим, что алгебраическая сложность дает адекватные оценки используемого вычислительного ресурса, только когда все операнды имеют одинаковую длину, например, при использовании стандартных типов данных.

При выполнении дробно-рациональных вычислений без округления [2] память, необходимая для операндов, динамически изменяется (как правило, возрастает), поэтому в этом случае более адекватными являются оценки фактически достаточного объема памяти и числа элементарных операций с битами. Эти

величины принято называть битовой сложностью (пространственной и вычислительной соответственно).

Битовая сложность абсолютно точной реализации симплекс-метода. Практическая реализуемость вычислений без округления, в частности, безошибочного решения задачи линейного программирования, определяется требуемыми для вычислений ресурсами: числом бит оперативной памяти и количеством операций с битами. Далее через $S(\lambda)$ будем обозначать число бит, требуемое для представления объекта λ, а через $C(\lambda)$ – число битовых операций, выполненных при нахождении представления объекта λ.

Предложение 1. *Пусть* $B = (b_{ij})$ – *целочисленная* $m \times m$ *матрица,* $l = \max\limits_{i,j=1,2,\ldots,n} S(a_{ij})$.

Тогда $S(\det A) \le n(\log_2 m + l)$.

Предложение 2. *Пусть* $B = (b_{ij})$ – *рациональная* $m \times m$ *матрица,* $l = \max\limits_{i,j=1,2,\ldots,m} S(b_{ij})$.

Тогда $S(\det B) \le m(\log_2 m + (2m+1)l)$.

Из формул Крамера для систем линейных алгебраических уравнений и доказанных утверждений вытекает

Следствие. *Если один численный элемент исходных данных имеет пространственную сложность не более* l, *то*

1) элементы симплекс-таблицы и обратной матрицы имеют пространственную сложность не более $4lm^2 + lm + m\log_2 m + 1$;

2) столбец симплекс-таблицы и обратной матрицы имеют пространственную сложность не более $4lm^3 + lm^2 + m^2 \log_2 m + m$;

3) для представления симплекс-таблицы достаточно $4lm^3 n + o(lm^2 n)$ *бит;*

4) для представления обратной матрицы достаточно $4lm^4 + o(lm^3)$ *бит.*

Поскольку $m < n$, а наиболее критическим ресурсом при использовании точных дробно-рациональных вычислений является память, то целесообразным является использование метода обратной матрицы.

Рассмотрим далее **параллельные версии симплекс-метода.**

Для реализации **метода симплекс-таблиц** наиболее подходящей является декомпозиция симплекс-таблицы по столбцам на число блоков, равное числу процессоров *N*.

Все столбцы симплекс-таблицы, за исключением левого столбца, делятся в равных пропорциях между *N* процессами, левый столбец, т.е. вектор значений базисных переменных и значение целевой функции на нем, рассылаются всем процессам и обрабатываются ими независимо.

Пример разбиения симплекс-таблицы S на блоки $S(K), K = 1,2,\ldots,N$ представлен на рисунке

Процесс $K = 1,2,\ldots,N$				
$S_{00} = Z$	$z_{\left\lceil \frac{(K-1)n}{N} \right\rceil + 1}$	$z_{\left\lceil \frac{(K-1)n}{N} \right\rceil + 2}$	…	$z_{\left\lceil \frac{Kn}{N} \right\rceil}$
$S_{10} = X_{B1}$	$S_{1\left\lceil \frac{(K-1)n}{N} \right\rceil + 1}$	$S_{1\left\lceil \frac{(K-1)n}{N} \right\rceil + 2}$	…	$S_{1\left\lceil \frac{Kn}{N} \right\rceil}$
$S_{20} = X_{B2}$	$S_{2\left\lceil \frac{(K-1)n}{N} \right\rceil + 1}$	$S_{2\left\lceil \frac{(K-1)n}{N} \right\rceil + 2}$	…	$S_{2\left\lceil \frac{Kn}{N} \right\rceil}$
⋮	⋮	⋮	⋱	⋮
$S_{m0} = X_{Bm}$	$S_{m\left\lceil \frac{(K-1)n}{N} \right\rceil + 1}$	$S_{m\left\lceil \frac{(K-1)n}{N} \right\rceil + 2}$	…	$S_{m\left\lceil \frac{Kn}{N} \right\rceil}$

Рис. Декомпозиция симплекс-таблицы по процессорам

Принятые соглашения положены в основу параллельного алгоритма **TabularSimplex.**

Агоритм использует одна широковещательная коммуникация между процессами при нахождении ведущего процесса K^* для обмена значениями Δ_{i_K}, $K = 1,2,\ldots,N$ и одна широкове-

щательная коммуникация для передачи ведущего столбца $S_{i_{K^*}}$ и числа l_{K^*}.

В табл. 1 приведена сводка требуемого алгебраического (т.е. количества используемых алгебраических операций) вычислительного ресурса для последовательной и параллельной реализаций метода симплекс-таблиц.

Таблица 1

Алгебраический вычислительный ресурс реализаций метода симплекс-таблиц

Оператор	Один процессор	N процессоров	
	Количество алгебраических операций	Количество пересылаемых операндов	Нагрузка на один процесс
Проверка условия оптимальности	$[1, N]$		$[1, n/N]$
Определение ведущей строки	$2m+n+2$		$2m+n/N+2$
Выбор ведущего процесса		N	$\lceil \log_2 N \rceil$
Пересылка ведущего столбца		$m+2$	
Пересчет симплекс-таблицы	$2(m+1)(n+1)$		$2m(1+n/N)$
Итого	$[1, n]+2mn+4(m+n+1)$	$m+N+2$	$[1, n/N]+2(n/N)(m+1)+4m+2+\log_2 N$

Из табл. 1 и описания алгоритма видно, что процессоры загружены равномерно, а эффективность распараллеливания растет с ростом сложности задачи, достигая в пределе 100\%.

Из описания **метода обратной матрицы** следует, что максимальный эффект от распараллеливания при поиске вводи-

мой в базис переменной достигается при декомпозиции исходных данных: вектора c^T и матрицы A, – по столбцам в равных пропорциях между процессорами на блоки

$$c(K)^T = \left(c_{\left\lceil \frac{(K-1)n}{N} \right\rceil + 1} \quad c_{\left\lceil \frac{(K-1)n}{N} \right\rceil + 2} \quad \cdots \quad c_{\left\lceil \frac{Kn}{N} \right\rceil} \right), \quad K = 1,2,\ldots,N;$$

$$A(K) = \begin{pmatrix} a_{1\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 1\right)} & a_{1\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 2\right)} & \cdots & a_{1\left(\left\lceil \frac{Kn}{N} \right\rceil\right)} \\ a_{2\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 1\right)} & a_{2\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 2\right)} & \cdots & a_{2\left(\left\lceil \frac{Kn}{N} \right\rceil\right)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 1\right)} & a_{m\left(\left\lceil \frac{(K-1)n}{N} \right\rceil + 2\right)} & \cdots & a_{m\left(\left\lceil \frac{Kn}{N} \right\rceil\right)} \end{pmatrix}, K = 1,2,\ldots,N.$$

При этом предполагается, что вектор двойственных переменных y размещен в каждом процессе $K = 1,2,\ldots,N$.

Эффект от распараллеливания при пересчете обратной матрицы B^{-1} будет максимальным при ее декомпозиции по строкам в равных пропорциях на блоки по числу процессов:

$$B^{-1}(K) = \begin{pmatrix} b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 1\right)1} & b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 1\right)2} & \cdots & b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 1\right)m} \\ b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 2\right)1} & b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 2\right)2} & \cdots & b_{\left(\left\lceil \frac{(K-1)m}{N} \right\rceil + 2\right)m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{\left(\left\lceil \frac{Km}{N} \right\rceil\right)1} & b_{\left(\left\lceil \frac{Km}{N} \right\rceil + 2\right)2} & \cdots & b_{\left(\left\lceil \frac{Km}{N} \right\rceil + 2\right)m} \end{pmatrix}, K = 1,2,\ldots,N.$$

Из описания метода обратной матрицы следует целесообразность декомпозиции вектора базисных переменных X_B и вектора g по таким же блокам строк:

$$X_B(K)=\begin{pmatrix} x_{B\left\lceil\frac{(K-1)m}{N}\right\rceil+1} \\ x_{B\left\lceil\frac{(K-1)m}{N}\right\rceil+2} \\ \vdots \\ x_{B\left\lceil\frac{Km}{N}\right\rceil} \end{pmatrix}; g(K)=\begin{pmatrix} g_{\left\lceil\frac{(K-1)m}{N}\right\rceil+1} \\ g_{\left\lceil\frac{(K-1)m}{N}\right\rceil+2} \\ \vdots \\ g_{\left\lceil\frac{Km}{N}\right\rceil} \end{pmatrix}; K=1,2,\ldots,N.$$

Способ декомпозиции обратной матрицы по процессам позволяет каждому процессу вычислить субвектор двойственных переменных:

$$y(K)=\left(B(K)^{-1}\right)^T c_B(K), \quad y=\sum_{K=1}^{N} y(k).$$

Таким образом, описание параллельной версии метода обратной матрицы оказывается сложнее описания табличного симплекс-метода. В основном это объясняется большей степенью неоднородности используемых структур. При выполнении одной итерации используется пять широковещательных коммуникаций между процессами:

1) широковещательная коммуникация между процессами при нахождении ведущего процесса K^c для обмена значениями z_{iK}, K=1,2,…,N;

2) широковещательная коммуникация для передачи вводимого в базис (ведущего) столбца A_{iK} и его номера;

3) широковещательная коммуникация между процессами при нахождении ведущего процесса K^r для обмена значениями $x_{B(K)} / g(K), c_{B(K)}, K=1,2,\ldots,N$;

4) широковещательная коммуникация для передачи ведущей строки;

5) широковещательная коммуникация для обмена между процессами двойственными субрешениями и нахождения двойственного решения.

Изложенное выше было использовано в алгоритме **InvBasMatrSimplex**. При выполнении одной итерации используется пять широковещательных коммуникаций между процессами.

В табл. 2 приведена сводка требуемого алгебраического вычислительного ресурса для последовательной и параллельной реализаций метода обратной матрицы.

Таблица 2

Алгебраический вычислительный ресурс реализаций метода обратной матрицы

Оператор	Один процессор	N процессоров	
	Количество алгебраических операций	Количество пересылаемых операндов	Нагрузка на один процесс
Проверка условия оптимальности	$2m\lceil 1,n\rceil$		$2m[1,n/N]$
Выбор c-ведущего процесса		N	$\lceil \log_2 N \rceil$
Пересылка ведущего столбца		$m+1$	
Нахождение g	$m(m\text{-}1)$		$m(m\text{-}1)/N$
Определение ведущей строки	$2m$	N	$2m/N+\log_2 N$
Модификация X_B	$1+2m$	1	$2m/N$
Модификация B^{-1}	$3m^2$	m	$3m^2/N$
Вычисление $y(K)$			
Вычисление y	$m(2m\text{-}1)$	N	$\lceil \log_2 N \rceil$
Итого	$2m[1,n]+6m^2+2m-1$	$2m+3N+2$	$2m[1,n/N]+6m^2/N+2m/N\ 3\log_2 N$

Из табл. 2 и описания алгоритма видно, что процессоры загружены равномерно, а эффективность распараллеливания растет с ростом сложности задачи, достигая в пределе 100\%.

Список литературы

1. Exact and Guaranteed Accuracy Solutions of Linear Programming Problems. by Distributed Computer Systems with MPI / *A. V. Panyukov, V. V. Gorbik* // Tambov University REPORTS: A Theoretical and Applied Scientific Journal. Series: Natural and Technical Sciences. Vol. 15, Issue 4, 2010. – P. 1392–1404. – URL: http://vestnik.tsutmb.ru/old/index.php?module=subjects\&func=viewpage\&pageid=165

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

Е.С. Городецкий

Нижегородский государственный университет им. Н.И. Лобачевского

ОРГАНИЗАЦИЯ МНОГОПОТОЧНЫХ ВЫЧИСЛЕНИЙ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ В ЗАДАЧЕ РАСЧЁТА КИНЕМАТИКИ РЫЧАЖНЫХ МЕХАНИЗМОВ

Пространственные рычажные механизмы являются важной составляющей современной техники и производственных технологий [8]. Механические системы разрабатываются для выполнения достаточно серьёзных задач, требующих высокой надёжности и точности. При конструировании механизма обычно требуется многократно выполнять полный расчет его движения для постепенной оптимизации модели. Наибольший интерес для конструктора представляют технологии и инструменты для оптимального моделирования механических систем. Тем не менее такие возможности практически отсутствуют в современных общедоступных системах автоматизированного проектирования. Задача параметрической оптимизации модели рычажного

механизма была формализована и описана автором [6]. Дальнейшая работа ведётся в направлении автоматизации решения этой задачи с распараллеливанием векторных вычислений на графических картах.

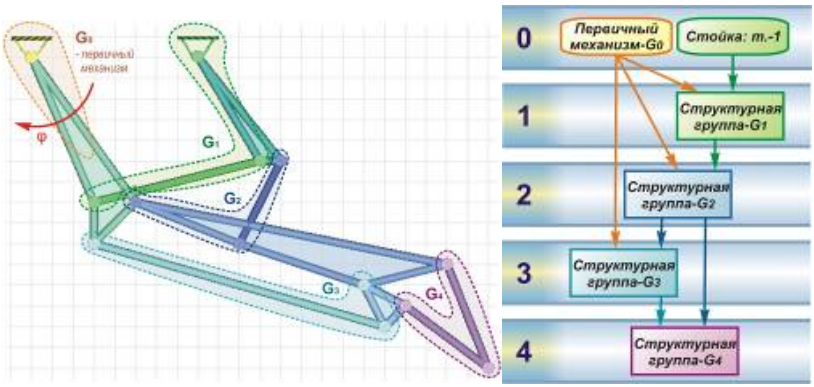
Постановка задачи. Для эффективного решения задачи оптимизации кинематики механизма требуется применение максимально быстрых алгоритмов вычисления положений, реализованных под современные многоядерные архитектуры. Схема параллельных вычислений и структур данных для расчета положений пространственного механизма была разработана автором в общем виде для CPU и GPU [5]. В данной работе описывается организация многопоточных вычислений на графическом процессоре с использованием технологии DirectX 11 Compute Shader [2], используемые структуры данных и результаты вычислительного эксперимента.

Введение в задачу о положениях пространственного рычажного механизма. Под *механизмом* понимается совокупность взаимосвязанных твёрдых тел, предназначенная для преобразования движения входов на одном или нескольких твёрдых телах в движение на других твёрдых телах (выходах). Механизм может быть представлен своей структурной схемой, на которой выделяют *звенья* (твёрдые тела), соединяющиеся в подвижные *кинематические пары* с помощью различных типов *геометрических элементов* (вращательных, сферических, поступательных и др.). В основе методов конструирования механизмов и расчета кинематики лежит *принцип образования механизмов по Ассуру*, утверждающий что «Всякий механизм представляет собою совокупность одного или нескольких *двухзвенных* (*первичных*) *механизмов* и одной или нескольких *групп нулевой подвижности* (*структурных групп*)» [8]. В работе рассматривается класс пространственных рычажных механизмов, которые могут быть образованы согласно принципу Ассура наслоением одноконтурных структурных групп.

Прямая задача о положении для механизмов ставится следующим образом. Задано движение всех входных звеньев,

удовлетворяющее связям, которые накладывают кинематические пары, соответствующих первичных механизмов. Необходимо найти движение всех остальных звеньев механизма либо сообщить о невозможности реализации положения в тот или иной момент времени.

Согласно закону Ассура прямая задача о положениях механизма сводится к последовательному решению ряда более простых подзадач расчета положений структурных групп механизма. На рис. 1, *а* приведён пример механизма выдвижения закрылков самолёта. Механизм состоит из одного первичного механизма и четырех структурных групп ВВВ (из трех вращательных пар), связанных друг с другом. Каждая группа имеет два входа, необходимых для расчета её положения. Исходя из структурной схемы механизма строится последовательность расчета структурных групп, показанная на рис. 1, *б* – вычисления производятся последовательно по уровням 0, 1, 2 и т.д., сверху вниз (стрелками показаны соединения групп друг с другом). Положения структурных групп ВВВ вычисляются аналитически. Произведя вычисления положений механизма для разных углов φ поворота входного звена, мы получим движение всех звеньев механизма.



а *б*

Рис. 1. Последовательность расчета структурных групп механизма

Важно отметить, что аналитический или численный расчет положения структурной группы механизма требует применения базовых векторных и матричных операций, а также широкого набора геометрических функций на их основе.

Моделирование структурной схемы механизма, разбиение механизма на группы и определение последовательности вычисления структурных групп реализовано в программной системе «Mechanics Studio.NET» [7]. Параллельный расчёт серии положений механизма на графическом процессоре выполняется по данным от системы моделирования.

Параллельный алгоритм вычисления серии N положений механизма. Вычисление серии [1..*N*] положений пространственного рычажного механизма по заданному начальному нулевому положению и входным данным положений первичных механизмов может быть выполнено следующим образом.

Для каждого положения **P** механизма от **1** до **N**.

Последовательно для каждой структурной группы **G** от **1** до **M**:

1) Выполнить расчет положения группы **G**;

2) Для каждого звена ***L*** механизма от **1** до **K**.

Если звену принадлежат точки или оси, положение *P* которых не вычислено, и имеются либо одна точка и две неколлинеарные оси, либо две точки и одна ось, либо три точки с вычисленным положением P, то выполнить пересчёт положения звена из его начального положения.

Ускорить вычисление положений рычажных механизмов можно за счёт применения аналитических методов расчета структурных групп [7], а также за счёт распараллеливания вычисления положений механизма, независимо для всех **P**. В связи с векторным характером вычислений и возможностью применения массового параллелизма по данным SIMD целесообразно использовать современные графические процессоры. Для реализации программы вычисления положений на графической карте была выбрана технология DirectX 11 Compute Shader [2], так как она ориентирована на работу с графическими картами разных

производителей (ATI и NVidia), достаточно близка к железу графических карт и позволяет эффективно задействовать возможности векторных вычислений языка HLSL.

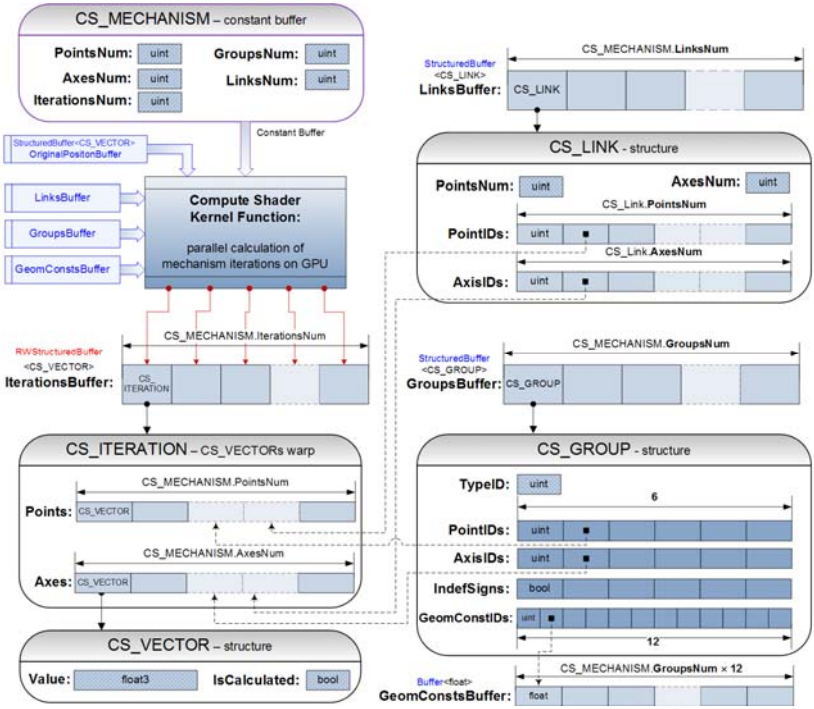


Рис. 2. Структуры данных для вычисления положений механизма на GPU

Структуры данных для расчёта серии положений механизма на GPU. Для расчёта положений механизма на графической карте данные упаковываются в структуры, представленные на рис. 2, передаются на графическую карту и используются программой вычислительного шейдера:

- CS_MECHANISM – константный буфер, хранящий размеры других буферов;
- *LinksBuffer* – буфер звеньев с индексами их точек и осей;

• *GroupsBuffer* – буфер групп с идентификатором типа, индексами точек и осей пар группы, знаками направления осей и индексами геометрических констант;

• *GeomConstsBuffer* – буфер геометрических констант для вычисления положений;

• *OriginalPositionBuffer* – буфер векторов точек и осей исходного положения механизма;

• *IterationsBuffer* – буфер для записи векторов точек и осей итераций механизма (свёртка элементов типа CS_ITERATION в одномерный массив векторов).

Программа вычислительного шейдера расчета положений механизма исполняется на графической карте на параллельных потоках, количество которых равно *числу итераций – IterationsNum*, т.е. числу положений механизма. Каждый поток шейдера независимо вычисляет вектора точек и осей *i*-го положения механизма и записывает их в *IterationsBuffer*. Алгоритм вычисления положений механизма описан блок-схемой на рис. 3. Алгоритм состоит в проходе по списку групп *GroupsBuffer*, последовательном решении задачи о положении для каждой структурной группы и расчёту положений тех звеньев из *LinksBuffer*, которые стало возможно посчитать по точкам и осям, полученным из положений пар группы. Расчёт положения структурной группы выполняется по определённым формулам для каждого типа групп [9] на основе данных из буфера *GroupsBuffer* и геометрических констант из *GeomConstsBuffer*. Расчёт положения звена выполняется путём преобразования его точек и осей из исходного положения *OriginalPositionBuffer*, путём умножения на матрицу поворота-переноса, полученную по изменению 3 известных векторов звена в текущем положении.

Результаты вычислительного эксперимента. Проведён эксперимент по сравнению времени вычисления серии N положений механизма для однопоточной реализации на C++/CLI. NET на CPU Intel Core 2 Quad Q6600 (2.4 GHz) и многопоточной реализации на Compute Shader 5.0 на GPU ATI Radeon HD 5850 (288 SIMD-массивов, 1440 скалярных ядер, 725 MHz). Для вы-

полнения вычислений на графической карте отдельно замерялось время *инициализации и передачи данных* на GPU, включающее упаковку, запись исходных данных в память GPU, загрузку результатов вычисления с GPU и их распаковку.

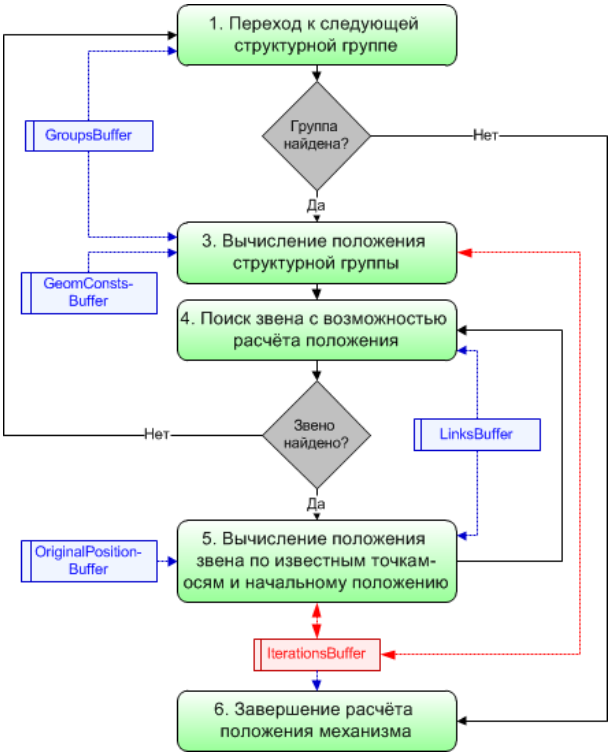
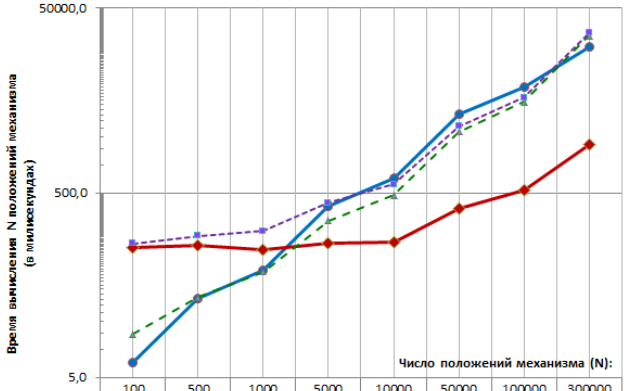


Рис. 3 Алгоритм вычисления положения рычажного механизма на GPU

Результат эксперимента (рис. 4) показывает ускорение вычислений в 100 раз при распараллеливании вычисления положений механизма на GPU, не учитывая инициализацию и передачу данных. С учётом времени инициализации данных на GPU выигрыш по сравнению с последовательными вычислениями на CPU незначителен. Это говорит о необходимости уменьшения объёма пере-

даваемых данных на GPU, возможного за счёт синтеза механизма непосредственно на GPU по параметрам.

Следует отметить что ранее представленные результаты экспериментов в публикациях [1, 3] содержали ошибочные данные о веремени вычислений на GPU. Это связано с недокументированной особенностью работы DirectX: при вызове функции выполнения вычислений Dispatch происходит только добавление этой команды в очередь, а сами вычисления на графической карте выполняются асинхронно, в тот момент, когда требуется их непосредственный результат. Поэтому обыкновенный замер времени вызова Dispatch показывал только постановку этой команды в очередь. Эта проблема была решена за счёт синхронизации GPU-вычислений с CPU при помощи механизма событий DirectX.



	100	500	1000	5000	10000	50000	100000	300000
Время вычисления на CPU (мс.)	7,167	35,804	72,334	358,922	720,008	3515,656	6997,624	18941,451
Время вычисления на GPU (мс.)	127,359	133,659	121,580	143,425	144,741	339,524	542,068	1685,281
Время инициализации GPU (мс.)	14,708	37,222	69,934	248,180	469,918	2302,529	4803,690	25297,309
Время инициализации и вычисления на GPU (мс.)	142,067	170,881	191,514	391,605	614,659	2642,053	5345,758	26982,590

Рис. 4. Время вычисления серии *N* положений механизма на CPU и GPU

Разработанные структуры данных и схема вычислений описывают параллельный алгоритм расчёта серии положений механизма, в котором каждое положение вычисляется независимо. Алгоритм реализован с использованием технологии вы-

числительных шейдеров для выполнения на современных графических процессорах с сотнями векторных вычислительных ядер, что позволяет добиться ускорения вычислений в сотни раз по сравнению с CPU. Высокопроизводительное вычисление положений механизма разработано для дальнейшего применения в решении задачи параметрической оптимизации рычажных механизмов, где для вычисления одного значения целевой функции требуется вычислить серию из десятков или сотен положений конфигурации механизма [6].

Работа выполнена в рамках гранта Президента Российской Федерации для государственной поддержки молодых российских ученых – кандидатов наук (МК-3473.2010.1), а также госконтракта 02.740.11.0839.

Список литературы

1. Городецкий Е.С. Результаты эксперимента по распараллеливанию вычисления серии положений пространственного рычажного механизма на графическом процессоре с помощью DirectX 11 Compute Shader // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2010. – URL: http://www.itlab.unn.ru/archive/ MSConf10/ msconf-2010 _book.pdf. – С. 85–88.

2. Городецкий Е.С. Применение технологии DirectX 11 Compute Shader для вычислений общего назначения на графическом процессоре // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та. – 2010. – С. 81–84.

3. Городецкий Е.С., Турлапов В.Е. Вычисление функций критериев и ограничений в задаче оптимизации пространственного рычажного механизма с распараллеливанием на графическом процессоре // Параллельные вычислительные технологии (ПаВТ’2010): тр. междунар. науч. конф. – Челябинск: ЮУрГУ, 2010. –

URL: http://omega.sp.susu.ac.ru/books/conference/PaVT2010. – С. 454–461.

4. Городецкий Е.С. Реализация параллельного вычисления серии положений пространственного рычажного механизма на графическом процессоре с помощью DirectX 11 Compute Shader // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XI Междунар. конф.-семинара. – Владимир: Изд-во Владимир. гос. ун-та, 2009. – С. 131–135.

5. Городецкий Е.С. Алгоритм параллельного вычисления серии положений пространственного рычажного механизма // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2009. – С. 102–108.

6. Городецкий Е.С. Задача параметрической оптимизации пространственных механизмов с параллельными вычислениями кинематики на графической карте // Высокопроизводительные параллельные вычисления на кластерных системах: материалы Седьмой Междунар. конф.-семинара. – Н.Новгород: Изд-во Нижегород. гос. ун-та, 2007. – С. 117–124.

7. Городецкий Е.С. Система проектирования кинематики пространственных механизмов Mechanics Studio .NET на основе Managed DirectX // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород: Изд-во Нижегород. гос. ун-та, 2007. – С. 48–52.

8. Заблонский К.И., Белоконев И.М., Щекин Б.М. Теория механизмов и машин: учеб. для вузов. – Киев: Выща шк., 1989. – 376 с.

9. Турлапов В.Е. Задача о положениях и классификация одноконтурных структурных групп пространственных рычажных механизмов // Прикладная геометрия. – 2000. Вып. 2. – № 2. – С. 1–22.

Ю.А. Дектерева, Д.В. Зимин, В.Я. Модорский

Пермский государственный технический университет

ИСПОЛЬЗОВАНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ КОМПЬЮТЕРОВ ДЛЯ РАСЧЕТА НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ ФЕРМЕННОЙ ОПРАВКИ ДЛЯ НЕПРЕРЫВНОЙ НАМОТКИ ТРУБ ИЗ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ

В данной работе представлены результаты численных экспериментов по исследованию новой конструкции оправки для высокоскоростной непрерывной намотки труб из композиционного материала. Технологическая особенность намотки стеклопластиковых труб заключается в том, что при намотке на упругую оправку изменяется характер распределения относительного напряжения арматуры, особенно в первых и последних слоях. Подогрев связующего при намотке способствует эффективной фильтрации его по толщине материала уже в процессе самой намотки, а потому уменьшает первоначальное усилие натяжения наполнителя. При намотке холодным связующим рост давления на оправку продолжается вплоть до последних витков.

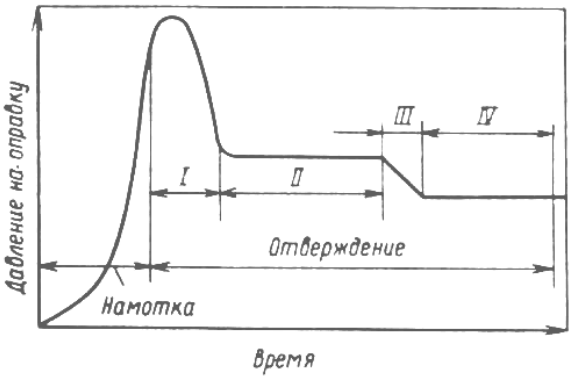


Рис. 1. Характер изменения давления на оправку в процессе намотки и отверждения: I – подъем температуры в печи; II – выдержка в печи; III – охлаждение в печи; IV – выдержка при 293 К

В соответствии с рис. 1 давление на оправку изменяется не только в процессе намотки, но и в процессе отверждения. Особенно интенсивному давлению подвержена оправка в начальный период отверждения, когда действующие давления (до 6,9–7,2 МПа) в 2–4 раза превосходят нагрузки от натяжения нити при намотке. К моменту окончания термообработки давление стеклопластика на оправку падает до 30–60 % от начального и при небольшом исходном натяжении может стать равным нулю.

Таким образом, в исследованиях интересует воздействие внешней нагрузки на оправку, эмитирующей воздействие отверждающейся трубы, вес конструкции оправки и его влияние на форму оправки (рис. 2). Оправка исполняется из стали, следовательно, материал будет иметь большую плотность и вес. Исходя из этого необходимо подобрать такую длину оправки, чтобы не происходил прогиб конструкции.

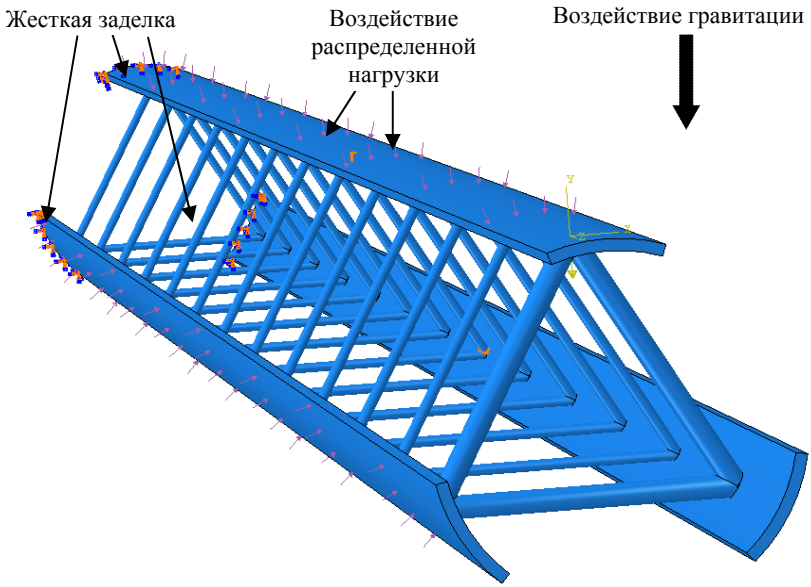


Рис. 2. Твердотельная модель одной секции оправки с нанесенными граничными условиями

На первом этапе исследований проводились расчеты одной секции оправки под собственным весом и подбиралась геометрическая конфигурация с минимальными напряжениями и максимальной длиной. На втором этапе производился расчет оправок с первого этапа под действием силы тяжести и внешней нагрузки. На третьем этапе производился расчет оправки, состоящей из трех секций.

Все расчеты проводились на SMP-системах. Для большинства расчетов одной секции достаточно было использования SMP-системы с процессором Intel Core Two Dou E7500 с оперативной память в размере 8 Гб и кластер ПГТУ; для расчета сборки использовался только кластер ПГТУ (два процессора Barcelona III, 8 Гб оперативной памяти на одном вычислительном узле).

В табл. 1 приведен результат исследования поведения конструкции оправки под действием собственного веса – первый этап вычислений.

Таблица 1

Расчет вариантов конструкции оправки под действием собственного веса

Название	Угол поворота	*L*	*D*	*h*	*r*	*n*	Напряжение σ, МПа	Перемещение *U*, мм
Вар. 1.1	0º	4	0,3	0,005	0,005	0,3	97	15,1
Вар. 1.2	180º	4	0,3	0,005	0,005	0,3	90,2	14,8
Вар. 1.3	0º	6	0,3	0,005	0,005	0,3	189,1	46,9
Вар. 1.4	180º	6	0,3	0,005	0,005	0,3	189	4,69
Вар. 1.5	0º	8	0,3	0,005	0,005	0,3	233,7	79
Вар. 1.6	0º	4	0,3	0,005	0,005	0,6	130,4	20
Вар. 1.7	0º	4	0,3	0,005	0,005	0,9	160	23

Примечание: *L* – длина оправки, *D* – диаметр оправки, *h* – толщина секторной планки, *r* – радиус ферменной конструкции, *n* – расстояние между фермами.

Таким образом, были найдены следующие закономерности:

- максимальные напряжения для оправки с углом поворота 180° незначительно ниже напряжений для оправки с углом поворота 0°;

• максимальные перемещения при повороте оправки изменяются незначительно;

• при удлинении оправки напряжения увеличиваются приблизительно на 70–80 %;

• при прореживании ферм напряжения и перемещения несколько увеличиваются;

• напряжения максимальны в центральных фермах.

В табл. 2 приведен результат исследования поведения конструкции секции оправки под действием нагружающего усилия (давление) размером в 0,6 МПа, приложенного к внешней поверхности секторной планки и собственным весом – второй этап расчетов. Крайняя ферма установлена на расстоянии 10 мм от свободного торца оправки.

Таблица 2

Расчет секции оправки доработанной модели под действием собственного веса и под действием нагружающего усилия (давление) размером в 0,6 МПа

Название	*L*	*D*	*h*	*r*	*n*	Вес (*m*)	Напряжение σ, МПа	Перемещение *U*, мм
Вар. 2.1	4	0,3	0,015	0,01	0,3	164.63	147,5	8,5
Вар. 2.2	4	0,3	0,015	0,005	0,3	145.93	–	–
Вар. 2.3	4	0,3	0,01	0,01	0,3	116.57	175	9,2
Вар. 2.4	4	0,3	0,005	0,01	0,3	71.65	618,1	13,7
Вар. 2.5	4	0,3	0,015	0,01	0,9	148.47	1040	28,9
Вар. 2.6	4	0,3	0,015	0,01	0,6	102.05	424,7	12,9
Вар. 2.7	4	0,3	0,015	0,01	0,6(0,3)	105.81	625,2	11,7
Вар. 2.8	6	0,3	0,015	0,01	0,3	245.16	237	24,5
Вар. 2.9	9	0,3	0,02	0,015	0,3	544.00	165,9	49,6
Вар. 2.10	4	0,6	0,015	0,01	0,3	324.95	223,2	6,7
Вар. 2.11	4	0,9	0,015	0,01	0,3	510.75	227	3,8
Вар.2.12	4	0,3	0,015	0,01	0,3(0,1)	164.63	146,7	8,5

Примечание: L – длина оправки, *D* – диаметр оправки, *h* – толщина секторной планки, *r* – радиус ферменной конструкции, *n* – расстояние между фермами.

Из результатов второго этапа вычислительного эксперимента были сделаны следующие выводы:

- при радиусах стержня фермы, меньших 10 мм, действующие напряжения становятся больше допустимых;
- при толщинах секторной планки, меньших 10 мм, действующие напряжения становятся больше допустимых;
- увеличение расстояния между секторными планками более 0,3 м приводит к значительному увеличению значений максимальных напряжений и перемещений;
- при увеличении длины конструкции до 9 м максимальные напряжения не превышают предел текучести стали;
- при увеличении диаметра конструкции до 0,9 м максимальные напряжения не превышают предел текучести, а максимальные перемещения уменьшаются пропорционально увеличению диаметра.

На третьем этапе рассматривалось воздействие нагружающего усилия (давление) размером в 0,6 МПа, приложенного к внешней поверхности секторной планки. Также учитывался собственный вес конструкции оправки.

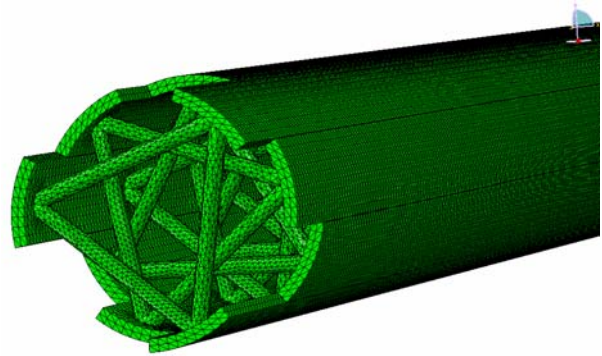


Рис. 3. Оправка из трех секций, разбитая на расчетную сетку

Секции ферменной оправки собирались с отступом 100 мм относительно друг друга. На рис. 3 изображена твердотельная модель трех секций ферменной оправки, разбитая на конечно-элементную сетку в сборке. Угол симметрии конструкции 40°.

Ниже приведен результат расчета трехсекционной оправки с учетом контакта между секторными планками. Длина оправки 4 м, диаметр 0,3 м. Толщина секторной планки 15 мм, радиус стержня 10 мм, расстояние между фермами 0,3 м. Коэффициент трения 0,6.

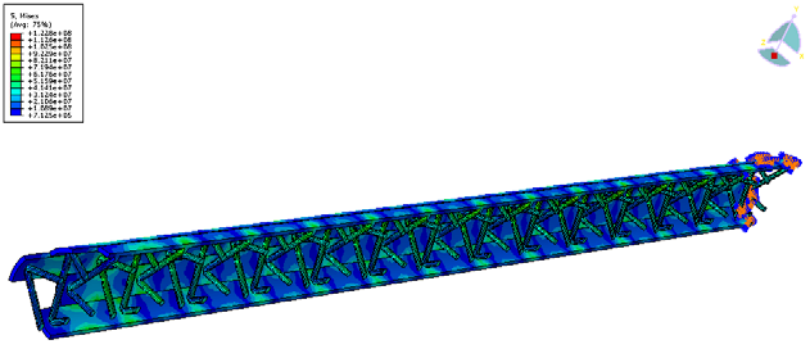


Рис. 4. Распределение полей напряжений, Па. Вид со свободного торца. Разрез

При расчете трехсекционной оправки были найдены следующие закономерности:

- при расчете трех секций конструкции оправки в сборке без учета контакта между секторными планками напряжение падает примерно на 12 %, а перемещение примерно на 15% относительно варианта № 2.1;
- при расчете трех секций конструкции оправки в сборке с учетом контакта между секторными планками и с коэффициентом трения равным 0,6, напряжение падает примерно на 16 %, а перемещение – примерно на 27 % относительно аналогичного варианта без учета контакта.

М.П. Димашова

Нижегородский государственный университет им. Н.И. Лобачевского

РЕАЛИЗАЦИЯ АЛГОРИТМА СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ MEAN SHIFT НА GPU[1]

Сегментация изображения – это разбиение изображения на множество покрывающих его областей [1]. Сегментация изображений является одной из базовых задач компьютерного зрения, поскольку анализ изображения часто начинается с его декомпозиции на области, с которыми связана существенная для данной задачи информация.

Алгоритмы сегментации изображений можно разделить на два класса [2]: интерактивные – использующие пользовательские подсказки и автоматические, – не требующие участия пользователя (однако существуют и другие классификации [3]). Автоматические алгоритмы сегментации, в свою очередь, также делятся на два класса: 1) выделение областей изображения с заданными свойствами, специфичными для конкретной предметной области; 2) разбиение изображения на однородные области. Второй класс алгоритмов автоматической сегментации не использует априорную информацию об изображении, но к свойствам получаемых сегментов предъявляются некоторые требования, такие как однородность текстуры внутри сегмента и непохожесть текстур смежных областей, гладкость границ сегментов и т.п. [1]. Алгоритмы сегментации, разбивающие изображение на однородные области, являются наиболее универсальными, поскольку не ориентированы на определенную предметную область и находят широкое применение в компьютерном зрении при распознавании изображений (медицинских снимков [4], кад-

[1] Работа выполнена при финансовой поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

ров видео автоматизированной сборки деталей и т.п.), поиске стереосоответствий [5] и т.д. Одним из таких автоматических алгоритмов сегментации изображений является алгоритм Mean shift [6].

В данной работе выполнена программная реализация алгоритма сегментации Mean shift на GPU с целью последующего использования в системе анализа медицинских снимков и системе нахождения стереосоответствий. Выбор именно этого алгоритма сегментации среди прочих других [1] связан со следующими фактами:

1. Mean shift алгоритм сегментации имеет математическое обоснование (не является эвристическим).

2. Mean shift алгоритм, относясь к классу алгоритмов кластеризации, не требует задания количества кластеров, в отличие от других алгоритмов кластеризации, применяемых для сегментации изображений (kmeans [1] и т.д.). Это означает, что Mean shift-алгоритм определяет количество сегментов изображения в процессе своей работы.

3. Итеративная обработка пикселей изображения в Mean shift допускает распараллеливание на графических сопроцессорах, так как результат обработки одного пикселя на каждой итерации не зависит от результата обработки остальных пикселей. Большинство алгоритмов сегментации изображений являются вычислительно трудоемкими, но не каждый укладывается в модель массивно-параллельных вычислений.

4. О хорошем «качестве» алгоритма сегментации изображений Mean shift свидетельствуют многочисленные публикации, в которых Mean shift применяется при решении конкретных задач компьютерного зрения [4, 5,7]. Кроме того, стоит отметить, что задача разбиения изображения на однородные области является некорректно поставленной задачей, поскольку одно и то же изображение можно разбить на сегменты разными способами, и нет объективного критерия, позволяющего определить, какое из разбиений является «правильным». Выбор «лучшего» алгоритма зависит от решаемой задачи. Для сравнения

качества методов сегментации существуют базы изображений, для которых известна некоторая «эталонная» сегментация.

Основное внимание в данной работе уделено возможности ускорения алгоритма сегментации Mean shift с использованием графических сопроцессоров. В качестве CPU реализации Mean shift, послужившей основой для дальнейшего распараллеливания на GPU, была взята реализация из популярной библиотеки компьютерного зрения с открытым исходным кодом OpenCV (https://code.ros.org/gf/project/opencv). Вопрос качества реализованного в данной работе алгоритма не рассматривается, качество оценено лишь субъективно (визуально результат совпадает с результатом OpenCV).

Постановка задачи. Алгоритм сегментации Mean shift, как уже упоминалось, относится к классу алгоритмов кластеризации. Задача сегментации изображения формулируется как задача кластеризации.

Дано цветное изображение $I = \{(r,g,b)_i, i = 1,...n\}$ – будем для простоты рассматривать цветовое пространство RGB.

Необходимо разбить заданную выборку (изображение) на кластеры (сегменты) в пространстве признаков, представляющем собой объединение цветового пространства и пространства координат изображения $RGBXY$. В качестве меры близости экземпляров выборки может служить евклидово расстояние в пространстве признаков.

Краткое описание алгоритма. Основная идея Mean shift-метода [4] заключается в том, что по входным данным (изображению) можно построить ядерную оценку для плотности вероятности распределения данных в пространстве признаков $RGBXY$. Далее делается естественное предположение о том, что локальные максимумы плотности вероятности соответствуют центрам кластеров. Из необходимого условия локального экстремума определяется выражение для вектора сдвига $m(p)$ точки пространства признаков $p \in RGBXY$, применяя который к точке p итеративно получаем последовательность точек, схо-

дящуюся к локальному максимуму оценки плотности вероятности (т.е. к центру ближайшего кластера):

$$m(p) = \frac{\sum_{i=1}^{n} p_i g_i}{\sum_{i=1}^{n} g_i} - p, \qquad (1)$$

где $g_i = g(\| (p - p_i) / h \|^2)$, $g(v) = -k'(v)$, h – параметр сглаживания, $K(v) = ck(v)$ – ядро оценки плотности вероятности.

Например, ядро Епанечникова

$$K(v) = c \begin{cases} 1 - \| v \|^2, \| v \| \le 1 \\ 0, \textit{иначе.} \end{cases}$$

Для объединенного пространства $RGBXY$ предполагается, что

$$K((r,g,b,x,y)) = ck((r,g,b))k((x,y)).$$

В данной работе по аналогии с OpenCV используется

$$g_i = \begin{cases} 1, \| (r,g,b) - (r,g,b)_i \| \le h_{RGB} \ \& \ \| (x,y) - (x,y)_i \| \le h_{XY} \\ 0, \textit{иначе.} \end{cases};$$

где h_{RGB} и h_{XY} – параметры сглаживания в соответствующих пространствах.

Теперь можно представить алгоритм сегментации Mean shift. Дано изображение $I = \{(r,g,b)_i, i = 1,...n\}$ и параметры сглаживания h_{RGB} и h_{XY}.

1. Для каждого пикселя $p = (r,g,b;x,y)$ применить итеративно сдвиг

$$p^{(t)} = p^{(t-1)} + m(p^{(t-1)}), \qquad (2)$$

где t – номер итерации, $p^{(0)} = p$. Критерии останова: $t < T$ или $\| m(p^{(t)}) \| < \varepsilon$, ($T$, ε задаются пользователем).

2. Объединить пиксели, «пришедшие» в окрестность одного локального максимума плотности вероятности, в один сегмент. На этом этапе возможна фильтрация сегментов, имеющих маленькую площадь.

Реализация алгоритма. Алгоритм сегментации изображений Mean shift, как следует из описания, состоит из двух частей: итеративный сдвиг пикселей (1-й этап) и выделение связных компонентов (2-й этап).

Первый этап является основным и наиболее вычислительно трудоемким. Именно первый этап алгоритма укладывается в модель массивно-параллельных вычислений и реализован с помощью технологии CUDA на GPU. Входное изображение загружается в 4-канальную текстуру GPU. Остальные входные данные – параметры алгоритма – передаются через разделяемую память. Для хранения выходного изображения (в текущей реализации оно представляет собой изображение исходного размера, содержащее цвета, к которым сошлась процедура итеративного сдвига) используется глобальная память GPU. Обработка каждого пикселя (итеративный сдвиг) выполняется отдельным потоком, т.е. ядро программы представляет собой вычисления по формулам (2) и (1).

Постобработка по выделению связных компонентов выполнена известным рекурсивным алгоритмом Flood fill на CPU.

Результаты. В этом разделе приводится сравнение времени работы выполненной GPU реализации алгоритма Mean shift и его CPU реализации, взятой из библиотеки OpenCV (таблица). Время постобработки по выделению связных компонентов в представленные оценки не входит, поскольку в обеих реализациях она выполняется на CPU и составляет несущественную часть общего времени работы алгоритма.

Вычислительные эксперименты проводились на следующих процессорах:

- CPU – Intel Core i5 750, 2.66 GHz.
- GPU – NVidia GeForce GT 220 (48 ядер) и NVidia GeForce GTX 470 (Fermi, 448 ядер).

Отметим, что CPU реализация Mean shift – однопоточная, т.е. использовалось только одно из четырех ядер Intel Core i5.

Сравнение времени работы Mean shift на CPU и GPU

Размер изображения (пикс.)	Время работы (мс)					Ускорение	
	Intel Core i5	NVidia GT 220 (без i/o данных на GPU)	NVidia GT 220 (с i/o данных на GPU)	NVidia GTX 470 (без i/o данных на GPU)	NVidia GTX 470 (с i/o данных на GPU)	NVidia GT 220 (с i/o данных на GPU)	NVidia GTX 470 (с i/o данных на GPU)
300×225	6895,88	742,577	743,583	80,3896	81,0411	9,273	85,091
700×525	37685,9	3740,96	3744,84	385,637	387,937	10,063	97,144
1000×750	77237,9	8184,33	8190,35	788,693	792,476	9,430	97,464
2000×1500	304973	timed out	timed out	3106,63	3122,19	timed out	97,679
3072×2304	707879	timed out	timed out	7234,29	7264,26	timed out	97,446

Оценка времени работы алгоритма на GPU проводилась с учетом и без учета времени копирования данных с CPU на GPU и обратно. Для NVidia GT 220 не приведено время работы на больших изображениях, поскольку для этого графического сопроцессора существует ограничение на время работы ядра. Далее приведена диаграмма ускорения, полученного на GPU по сравнению с CPU (рис. 1), и диаграмма ускорения, приходящегося на одно ядро GPU (рис. 2).

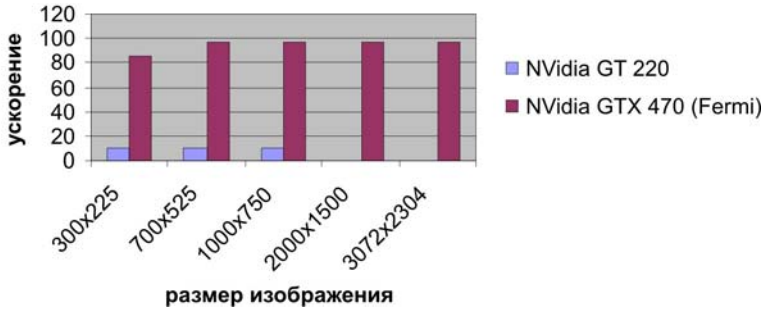


Рис. 1. Ускорение GPU реализации по сравнению с CPU

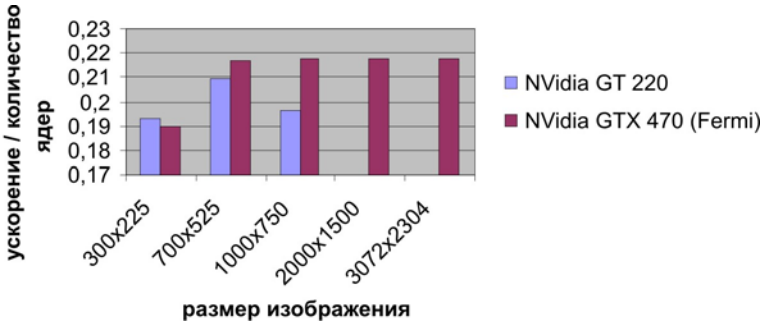


Рис. 2. Ускорение, приходящееся на одно ядро GPU

Ранее алгоритм сегментации изображений Mean shift уже реализовывался на GPU. В работе [6] пишут об ускорении до 134 раз на графическом сопроцессоре NVidia GTX 280 (240 ядер). Однако сравнивать ускорение, полученное в [6] и в данной работе не совсем корректно, поскольку в [6] не описан объем проводимых вычислений (неизвестны параметры алгоритма и его реализация).

Приведено сравнение времени работы последовательной реализации алгоритма сегментации изображений Mean shift, взятой из библиотеки компьютерного зрения OpenCV и выполненной GPU-реализации. Оценка времени работы проводилась на двух видах графических сопроцессоров, включая NVidia Fermi. Современные GPU позволили получить для алгоритма Mean shift ускорение до 98 раз.

Автор благодарен д.т.н. В.Е. Турлапову за обсуждение результатов работы.

Список литературы

1. Шапиро Л., Стокман Дж. Компьютерное зрение. – М.: Бином. Лаборатория знаний, 2006. – 752 с.

2. Баринова О., Вежневец А. Методы сегментации изображений: автоматическая сегментация. – URL: http://cgm.computergraphics. ru/content/view/147.

3. Zhang Y. Advances in Image And Video Segmentation. – USA: IRM Press., 2006.

4. Kim E., Wang W., Li H., Huang X. A parallel annealing methods for automatic color cervigram image segmentation. – URL: http://edwardkim.net/publications/edkim_miccaigpu2009.pdf.

5. Klaus A., Sormann M., Karner K. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. ICPR 2006. – URL: http://old.vrvis.at/2d3d/technology/stereomatching/images/segment-based-stereomatching.pdf.

6. Comaniciu D., Meer P. Mean shift: A robust approach towards feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence, 24: 603–619, 2002.

7. Wang Z., Zheng Z. A region based stereo matching algorithm using cooperative optimization. CVPR, 2008. – URL: http://vision. middlebury.edu/stereo/eval/papers/CORegion.pdf.

Д.Н. Добряев, Н.В. Данилова

Нижегородский государственный университет им. Н.И. Лобачевского

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ГЛОБАЛЬНОГО ПОИСКА С ОБОБЩЁННОЙ ХАРАКТЕРИСТИКОЙ КВАДРАТИЧНОГО ТИПА

Быстрое развитие вычислительных средств, в частности появление возможности распараллеливания, позволяет решать оптимизационные проблемы [1, 2, 3]. Однако наряду с использованием новых высокопроизводительных технологий важное место занимает используемая методика поиска оптимума. Поэтому создание систем, которые помимо эффективного использования вычислительных ресурсов предлагают широкий спектр возможностей выбора метода в зависимости от поставленной задачи, является перспективным направлением на пути получения значимых результатов. А это, в свою очередь, требует создания новых алгоритмов, адаптированных к специфике решаемой задачи.

В настоящей статье предлагается новый параллельный алгоритм глобального поиска, основанный на характеристической структуре решающего правила, в котором используются обобщённые характеристики квадратичного типа. Исследуются условия сходимости метода к глобальному минимуму. Приводятся результаты вычислительного эксперимента по прикладной задаче.

Обобщённая характеристика квадратичного типа и условия сходимости. В данной работе предлагается новый характеристический алгоритм [1] с обобщённой характеристикой квадратичного типа

$$R(i) = \alpha m(x_i - x_{i-1}) + \beta \frac{(z_i - z_{i-1})^2}{m(x_i - x_{i-1})} - \gamma(z_i + z_{i-1}) + \delta|z_i - z_{i-1}|, \quad (1)$$

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \xi \frac{z_t - z_{t-1}}{m}, \quad (2)$$

где $\alpha > 0, \beta, \gamma, \delta, \xi \geq 0$ – параметры. Величина m вычисляется следующим образом:

$m = \begin{cases} rM, M > 0 \\ 1, M = 0 \end{cases}$, где $M = \max\limits_{1 \leq i \leq \tau} \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}}$, а $r > 1$ – параметр метода.

Теорема 1. Пусть точка x^* является предельной точкой последовательности испытаний $\{x^k\}$, порождаемой характеристическим алгоритмом при решении задачи $f(x) \to \inf$ на отрезке $[a,b]$, причем $x^* \neq a$ и $x^* \neq b$. Пусть для параметров выполнено

$$2\sqrt{\alpha\beta} + \delta > \gamma, \beta > \alpha, \xi < \frac{r}{2} \text{ или } 2\sqrt{\alpha\beta} + \delta \geq \gamma, \beta \leq \alpha, \xi < \frac{r}{2}. \quad (3)$$

Тогда последовательность $\{x^k\}$ содержит две подпоследовательности, одна из которых сходится к x^* слева, а другая справа.

Доказательство. Для двусторонней сходимости требуется [2]:

а) если при $k \to \infty$ точка $\overline{x} \in \left[x_{i(k)-1}, x_{i(k)} \right]$ и $x_{i(k)-1} \to \overline{x}, x_{i(k)} \to \overline{x}$, тогда

$$R\left(i\left(k\right)\right) \to -\mu z\left(\overline{x}\right) + c; \tag{4}$$

б) когда, начиная с некоторого шага поиска, интервал $\left(x_{i-1}, x_i\right), i = i\left(k\right)$ не содержит точек поисковых испытаний, справедливо

$$\lim_{k\to\infty} R\left(i\right) > -\mu \min\left\{z\left(x_{i-1}\right), z\left(x_i\right)\right\} + c; \tag{5}$$

в) $$\max\left\{x^{k+1} - x_{t-1}, x_t - x^{k+1}\right\} \le \nu\left(x_t - x_{t-1}\right), \tag{6}$$

где μ, c, ν – некоторые константы, причём $\mu \ge 0, 0 < \nu < 1$.

Найдём параметры $\alpha, \beta, \gamma, \delta, \xi$, при которых выполнены условия теоремы. Пусть функция *z(x)* удовлетворяет условию Липшица

$$\left|z\left(x'\right) - z\left(x''\right)\right| \le L\left|x' - x''\right|, x', x'' \in \left[a, b\right]. \tag{7}$$

а) При стягивании интервала $\left(x_{i-1}, x_i\right)$ к точке $\overline{x}$ характеристика интервала будет стремиться к величине $-2\gamma z\left(\overline{x}\right)$, т.е. для выполнимости (4) положим $\mu = 2\gamma, c = 0$.

б) Пусть для интервала $\left(x_{i-1}, x_i\right)$ справедливо $z_{i-1} = z_i$. Тогда характеристика

$$R(i) = \alpha m(x_i - x_{i-1}) - \gamma\left(z_i + z_{i-1}\right) > -\gamma\left(z_i + z_{i-1}\right) = -2\gamma \min\left\{z_{i-1}, z_i\right\},$$

поскольку, начиная с некоторого шага, $x_i - x_{i-1} > \Delta > 0$.

Предположим теперь, что $z_{i-1} \ne z_i$. Имеем характеристику

$$R(i) = \left|z_i - z_{i-1}\right|\left(\alpha A + \frac{\beta}{A} + \delta\right) - \gamma\left(z_i + z_{i-1}\right),$$

где $A = \dfrac{m(x_i - x_{i-1})}{|z_i - z_{i-1}|}$.

Верно, что $\alpha A + \dfrac{\beta}{A} \geq 2\sqrt{\alpha\beta}$. (8)

Пусть $2\sqrt{\alpha\beta} + \delta > \gamma$, тогда $R(i) > \gamma|z_i - z_{i-1}| - \gamma(z_i + z_{i-1}) =$
$= -2\gamma \min\{z_i, z_{i-1}\}$.

Из выражения для вычисления *m* следует, что

$$A = \frac{m(x_i - x_{i-1})}{|z_i - z_{i-1}|} > 1, \quad (9)$$

поэтому при $\sqrt{\dfrac{\beta}{\alpha}} \leq 1, \beta \leq \alpha$ выполнено $\alpha A + \dfrac{\beta}{A} > 2\sqrt{\alpha\beta}$ всегда и условие (5) верно при нестрогом неравенстве $2\sqrt{\alpha\beta} + \delta \geq \gamma$.

в) $x_t - x^{k+1} = 0,5(x_t - x_{t-1}) + \xi\dfrac{(z_t - z_{t-1})}{m} \leq \left(0,5 + \dfrac{\xi M}{m}\right)(x_t - x_{t-1}) = \left(\dfrac{1}{2} + \dfrac{\xi}{r}\right)(x_t - x_{t-1}).$

$\dfrac{1}{2} + \dfrac{\xi}{r} < 1$, $\dfrac{\xi}{r} < \dfrac{1}{2}$, $\xi < \dfrac{r}{2}$. ***Теорема доказана.***

Теорема 2. Пусть функция *z(x)* удовлетворяет на отрезке $[a,b]$ условию Липшица. Тогда точка x^* глобального минимума функции *z(x)* на данном отрезке является предельной точкой последовательности испытаний, порождаемой алгоритмом с обобщённой характеристикой, если выполнено

$$m > \frac{\gamma L}{\alpha}, \xi < \frac{r}{2}. \quad (10)$$

Доказательство. Для сходимости к глобальному минимуму требуется выполнение условий (4) и (6) теоремы о двусторонней сходимости [2], а вместо (5)

$$\lim_{k\to\infty} R(i) > \frac{\mu}{2}\left(L(x_i - x_{i-1}) - z_{i-1} - z_i\right) + c. \quad (11)$$

Имеем характеристику $R(i) \geq \alpha m(x_i - x_{i-1}) - \gamma\left(z_i + z_{i-1}\right)$, которая должна быть больше $\frac{\mu}{2}\left(L\left(x_i - x_{i-1}\right) - z_{i-1} - z_i\right) + c$. Ранее мы положили, что $m = 2\gamma$, $c = 0$, поэтому должно быть выполнено

$$\alpha m(x_i - x_{i-1}) - \gamma\left(z_i + z_{i-1}\right) > \gamma L\left(x_i - x_{i-1}\right) - \gamma\left(z_i + z_{i-1}\right). \quad (12)$$

Неравенство верно, если

$\alpha m > \gamma L$, т.е. $m > \frac{\gamma}{\alpha} L$. ***Теорема доказана.***

Параллельный алгоритм с обобщённой характеристикой. Распараллеливание решающего правила алгоритма с обобщённой характеристикой заключается в выборе нескольких интервалов (по количеству процессоров) с максимальными характеристиками и независимого вычисления в них значений целевой функции [3, 5]. Пусть имеется $p > 1$ вычислительных узлов.

Шаг 1 (инициализация). Первые $p > 1$ испытаний алгоритма проводятся в произвольных точках $x^1, x^2, \ldots, x^p$ отрезка $[a, b]$. Вычисляются значения $z_i = f\left(x_i\right), 1 < i < p$.

Шаг 2 (начало итерации – формирование системы интервалов). Координаты испытаний $x^1, x^2, \ldots, x^k, k = k\left(n\right)$ перенумеровываются в порядке возрастания:

$$a = x_0 < x_1 < \ldots < x_{\tau-1} < x_\tau = b, \quad (13)$$

где $\tau = k\left(n\right) - 1$.

Шаг 3 (вычисление характеристик интервалов). Каждому интервалу $(x_{i-1}, x_i), 1 \leq i \leq \tau$, ставится в соответствие число (характеристика)

$$R(i) = \alpha m(x_i - x_{i-1}) + \beta \frac{\left(z_i - z_{i-1}\right)^2}{m\left(x_i - x_{i-1}\right)} - \gamma\left(z_i + z_{i-1}\right) + \delta\left|z_i - z_{i-1}\right|.$$

Шаг 4 (выбор интервалов с p максимальными характеристиками). Среди характеристик $R(i)$ выбираются p максимальных и соответствующие им интервалы.

Шаг 5 (формирование координат новых испытаний). В выбранных интервалах формируются точки испытаний $x^{k+s} = \frac{x_{t_s} + x_{t_s-1}}{2} - \xi \frac{z_{t_s} - z_{t_s-1}}{m}$ для $1 \le s \le p$.

Шаг 6 (проверка условия остановки). Если выполняется условие остановки

$$\min_{1 \le s \le p} \left(x_{t_s} - x_{t_s-1} \right) \le \varepsilon, \qquad (14)$$

то вычисления заканчиваются. В противном случае осуществляется вычисление значений функции в точках $x^{k+1}, \ldots, x^{k+p}$ – каждое испытание на своём процессоре.

Из [5] следует, что условия сходимости метода с обобщённой характеристикой совпадают с условиями сходимости его последовательного прототипа.

В системе ***PSoDI*** [4] реализована возможность использования параллельного алгоритма глобального поиска с обобщённой характеристикой квадратичного типа для решения многомерных многоэкстремальных задач оптимизации в рамках многошаговой схемы на многоядерных архитектурах.

Пример использования предложенного параллельного алгоритма при решении задачи стационарного потокораспределения в гидросистеме. Спектр применения методов оптимизации очень широк. Одной из задач, решение которых можно свести к поиску экстремума функции, является задача нахождения стационарного потокораспределения в гидросистемах [6].

Применение данной методики продемонстрировано на примере решения задачи нахождения равновесных режимов системы циркуляции теплоносителя (СЦТ) ядерной энергетиче-

ской установки, являющейся частным видом гидросистемы. В силу большой размерности и многоэкстремальности подобных задач целесообразно применять параллельные методы поиска оптимума.

На рисунке представлена расчетная схема СЦТ, соответствующая установке ВВЭР-440 и содержащая 6 параллельных подключенных к реактору петель циркуляции.

Прошедший через реактор *1* нагретый теплоноситель поступает по трубопроводам в несколько параллельных петель *2* с теплообменниками *3*, где он охлаждается и, пройдя насосы *4*, снабженные электродвигателями *5*, возвращается в реактор. Стрелками показано направление течения теплоносителя. Нарушение нормальной работы СЦТ недопустимо с точки зрения безопасности реактора.

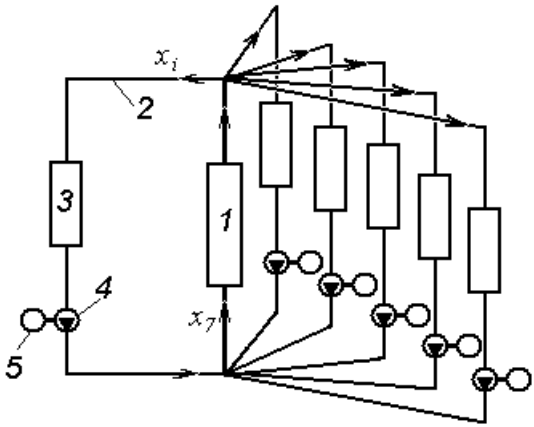


Рис. Расчетная схема СЦТ

Состояния равновесия данной системы являются стационарными точками соответствующей функции Ляпунова [7]. Для рассматриваемой СЦТ задача нахождения устойчивых состояний равновесия в шестимерном фазовом пространстве сводится к поиску минимумов этой функции.

Минимизируемая функция для данной схемы имеет вид [6]

$$f(x)=\sum_{i=1}^{6} S_i(x_i)+S_7(x),$$

здесь $S_i(x_i)=\begin{cases}\dfrac{a_{1i}x_i^3}{3}-\dfrac{b_i\omega_i x_i}{2}-c_i\omega_i^2 x_i, & x_i\ge 0;\\ -\dfrac{a_{2i}x_i^3}{3}-\dfrac{b_i\omega_i x_i}{2}-c_i\omega_i^2 x_i, & x_i<0;\end{cases}$, $i=\overline{1,6}$

$$S_7(x)=\begin{cases}\dfrac{d_1}{3}\left(\sum\limits_{j=1}^{6}x_j\right)^3, & \sum\limits_{j=1}^{6}x_j\ge 0;\\ -\dfrac{d_2}{3}\left(\sum\limits_{j=1}^{6}x_j\right)^3, & \sum\limits_{j=1}^{6}x_j<0,\end{cases}$$

где x_i – расход теплоносителя в i-й петле, $i=\overline{1,6}$; функции $S_i(x_i)$, $i=\overline{1,6}$, представляют собой интегралы от гидравлических характеристик участков с теплообменниками, $S_7(x)$ – интеграл от гидравлической характеристики участка активной зоны; a_{1i}, a_{2i}, b_i, c_i, d_1, d_2 – положительные коэффициенты, величины которых зависят от гидравлических сопротивлений участков и гидравлических характеристик насосов; ω_i – угловая скорость вращения ГЦН в i-й петле.

Расчеты проводились при следующих заданных параметрах:

$\omega_1=\omega_2=\omega_3=\omega_4=\omega_5=\omega_6=1; a_{1i}=1,4; a_{2i}=4; b_i=4; c_i=6;$
$d_1=d_2=0,03; i=\overline{1,6}.$

Поиск минимумов шестимерной функции $f(x)$ был проведен в системе ***PSoDI*** с использованием параллельного алгоритма с обобщённой характеристикой в рамках многошаговой схемы редукции размерности на восьми ядрах (2 четырёхядерных процессора *Intel(R) Xeon(R) CPU E5504 @ 2.0GHz*). Были найдены следующие устойчивые состояния, совпадающие с экстремумами:

$x_1 = -0,90; x_2 = 2.93; x_3 = 2,93; x_4 = 2,93; x_5 = 2,93; x_6 = 2,93.$

$f_{\min} = -84,30.$

$x_1 = 2,93; x_2 = -0,90; x_3 = 2,93; x_4 = 2,93; x_5 = 2,93; x_6 = 2,93.$

$f_{\min} = -84,30.$

$x_1 = 2,93; x_2 = 2,93; x_3 = -0,90; x_4 = 2,93; x_5 = 2,93; x_6 = 2,93.$

$f_{\min} = -84,30.$

$x_1 = 2,93; x_2 = 2,93; x_3 = 2,93; x_4 = -0,90; x_5 = 2,93; x_6 = 2,93.$

$f_{\min} = -84,30.$

$x_1 = 2,93; x_2 = 2,93; x_3 = 2,93; x_4 = 2,93; x_5 = -0,90; x_6 = 2,93.$

$f_{\min} = -84,30.$

$x_1 = 2,93; x_2 = 2,93; x_3 = 2,93; x_4 = 2,93; x_5 = 2,93; x_6 = -0,90.$

$f_{\min} = -84,30.$

$x_1 = 2,55; x_2 = 2,55; x_3 = 2,55; x_4 = 2,55; x_5 = 2,55; x_6 = 2,55.$

$f_{\min} = -87,58.$

Глобальный минимум этой функции соответствует равновесному режиму циркуляции. Наличие других режимов работы, соответствующих остальным минимумам функции $f(x)$, недопустимо с точки зрения безопасности работы, и задача конструктора состоит в выборе гидравлических характеристик, при которых состояние равновесия минимально и устойчиво «в целом». Проверка полученного решения, основанная на результатах работ [6], показала правильность полученного решения.

В настоящей работе предложен новый алгоритм глобального поиска, основанный на характеристической структуре решающего правила, в котором используются обобщённые характеристики квадратичного типа. Найдены условия сходимости метода к глобальному минимуму. Представлены результаты решения прикладной задачи по нахождению стационарного потокораспределения в гидросистеме с использованием параллельного алгоритма на многоядерной системе.

Список литературы

1. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-Convex Constraints Sequential and Parallel Algorithms – Dordrecht. Kluwer Academic Publishers, the Netherlands, 2000. – 728 p.

2. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация: учеб. пособие. – Н. Новгород: Изд-во ННГУ, 2007. – 489 с.

3. Sergeyev Ya.D., Grishagin V.A. Parallel asynchronous global search and the nested optimization scheme // Journal of Computational Analysis & Applications. 2001. – 3(2). – P. 123–145.

4. Добряев Д.Н., Гришагин В.А. Программная система конструирования и исследования характеристических методов многоэкстремальной оптимизации // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород, 3–4 апреля, 2007. – 68–71 с.

5. Strongin R.G., Sergeyev Ya.D., Grishagin V.A. Parallel Characteristical Algorithms for Solving Problems of Global Optimization // Journal of Global Optimization. 1997. – No. 10. – P. 185–206.

6. Смирнов Л.В., Данилова Н.В. Основы прикладной аналитической гидромеханики напорного течения несжимаемой жидкости: учеб.-метод. пособие. – Н. Новгород: Изд-во Нижегородского гос. ун-та, 2009. – 65 с.

7. Применение прикладной аналитической гидромеханики и методов принятия оптимальных решений в задаче нахождения потокораспределения в гидросистемах / Л.В. Смирнов [и др.] // Вестн. ННГУ. Серия «Матем. моделирование. Опт. управление». – 2010. – № 2(1). – С. 144–154.

П.Н. Дружков, Н.Ю. Золотых, И.Б. Мееров, А.Н. Половинкин

Нижегородский государственный университет им. Н.И. Лобачевского

РЕАЛИЗАЦИЯ АЛГОРИТМА ГРАДИЕНТНОГО БУСТИНГА ДЕРЕВЬЕВ РЕШЕНИЙ

Машинное обучение является подразделом весьма обширной области науки, изучающей искусственный интеллект. Алгоритмы машинного обучения приходят на помощь при решении задач, для которых сложно или невозможно придумать явный алгоритм решения, и, следовательно, практические сферы их применения огромны: от прогнозирования погоды, экономических и социальных процессов, медицинской диагностики до детектирования объектов на фото или видео, распознавания текста, речи, создания антивирусных программ и алгоритмов фильтрации рекламы и спама.

Авторами данной работы была выполнена программная реализация одного из наиболее перспективных алгоритмов обучения с учителем – алгоритма градиентного бустинга деревьев решений (GBT – gradient boosting trees) [1, 2]. Насколько известно авторам, это первая открытая C/C++ реализация данного метода. Мы приводим результаты экспериментов, проведенных с использованием этой реализации на наборах реальных данных, взятых из репозитория UCI (UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml). Результаты вычислительного эксперимента свидетельствуют о конкурентоспособности предлагаемой реализации по сравнению с реализациями других алгоритмов. Разработка ведется в рамках популярной библиотеки компьютерного зрения с открытым исходным кодом OpenCV (OpenCV. URL: http://opencv.willowgarage.com.).

Постановка задачи. Одной из задач, изучаемых в машинном обучении, является *задача обучения с учителем*. В рамках этой задачи дано некоторое множество *объектов X*. Каждому объекту $x \in X$ поставлена в соответствие величина *y*, называемая *выходом*, или *ответом*, и принадлежащая множеству

допустимых ответов *Y*. Упорядоченная пара «объект–ответ» (x, y), где $x \in X, Y \in y$, называется *прецедентом*. Требуется восстановить зависимость между входом и выходом, основываясь на данных о конечном наборе прецедентов, называемом *обучающей выборкой*: $\left\{(x_i, y_i) \middle| x_i \in X, y_i \in Y, i = \overline{1, N}\right\}$. Другими словами, задача состоит в построении функции *f* из некоторого множества *K*, которая, получив на вход *x*, предсказала бы значение ответа *y* как можно точнее. Процесс нахождения *f* называется *обучением* или *настройкой* модели. В случае конечного *Y* говорят о *задаче классификации*, если $Y = \mathbf{R}$ – *о задаче восстановления регрессии* [3].

Метод решения. Один из общих подходов решения задач обучения заключается в комбинировании моделей. Две основные конкурирующие идеи данного подхода – бэггинг (*bagging* от *Bootstrap Aggregating*) [4] и бустинг (*boosting*) [5]. Первая из них состоит в построении множества независимых (между собой) моделей с дальнейшим принятием решения путем голосования в случае задачи классификации и усреднения в случае регрессии. Данный подход реализован в алгоритме случайных деревьев [6] (*random trees* или *random forest*). Бустинг, в противоположность бэггингу, обучает каждую следующую модель с использованием данных об ошибках предыдущих моделей.

Алгоритм градиентного бустинга деревьев решений является развитием бустинг-идеи. Он позволяет строить аддитивную функцию в виде суммы деревьев решений итерационно по аналогии с методом градиентного спуска. Данный подход позволяет расширить круг решаемых этим алгоритмом задач, а также зачастую получить выигрыш в точности предсказания.

Экспериментальные результаты. В данном разделе приведены некоторые экспериментальные результаты, показывающие достоинства и недостатки метода градиентного бустинга. Наряду с подходом, которому посвящена данная работа, были рассмотрены и конкурирующие алгоритмы: одиночные дере-

вья решений (алгоритм CART [7]), случайные деревья (случайные леса) [6], машина опорных векторов [8]. Программной основой проведенных экспериментов является открытая библиотека компьютерного зрения OpenCV: все результаты, относящиеся к конкурирующим алгоритмам, были получены непосредственно с помощью ее компонентов: CvDTree, CvRTrees, CvERTrees и CvSVM.

Эксперименты проводились на наборах реальных данных, взятых из репозитория UCI. Их краткие характеристики приведены в табл. 1.

Таблица 1

Тестовые наборы данных

Название	Общее количество прецедентов	Число переменных (количественные/ номинальные)	Количество классов
Восстановление регрессии			
auto-mpg	398	7 (4/3)	—
Computer hardware	209	8 (7/1)	—
Concrete slump	103	9 (9/0)	—
Forestfires	517	12 (10/2)	—
Boston housing	506	13 (13/0)	—
imports-85	201	25 (14/11)	—
Servo	167	4 (0/4)	—
Abalone	4177	8 (7/1)	—
Классификация			
Agaricus lepiota	8124	22 (0/22)	2
Liver disorders	345	6 (6/0)	2
Car evaluation	1728	6 (0/6)	4

Сравнение различных алгоритмов машинного обучения производилось по результатам 10-кратного скользящего контроля, с помощью которого осуществлялся выбор наилучших значений параметров для каждой конкретной задачи и каждого алгоритма. Тестовая ошибка считалась при помощи нескольких критериев:

1) средняя абсолютная ошибка (*average-absolute-error*):

$$\mathrm{Err}_{\mathrm{rms}} = \frac{1}{10}\sum_{k-1}^{10}\frac{\sum_{i=1}^{T_k}\left|y_{jk,i} - f(x_{jk,i})\right|}{T_k},$$

где T_k – объем k-й тестовой выборки, а $(x_{jk,i}, y_{jk,i})$ – i-й прецедент k-й тестовой выборки;

2) корень среднеквадратичной ошибки (*root-mean-squared error*):

$$\mathrm{Err}_{\mathrm{abs}} = \frac{1}{10}\sum_{k-1}^{10}\sqrt{\frac{\sum_{i=1}^{T_k}\left(y_{jk,i} - f(x_{jk,i})\right)^2}{T_k}};$$

3) для задачи классификации велся подсчет частоты неправильной классификации прецедентов тестовой выборки:

$$\mathrm{Err}_{\mathrm{misclass}} = \frac{1}{10}\sum_{k-1}^{10}\frac{\sum_{i=1}^{T_k}\left(y_{jk,i} \neq f(x_{jk,i})\right)}{T_k}.$$

Прецеденты с пропущенными значениями удалялись из обучающей и тестовой выборок при использовании CvSVM.

Наименьшие из полученных описанным способом ошибок приведены в табл. 2 и 3. Из этих данных видно, что алгоритм градиентного бустинга, как правило, дает результат, близкий к наилучшему, для конкретной задачи, что подтверждает его универсальность и способность подстраиваться под специфику решаемой задачи. В то же время для некоторых из рассматриваемых задач существуют алгоритмы, дающие меньшую тестовую ошибку.

Для некоторых из перечисленных в табл. 1 наборов данных были проведены и другие эксперименты. Данные случайным образом разбивались на обучающую и тестовую выборки в соотношении 9:1 для наборов Boston housing, Computer hardware и Servo, и 8:2 для Abalone, после чего выполнялось обучение и предсказание. Процесс повторялся 100 раз для каждой задачи. Данный подход также позволяет проследить влияние значений некоторых параметров на обобщающую способность метода и служит методом для сравнения различных алгоритмов.

Таблица 2

Корни среднеквадратических ошибок (RMS) и средние абсолютные ошибки (ABS), полученные различными алгоритмами при 10-кратном перекрестном контроле

Название	Градиентный бустинг (GBT)		Дерево решений (CvDTree)		Случайные деревья (CvRTrees)		Случайные деревья (CvERTrees)		Машина опорных векторов (CvSVM)	
	RMS	ABS	RMS	ABS	RMS	ABS	RMS	ABS	RMS	ABS
auto-mpg	2,682	2	3,133	2,238	2,653	1,879	2,955	2,147	4,042	2,981
Computer hardware	23,55	12,62	30,13	15,62	26,02	11,62	19,12	9,631	50,51	37
Concrete slump	2,524	2,257	3,727	2,923	3,193	2,6	2,945	2,359	2,164	1,767
Forestfires	35,15	18,74	38,09	17,26	35,22	17,79	34	16,64	45,51	12,9
Boston housing	2,914	2,033	3,653	2,602	3,042	2,135	3,127	2,196	5,71	4,049
imports-85	1827	1306	2317	1649	1821	1290	2146	1487	2583	1787
Servo	0,385	0,238	0,455	0,258	0,418	0,247	0,686	0,42	0,884	0,655
Abalone	2,144	1,47	2,281	1,604	2,115	1,492	2,124	1,498	2,644	2,091

Таблица 3

Средние частоты неправильной классификации прецедентов, полученные различными алгоритмами при 10-кратном перекрестном контроле

Название	Градиентный бустинг (GBT)	Дерево решений (CvDTree)	Случайные деревья (CvRTrees)	Случайные деревья (CvERTrees)	Машина опорных векторов (CvSVM)
Agaricus lepiota	0	0,000123	0	0	0
Liver disorders	0,251357	0,30543	0,227828	0,254299	0,278582
Car evaluation	0	0,0513824	0,0364987	0,0394574	0,0509819

Средние значения ошибок, полученных в результате этих экспериментов, приведены в табл. 4. На рассмотренных нами

наборах данных градиентный бустинг в большинстве случаев превосходит метод случайных деревьев или показывает сравнимый результат.

Таблица 4

Сравнение средних ошибок алгоритмов градиентного бустинга и случайных деревьев

Набор данных	Тестовая ошибка алгоритма градиентного бустинга		Тестовая ошибка алгоритма случайных деревьев	
	Средняя абсолютная ошибка	Средняя квадратичная ошибка	Средняя абсолютная ошибка	Средняя квадратичная ошибка
Boston housing	1,995	8,234	2,13	9,689
Computer hardware	10,29	1096,1	13,996	2047,8
Servo	0,198	0,237	0,24	0,257
Abalone	1,517	4,732	1,514	4,653

На рисунке изображены бокс-диаграммы, которые показывают разброс результатов эксперимента на наборе данных Servo. Слева находятся диаграммы, соответствующие алгоритму градиентного бустинга с использованием деревьев решений различных размеров: глубина от 1 до 4. Самая правая бокс-диаграмма соответствует методу случайных деревьев. Таким образом, можно наблюдать снижение тестовой ошибки бустинга при увеличении глубины используемых деревьев. Также эта диаграмма иллюстрирует небольшое превосходство алгоритма градиентного бустинга над случайными деревьями на рассматриваемом наборе данных.

Данные экспериментов свидетельствуют о конкурентоспособности программной реализации, выполненной авторами данной работы. Текущая реализация интегрирована в библиотеку с открытыми исходными кодами OpenCV, что, во-первых, дает возможность легко проводить сравнение качества работы различных алгоритмов машинного обучения для конкретных прикладных задач, во-вторых, предоставляет удобные средства для применения алгоритма градиентного бустинга деревьев решений в задачах компьютерного зрения.

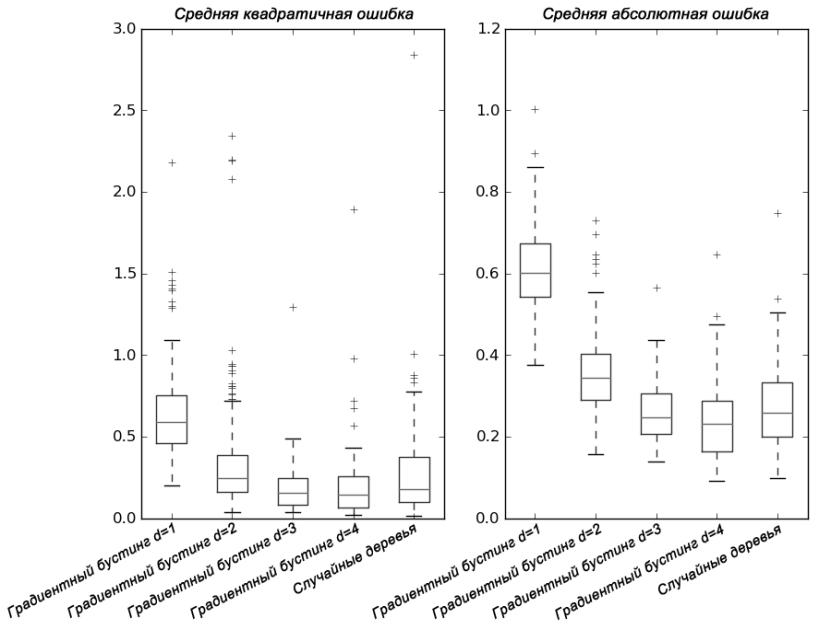


Рис. Бокс-диаграммы для результатов тестирования алгоритма градиентного бустинга с использованием деревьев решений различной глубины и метода случайных деревьев на наборе данных Servo

Мы благодарим В.Л. Ерухимова (компания ITSeez) за ценные замечания и постоянное внимание к работе.

Работа выполнена при поддержке федеральной целевой программы «Научные и научно-педагогические кадры инновационной России», госконтракт 02.740.11.5131.

Список литературы

1. Friedman J.H. Greedy Function Approximation: a Gradient Boosting Machine. Technical report. Dept. of Statistics. – Stanford University, 1999.

2. Friedman J.H. Stochastic Gradient Boosting. Technical report. Dept. of Statistics. – Stanford University, 1999.

3. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2008.

4. Breiman L. Bagging predictors // Machine Learning. – 1996. – Vol. 26, No. 2. – P. 123–140.

5. Freund Y., Schapire R. Experiments with a New Boosting Algorithm // Machine Learning: Proceedings of the Thirteenth International Conference, 1996.

6. Breiman L. Random Forests // Mach. Learn. – 2001. – Vol. 45, No. 1. – P. 5–32.

7. Breiman L., Friedman J., Olshen R., Stone C. Classification and Regression Trees. – Wadsworth, 1983.

8. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. – М.: Наука, 1979. – 448 с.

О.В. Джосан

Московский государственный университет имени М.В. Ломоносова

О ПЕРСПЕКТИВАХ И ПРОБЛЕМАХ ЭКЗАФЛОПСНЫХ ВЫЧИСЛЕНИЙ В РОССИИ

Осенью 2009 года, во время проведения конференции «Научный сервис в сети Интернет–2009» [1] группой энтузиастов было зарегистрировано доменное имя exascale.ru – российский аналог домена exascale.org международного сообщества International Exascale Software Project [2]. Сделано это было в рамках вечной проблемы «научных отцов и детей» – молодежь требует революции, а старшее поколение – соблюдения традиций. Молодым хотелось доказать, что те проблемы, которые обсуждают «отцы», устарели и потеряли актуальность, необходимо выходить на новый уровень и думать о препятствиях, которые нам обещают к 2018 году. Иначе шансов быть на достойном месте в мировом суперкомпьютерном сообществе у России немного. Однако широкой поддержки тогда эта тема не получила и была в целом признана несвоевременной.

За прошедший год в России многое изменилось: для широкой общественности стал доступен «полпетафлопса» [3]. В обсуждение ситуации с «пето» и «экза» вовлекалось все больше и больше ученых из разных организаций и сфер научной деятельности. На российских суперкомпьютерных конференциях появились доклады о масштабируемости на тысячи ядер, моделировании задач, которые раньше считались нерешаемыми, и т.д. Перспектива «экза» стала уже не столь далекой и абстрактной. Нужно ли «экза» в России к 2018 году? Есть ли перспективы его достичь? Какие ресурсы существуют для этого в стране? Появятся ли новые парадигмы программирования или принципиальные подходы к построению вычислительных устройств? Вопросов очень много. На большинство из них пока нет ответов, но уже пришло время формулировать проблемы, которые требуют решения.

В рамках международной конференции «Научный сервис в сети Интернет–2010» прошло первое заседание Российского сообщества экзакомпьютерных технологий [4], посвященное обсуждению актуальных проблем экзавычислений в России и мире. Данная статья подводит итоги обсуждения и рассматривает следующие вопросы:

- проблемы подготовки кадров для работы с экзавычислителями;
- актуальные задачи, требующие расчетов на сверхбольших компьютерах;
- проблемы создания программного обеспечения для экзакомпьютеров.

Приведенный список проблем никак не может считаться исчерпывающим и формулирующим основные проблемы перехода к «экзаскейлу», однако представляет практический интерес и позволяет сформулировать «руководство к действию» на ближайшую перспективу для российских исследователей.

Автор выражает благодарность всем участникам первого заседания сообщества: Лацису Алексею (ИПМ РАН), Дбар Светлане (ИПМ РАН), Орлову Антону (ИПС РАН), Андрею Адинцу (НИВЦ МГУ), Шворину Артему (ИПС РАН), Климову Юрию (ИПМ РАН), Сальникову Алексею (ВМК МГУ), Андрее-

ву Дмитрию (ВМК МГУ), Серебрийскому Артему (ВМК МГУ), Коробкову Сергею (ВМК МГУ), Позднееву Александру (ВМК МГУ), Коржу Антону (Т-платформы), ДавыдовуАлександру (ИПМ РАН), Андрюшину Дмитрию (НИЦЭВТ).

Проблема подготовки кадров для работы с экзавычислителями. Проблема образования в России в последнее время обсуждается с разных сторон: почему в школах стали хуже учить, почему все ПТУ стали вузами, почему в вузах учат тому же, что 20 лет назад? Вопрос обучения на разных стадиях весьма многогранный и острый для нашей страны. В области суперкомпьютерного обучения мы безнадежно отстаем от развитых стран. К сожалению, точной статистики нет, но слово «суперкомпьютер» знают далеко не все россияне. Даже в Московском университете, где установлен тринадцатый по мощности в мире суперкомпьютер[3].

Суперкомпьютерное образование активно развивается в рамках Суперкомпьютерного консорциума университетов России [5]. Основная деятельность в области суперкомпьютерного образования связана с обучением в вузах. Однако экзасистемы требуют не только высококвалифицированных специалистов, которые их разрабатывают, и ученых, которые их используют для решения прикладных задач, но и обслуживающего персонала.

К сожалению, далеко не все процессы в настоящее время удается автоматизировать, поэтому появляется потребность в таких участниках суперкомпьютерного процесса, как, например, операторы, которые следят за показанием датчиков системы и вовремя реагируют на сбои. Также для обслуживания компьютера необходим персонал, который будет оперативно менять вышедшие из строя компоненты. На системах пета- и тем более экза-уровня регулярно выходящие из строя компоненты – это обыденная ситуация. Говорят, что в ближайшем будущем каждому суперкомпьютеру нужен будет водопроводчик или сантехник. Сейчас это воспринимается скорее как шутка, однако с повсеместным внедрением водяного охлаждения разработчикам суперкомпьютеров вплотную приходится общаться со специалистами в этой области.

Для обслуживания любой информационной системы традиционно требуется человек под кодовым названием «системный администратор» или «эникейщик» – человек, который пытается наладить взаимодействие пользователей с системой. В больших суперкомпьютерных системах функциональность системного администратора подразделяется на несколько специальностей. Например, установка различных прикладных программных пакетов на системы петафлопсного уровня уже вызывает огромное количество проблем. Зачастую прикладной исследователь не обладает компетенцией, чтобы такие проблемы решать, а системному администратору не до этого: потратить пару недель на «допиливание пакета» – это непозволительная роскошь. В системах экзафлопсного уровня такие задачи придется решать регулярно. Поэтому необходимо готовить соответствующих специалистов.

В настоящее время в России не существует многоуровневой системы образования. Считается, что нужно либо получить высшее образование, либо обходиться без образования совсем. Перспектива «экза» может послужить фактором развития образования в суперкомпьютерной области по схеме многоуровневости: когда в отрасли требуются не только специалисты с высшим образованием, но и люди, которым достаточно среднеспециального образования или краткосрочных курсов повышения квалификации.

Актуальные задачи для экзавычислителей. Перспектива появления вычислительных систем экзауровня ставит суперкомпьютерное сообщество перед проблемой определения круга задач, которые можно будет решить с появлением систем такого уровня. Многие специалисты задаются вопросом, а нужен ли будет в 2018 году тот самый экзакомпьютер или это создание «сферического коня в вакууме»? На сегодняшний день человечество смогло придумать всего несколько реальных задач, которые для своего решения требуют экзавычислителя и не могут быть решены на сегодняшних суперкомпьютерах.

Одной из основных целей сообщества International Exascale Software Project была формулировка списка задач, которые

требуют экзавычислений для своего решения. Интересна проекция этого списка на российское суперкомпьютерное сообщество. Какие перспективные направления исследований в России могут потребовать экзафлопсных вычислений?

Одним из таких направлений, безусловно, станет более точное предсказание погоды. В России в настоящее время ведутся подобные исследования. Еще одной важной задачей, которая решается в нашей стране и требует значительных вычислений, является моделирование лекарственных препаратов. Также важной отраслью для нашей страны является нефтегазовая отрасль, в которой суперкомпьютерное моделирование применяется для существенного сокращения стоимости разработки.

Антон Корж предположил, что экзокомпьютер нужен для того, чтобы стать последним компьютером на старых принципах: основная задача, которую решит экзакомпьютер – это создание компьютера на новых принципах, возможно квантового.

Проблема формулировки задач для суперкомпьютера экзауровня сводится к анализу междисциплинарных исследований, которые проводятся с использованием суперкомпьютеров. В настоящее время в мире идет активное формирование междисциплинарных исследовательских команд, в том числе включающих специалистов по суперкомпьютерному моделированию. Такое объединение усилий позволяет решать задачи, которые раньше казались неразрешимыми. Для этой цели создаются специализированные междисциплинарные исследовательские центры, где совместно работают исследователи из разных областей. Для России эта тенденция не характерна. Междисциплинарные работы проводятся узкими группами, часто для иностранных заказчиков, поэтому решаемые задачи не отличаются масштабностью. Для широкого распространения междисциплинарных исследований прежде всего необходимо обучение прикладных пользователей основам суперкомпьютерных вычислений и развитие коммуникаций специалистов в суперкомпьютерной области и прикладных пользователей.

Проблема создания программного обеспечения для экзакомпьютеров. Сложно предсказать, какая парадигма про-

граммирования будет актуальна через 8 лет. Однако можно попытаться сформулировать проблемы, которые эта парадигма будет решать. Вероятно, что основными проблемами программистов, пишущих программы для экзакомпьютеров, станет ввод-вывод данных и масштабируемость программ.

Проблемы ввода-вывода возникают и в большинстве приложений, которые сейчас выполняются на петафлопсных суперкомпьютерах. Неравномерность коммуникаций приводит к появлению простоев вычислительных узлов и замедлению общего процесса вычислений. Возможны два подхода к решению проблемы ввода-вывода: использование специализированных библиотек для оптимизации ввода-вывода в программе или использование более совершенных аппаратных конфигураций, позволяющих оптимизировать ввод-вывод. Оба подхода обладают классическими недостатками: первый слишком сложен для прикладного пользователя, второй существенно увеличивает стоимость установки. Возможным решением проблемы ввода-вывода в больших системах может стать новая концепция программирования, в рамках которой накладывается строгое ограничение, требующее разделения параллельной программы на части «ввод-вывод» и «счет». Имея структуру программы с разделением частей программы на эти две категории, можно существенно оптимизировать и балансировать загруженность вычислительных узлов.

Масштабируемость задач на системе с миллионом вычислительных ядер – это проблема, которую уже приходится решать программистам. К сожалению, большинство прикладных пакетов для суперкомпьютерного моделирования имеют существенные ограничения на масштабируемость, т.е. на максимальное количество процессоров, для которых будет получаться заметное ускорение. Одним из решений этой проблемы может стать переход к программированию с использованием односторонних коммуникаций. Ярко выраженная проблема парадигмы односторонних коммуникаций – сложность в понимании и применении на практике для специалистов, которые ранее исполь-

зовали MPI для написания программ. В этом аспекте перспективной представляется разработка системы автоматизированного перевода программ из модели двухсторонних коммуникаций в модель односторонних, т.е. системы, которая скроет от пользователя трудности, связанные с переходом на односторонние коммуникации. При этом такая система заметно увеличит масштабируемость программы.

Перспектива появления вычислительных систем экзафлопсного уровня формулирует большое количество задач и проблем, которые придется решать. В частности, это проблемы создания программного и аппаратного обеспечения для достижения экзафлопсной производительности, проблемы суперкомпьютерного образования, проблемы постановки реальных задач, требующих таких больших вычислений. Отрасль суперкомпьютерных вычислений в России стремительно развивается в последние годы. В данной статье показана актуальность перечисленных выше проблем экзакомпьютерных вычислений для России. Рассмотрены некоторые пути их решения.

Список литературы

1. Сайт конференции «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность–2009». – URL: http://agora.guru.ru/display.php?conf=abrau2009.

2. International Exascale Software Project. – URL: http://exascale.org.

3. Суперкомпьютер «Ломоносов». – URL: http://www.t-platforms. ru/ru/clusters/clusters/unique/lomonosov.html.

4. Российское сообщество экзакомпьютерных технологий. – URL: http://exascale.ru.

5. Суперкомпьютерный консорциум университетов России. – URL: http://hpc-russia.ru.

А.В. Евсеев, Ф.Н. Ясинский

Ивановский государственный энергетический университет

О РЕШЕНИИ УРАВНЕНИЯ НАВЬЕ-СТОКСА В ПЕРЕМЕННЫХ «ФУНКЦИЯ ТОКА – ВИХРЬ» С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ С НЕСКОЛЬКИМИ ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

Одним из основополагающих уравнений гидродинамики является уравнение Навье-Стокса, которое традиционно записывается в системе «Скорость – Давление» [1]:

$$\begin{cases} \dfrac{\partial \vec{u}}{\partial t} = -\left(\vec{u}\cdot\nabla\right)\vec{u} + \upsilon\Delta\vec{u} - \dfrac{1}{\rho}\nabla P + \vec{F}; \\ \operatorname{div}\vec{u} = 0, \end{cases} \tag{1}$$

где $\vec{u}$ – вектор скорости, P – давление среды, $\vec{F}$ – ускорение макрочастиц среды, вызванное обменом количеством движения с соседними макрочастицами, υ – кинематическая вязкость.

Кроме того, это уравнение можно свести к системе в переменных «Функция тока – Вихрь», которая может быть более удобной в некоторых задачах [1]. Если ограничиться двумерным случаем, то можно ввести следующие определения функции тока:

$$\begin{cases} u_1 = \dfrac{\partial \Psi}{\partial x_2}; \\ u_2 = -\dfrac{\partial \Psi}{\partial x_1} \end{cases} \tag{2}$$

и вихря:

$$\omega = \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}. \tag{3}$$

Соответственно уравнение Навье-Стокса преобразуется в следующую форму [1]:

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + u_1 \dfrac{\partial \omega}{\partial x_1} + u_2 \dfrac{\partial \omega}{\partial x_2} = \upsilon \left(\dfrac{\partial^2 \omega}{\partial x_1^2} + \dfrac{\partial^2 \omega}{\partial x_2^2} \right); & (4) \\ \dfrac{\partial^2 \Psi}{\partial x_1^2} + \dfrac{\partial^2 \Psi}{\partial x_2^2} = -\omega. & (5) \end{cases}$$

На твёрдых и неподвижных поверхностях граничные условия задаются следующим образом:

$$u_1\big|_{\Gamma} = 0;$$
$$u_2\big|_{\Gamma} = 0;$$
$$\Psi\big|_{\Gamma} = \text{const};$$
$$\omega\big|_{\Gamma} = -\frac{\partial^2 \Psi}{\partial n^2}\bigg|_{\Gamma},$$

где n – нормаль к границе Г.

На входных и выходных отверстиях задаётся профиль скорости, а расчётные формулы для функции тока и вихря получаются либо интегрированием, либо дифференцированием в соответствии с их определениями (2)–(3).

Также используемая модель предполагает, что кинематическая вязкость задана константой.

В процессе решения уравнения Навье-Стокса необходимо попутно решать уравнения Лапласа, Пуассона и множество трёхдиагональных нелинейных систем.

Уравнение Пуассона и Лапласа в вычислительном смысле являются «тяжёлым», так как обычно они интегрируются с помощью некоторого итерационного метода и содержат большое количество операций на каждой итерации.

Хотя решение трёхдиагональной системы является достаточно несложным и быстрым действием, алгоритм решения уравнения Навье-Стокса на каждой итерации содержит N^2 (N – количество узлов сетки) таких операций.

Одним из вариантов ускорения вычислительного процесса является использование графических ускорителей, представляющих собой массивно-параллельные процессоры с общей памятью. В отличие от центрального процессора с несколькими

ядрами один графический процессор содержит несколько сотен ядер, объединённых в потоковые мультипроцессоры, которые могут проводить вычисления параллельно.

Наиболее развитой на данный момент технологией является система CUDA, предложенная компанией Nvidia в 2007 году [7, 8]. В данной технологии вычисления производятся множеством блоков, состоящих из некоторого числа обрабатывающих потоков. В отличие от MPI, Nvidia CUDA представляет собой систему с общей памятью. Однако каждое обращение к общей памяти приводит к существенным задержкам, во время которых вычисления потоком не проводятся. Чтобы избежать подобных задержек, каждый потоковый мультипроцессор имеет некоторый объём разделяемой памяти, которая является быстрой общей памятью для всех потоков одного блока (рисунок). Каждый потоковый мультипроцессор состоит из 8 ядер, называемых CUDA-ядрами. Наиболее современные графические ускорители содержат до 60 потоковых мультипроцессоров или 480 CUDA-ядер.

Данные устройства хорошо себя показали при решении уравнения Пуассона [3], где использовалось одно устройство с 2 потоковыми мультипроцессорами. В то же время современные графические ускорители позволяют объединять их в общие вычислительные блоки по технологии Nvidia SLI или Nvidia Quadro SLI для объединения 2 или 4 графических карт.

Целью данного исследования было показать принципиальную возможность проведения гидродинамических вычислений на нескольких графических ускорителях. Упрощённая схема системы с двумя графическими ускорителями приведена на рисунке. В качестве тестовой аппаратуры выступал суперкомпьютер Ивановского института ГПС МЧС России с двумя графическими ускорителями GTX 295, каждый из которых имеет по два чипа на плате. Общее количество ядер равно 240 CUDA-ядер в чипе × 2 чипа × 2 карты = 960 CUDA-ядер при суммарном объёме видеопамяти 3,5 Гбайт.

В системе CUDA вычисления на каждой графической карте разбиваются на блоки потоков. Каждый блок может содержать до 512 потоков, причём они могут быть упорядочены линейно или в двух- или трёхмерную сетку. Отличительной особенностью CUDA является то, что в ней нет возможности осуществить глобальную синхронизацию вычислений между блоками иным способом, кроме как закончив вычисления во всех блоках. В связи с этим каждый этап вычислительного алгоритма может использовать собственное, оптимальное для него разделение задачи между блоками.

Однако в системе CUDA с несколькими графическими ускорителями присутствует одно важное архитектурное ограничение – не существует иного способа обмена данными между чипами, кроме как через ОЗУ компьютера. Таким образом, при проектировании алгоритмов решения необходимо учитывать эти ограничения и сводить количество пересылок данных на стыках к минимуму.

Заметим, что в данном случае можно рассматривать систему с несколькими графическими ускорителями CUDA как систему с разделённой памятью. Подобная аппроксимация позволяет применять методы, рассчитанные на системы параллельного программирования MPI. Так, например, решение уравнения Пуассона можно разделить на отдельные независимые участки со своими граничными условиями, обновляемыми на каждой итерации [2, 4]. В то же время эти независимые участки можно решать стандартными способами на каждой графической карте в отдельности [3].

Сложности возникают при решении трёхдиагональных систем. В случае с несколькими графическими ускорителями общее поле решения равномерно распределено между картами. Как уже указывалось выше, в этом случае пересылки данных необходимо осуществлять через центральный процессор и ОЗУ, следовательно, эффективные параллельные алгоритмы, например параллельная циклическая редукция [3, 6], становятся крайне неэффективными, если прогонку необходимо провести на данных, находящихся на разных картах.

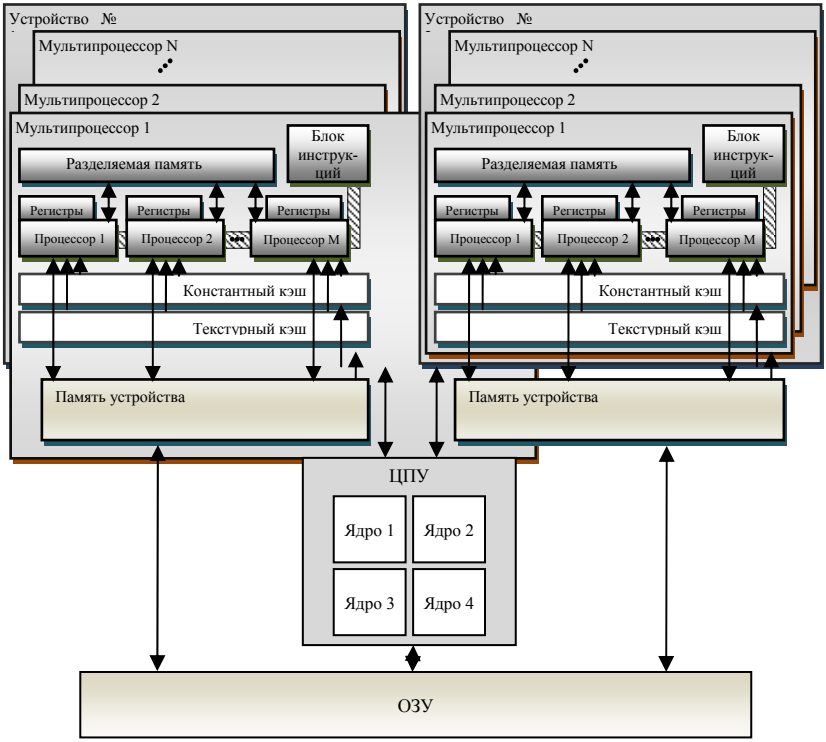


Рис. Архитектура системы с двумя устройствами CUDA

Поэтому, чтобы уменьшить количество пересылок данных между картами на данном этапе решения уравнения Навье-Стокса, было принято решение перестраивать данные так, чтобы прогонка всегда осуществлялась в пределах одной карты. Это позволяет использовать один общий алгоритм решения метода прогонки [3] с разным набором данных.

В двумерном случае решение уравнение Навье-Стокса предполагает вычисление прогонки для всей области данных в двух направлениях. Пусть данные будут разделены между картами полосками по строкам. В этом случае первая прогонка будет проходить на каждой карте независимо. Для второй прогонки необходимо подготовить данные, а именно транспонировать их. Транспо-

нирование также может быть эффективно выполнено с использованием графических ускорителей, поскольку каждый из них может повернуть свой участок данных в своей памяти, а затем они передают свои транспонированные участки в ОЗУ, где центральный процессор их записывает вертикальными полосками. В итоге получается транспонированное общее поле данных, которое затем снова разделяется на горизонтальные полоски и передаётся на карты, где снова проходит прогонка и т.д.

Таким образом, предложенный алгоритм позволяет снизить накладные расходы на пересылках данных, поскольку пересылки больших объёмов памяти более выгодны, чем множество маленьких пересылок данных.

Список литературы

1. Алгоритмы и программы для многопроцессорных суперкомпьютеров / В.В. Пекунов [и др.]. – Иваново: Изд-во Ивановск. гос. энерг. ун-та, 2007.

2. Евсеев А.В. Методы решения уравнения Пуассона. – Иваново: ИГТА, 2009.

3. Евсеев А.В. Вопросы распараллеливания уравнения Пуассона и сравнение эффективности различных вариантов // Высокие технологии, исследования, промышленность. Исследование, разработка и применение высоких технологий в промышленности: сб. тр. XI Междунар. науч.-практ. конф. – СПб.: Изд-во Политехн. ун-та, 2010. С. 46–52.

4. Евсеев А.В., Ясинский Ф.Н. Распараллеливание методов решения уравнения Пуассона // Высокопроизводительные параллельные вычисления на кластерных системах: материалы XI Междунар. науч.-практ. конф. – Владимир: Изд-во Владимирск. гос. ун-та, 2009. – С. 166.

5. Самарский А.А. Введение в численные методы. – М.: Наука, 1982.

6. Zhang, Y. Fast tridiagonal solvers on the GPU / Y. Zhang, J. Cohen, J.D. Owens // Principles and Practice of Parallel Programming, 2010. – P.127–136.

7. Специализированный курс «Архитектура и программирование массивно-параллельных вычислительных систем» на основе технологии CUDA в МГУ им. Ломоносова. – URL: http://cuda.cs.msu.su.

8. NVIDIA GPU Computing Developer. – URL: http://developer.nvidia.com/object/gpucomputing.html.

С.А. Ермаков, Е.Б. Замятина, А.А. Козлов

Пермский государственный университет

ОПТИМИЗАЦИЯ РАСПРЕДЕЛЕННОГО ИМИТАЦИОННОГО ЭКСПЕРИМЕНТА ПО ВРЕМЕНИ И ПО НАДЕЖНОСТИ

Метод имитационного моделирования широко используется в качестве метода исследования сложных динамических систем в самых разных областях знаний: в бизнесе, производстве, здравоохранении, при проектировании компьютерных систем, летательных аппаратов и т.д. Системы имитационного моделирования часто используются в качестве ключевого компонента в системах поддержки принятия решения. Именно поэтому необходимо, чтобы результаты моделирования были достоверны, а следовательно, и процесс проведения имитационного эксперимента должен быть *надежным*, защищенным от сбоев, которые могут привести к потере важной информации. Поскольку объектом моделирования являются сложные системы, то и время моделирования может быть весьма и весьма значительным. По этой причине возникает настоятельная необходимость в *сокращении времени* имитационного прогона.

Сокращение времени имитационного прогона возможно, если при его проведении существует доступ к ресурсам сразу

нескольких вычислительных узлов. В этом случае имитационную модель разбивают на логические процессы, которые обмениваются информацией, посылая сообщения друг другу. Тем не менее при проведении распределенного имитационного эксперимента возникает ряд проблем, которые становятся причиной временных потерь. Первой причиной является организация *алгоритма синхронизации* распределенных по вычислительным узлам объектов имитационной модели, второй - возникновение *дисбаланса*.

В настоящей работе авторы предлагают решения, которые в той или иной мере позволяют снизить потери, связанные с проблемами синхронизации объектов модели и неравномерным распределением нагрузки на узлы вычислительной системы. Решения и в том, и в другом случаях связаны с использованием знаний исследователя о поведении имитационной модели. При разработке оригинальных алгоритмов синхронизации и динамического равномерного распределения нагрузки на вычислительные узлы авторы используют методы искусственного интеллекта: мультиагентный подход, онтологии, экспертные системы.

Разработка перечисленных выше алгоритмов выполнялась для системы имитационного моделирования Triad.Net, которая первоначально предназначалась для автоматизированного проектирования и моделирования вычислительных систем (ВС). По этой причине целесообразно предварительно рассмотреть особенности представления имитационной модели в Triad [1], тем более что организация процесса моделирования в Triad. Net допускает динамическое изменение структуры имитационной модели, разрешая исследователю выполнять операции над структурой модели и над моделью в целом.

Описание имитационной модели. Описание имитационной модели в Triad состоит из трех слоёв: слоя структур (STR), слоя рутин (ROUT) и слоя сообщений (MES). Таким образом, модель в системе Triad можно определить как M={STR, ROUT, MES}.

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюсы (входные P_{in} и выходные P_{out}), которые служат соответственно для приёма и передачи сообщений. Слой структур можно представить графом. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами. Объекты действуют по определённому алгоритму поведения, который описывают с помощью рутины (rout). Рутина представляет собой последовательность событий e_i, планирующих друг друга ($e_i \in E, i=1 \div n$, E – множество событий, множество событий рутины является частично упорядоченным в модельном времени). Выполнение события сопровождается изменением состояния объекта. Состояние объекта определяется значениями переменных рутины. Таким образом, система имитации является событийно-ориентированной. Рутина так же, как и объект, имеет входные (Pr_{in} и выходные Pr_{out}) полюсы. Входные полюсы служат соответственно для приёма сообщений, выходные полюсы – для их передачи. Совокупность рутин определяет слой рутин ROUT. Слой сообщений (MES) предназначен для описания сообщений сложной структуры.

Трехслойное представление модели позволяет исследователю изучать моделируемый объект по частям (например, только структуру), постепенно усложняя и детализируя модель. Кроме того, допускается изменение структуры модели во время имитационного прогона: добавление или удаление вершин, полюсов, дуг и ребер. Допускается также операция наложения рутины на вершину и другие операции над моделью в целом. В случае распределенного выполнения процесса имитации возможность *выполнения операций* над моделями являются дополнительными аргументами для проведения балансировки нагрузки на вычислительные узлы, поскольку вычислительная нагрузка на узлах будет произвольно изменяться.

Алгоритмом имитации называют совокупность объектов, функционирующих по определённым сценариям, и синхронизи-

рующий их алгоритм. Последовательный алгоритм является событийно ориентированным. Каждая рутина имеет свой локальный календарь событий ($tloc_i$). При выполнении очередного события выполняется поиск события с минимальным временем (поиск в календарях событий всех рутин) и управление передается рутине, которая включает это событие. Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты – информационные процедуры и условия моделирования, реализующие *алгоритм исследования*. Информационные процедуры ведут наблюдение только за теми элементами модели (событиями, переменными, входными и выходными полюсами), которые указаны пользователем. Условия моделирования задают список информационных процедур, выполняющих наблюдение за элементами модели, анализируют результат работы информационных процедур и определяют, выполнены ли условия завершения моделирования.

Имитационная модель, описанная на языке Triad, может быть запущена на моделирование с помощью специального оператора *simulate*: *simulate* <список элементов моделей, за которыми установлено наблюдение> *on conditions of simulation* <наименование> (список фактических настроечных параметров>)[<список входных и выходных фактических параметров интерфейса>].

Следует отметить, что исследователь может одновременно запустить на моделирование сразу несколько моделей, указав элементы, за которыми будут следить информационные процедуры, перечисленные в условиях моделирования. Кроме того, можно указать сразу несколько условий моделирования. В этом случае каждая модель или ее копия могут выполняться на отдельном вычислительном узле, как на ВС с кластерной архитектурой, так и на ВС с GRID-архитектурой, параллельно.

Алгоритм синхронизации. Традиционно используются два подхода к реализации алгоритма синхронизации объектов моделирования, распределенных по разным вычислительным узлам: *консервативный* и *оптимистический*. Основной целью

этих алгоритмов является обеспечение выполнения каждым логическим процессом событий в порядке не убывания их временных меток, чтобы сохранить причинно-следственные связи.

Задача консервативного алгоритма – определить время, когда обработка очередного события из списка необработанных событий является «безопасной». Событие является безопасным, если можно гарантировать, что процесс в дальнейшем не получит от других процессов сообщение с меньшей временной меткой. В отличие от консервативных алгоритмов, не допускающих нарушения ограничения локальной каузальности, оптимистические методы не накладывают такого ограничения. При нарушении локальной каузальности (логический процесс получает событие, имеющее временную отметку меньшую, чем уже обработанные события) происходит восстановление правильного порядка событий [2, 3]. Самый известный алгоритм – Time Warp. Для восстановления порядка выполняется откат, последующие события обрабатываются повторно в хронологическом порядке. Откатываясь назад, процесс восстанавливает состояние имитационной модели, которое было до обработки событий (все состояния модели сохраняются), и отказывается от сообщений, отправленных в результате обработки событий, которые необходимо отменить в результате отката. Для отказа от этих сообщений разработан механизм антисообщений.

Для некоторых специфических моделей консервативные алгоритмы показывают производительность, превосходящую оптимистические в несколько раз, для других моделей выигрыш во времени дает оптимистический алгоритм. Анализ консервативного и классического алгоритмов показывает, что для увеличения производительности консервативного алгоритма необходимо расширить горизонт безопасных событий, а для оптимистического – сократить количество откатов, сдержав чрезмерное продвижение модельного времени. Существует большое количество модификаций алгоритмов синхронизации [4]. Все они предназначены для оптимизации алгоритма по времени.

Так или иначе, если алгоритм синхронизации располагает информацией о модели (lookahead, lookback, временная метка следующего события и т.п.), то производительность его повышается. Таким образом, для оптимизации времени выполнения распределенного имитационного эксперимента целесообразно использовать информацию о модели, в том числе и знания исследователя о конкретной модели, что позволит адаптировать алгоритм синхронизации в каждом конкретном случае. Авторы предлагают использовать для хранения информации исследователя о модели продукционные правила. Кроме того, знания должны извлекаться из самой модели как во время ее трансляции, так и во время ее функционирования.

Рассмотрим алгоритм, основанный на применении такой информации, созданный на базе оптимистического алгоритма. С точки зрения моделирования главным является соблюдение правильного порядка выполнения модели во времени. Итак, пользователь обычно имеет представление о поведении модели. В результате могут быть построены продукционные правила в виде IF e_1 AND e_2 AND e_3 AND … AND e_n THEN e_k CF <0..100>. Здесь e_k зависит от e_1, e_2, ,e_n. CF – коэффициент доверия. 0 – нет доверия, 100 – максимальное доверие. Правила отражают каузальную зависимость между событиями, а поскольку знания не являются точными, то каждому правилу приписывают коэффициент доверия.

Однако детальное описание поведения модели в виде правил достаточно затруднительно. Необходимо извлекать информацию о поведении модели автоматически. Эту информацию также будем хранить в виде правил (системные правила). Системные правила формируются на стадии трансляции (процедуры обработки событий и сообщений) и на стадии выполнения модели (анализ последовательности обработки событий). Для сбора, обработки и распространения информации алгоритма синхронизации целесообразно реализовать специальных агентов на вычислительных узлах. Информация (центральная база знаний) и главный управляющий процесс нахо-

дятся на выделенном сервере. Агент сбора и обработки информации определяет безопасное событие следующим образом: происходит поиск события в заключениях всех правил в базе знаний, а далее, если событие обнаружено, то выполняется логический вывод по правилам. Далее выбирают событие с максимальным коэффициентом доверия.

Проведенные испытания (для разных моделей и на ВС с различными конфигурациями) показали, что алгоритм дает выигрыш во времени. Особенно удачные результаты были получены для модели клиент-сервер (15 %).

Балансировка нагрузки на вычислительные узлы. Распределенная имитационная модель представляет собой совокупность объектов Triad-модели, которые располагаются на различных вычислительным узлах ВС и обмениваются информацией друг с другом. Распределение объектов выполняется на этапе статической балансировки исходя из структурных особенностей модели (клики располагаются на одном узле). При функционировании модели может возникнуть дисбаланс (гетерогенность вычислительных узлов и линий связи, гетерогенность имитационной модели). Восстановление равномерного распределения нагрузки выполняется динамической системой балансировки.

Существует большое количество алгоритмов, решающих эту задачу, но многие из них является «жесткими», при изменении модели или условий моделирования, алгоритма синхронизации, алгоритмы работают менее эффективно или вообще не дают никакого эффекта [5]. Авторы настоящей работы предлагают учитывать и особенности имитационной модели, и условий моделирования, и особенности вычислительной среды. Динамическая система балансировки TriadBalance является *мультиагентной*, она состоит из совокупности агентов разных типов: агента-датчика вычислительного узла; агента-датчика имитационной модели; агента анализа; агента миграции; агента распределения. Агенты каждого типа действуют по своему сценарию для достижения цели, а вместе они реализуют балансировку распределенной имитационной модели.

Алгоритм балансировки является децентрализованным и представляет собой последовательность шагов. 1. Агенты-датчики ВС (они располагаются на каждом вычислительном узле) регулярно на всем протяжении имитационного эксперимента собирают статистику о загруженности узлов, о состоянии линий связи, о наличии свободной памяти и размещают ее в базе данных (доска объявлений). Каждый вычислительный узел располагает своей доской объявлений. 2. Агент анализа, сканируя доски объявлений, с помощью правил из базы знаний определяет, необходимо ли выполнять балансировку. Например, при превышении показателя загрузки вычислительного узла агент анализа принимает решение о необходимости выполнения балансировки. В этом случае агент анализа обращается к агенту распределения. 3. Агент распределения извлекает информацию из базы данных. Информация в базе данных появляется в результате работы агента-датчика имитационной модели. Для извлечения информации о модели используют механизм информационных процедур. В частности, агенты-датчики собирают информацию о частоте выполнения событий имитационных объектов, частоте обменов между ними (частота появления сообщений на входных и выходных полюсах рутин). Таким образом, агент распределения на основании анализа данных на доске объявлений может сделать вывод о том имитационном объекте, который следует перенести на другой узел. Кроме того, он общается с агентами распределения, расположенными на соседних вычислительных узлах. Он сообщает о перегрузке, о том, что ему необходимо освободиться от некоторых своих имитационных объектов. Агенты распределения других вычислительных узлов извещают о своей нагрузке, пополняя базу данных агента распределения, корректируя правила в базе знаний этого агента. Во время сеанса взаимодействия агенты распределения других узлов могут сообщить о том, что они незагружены. Далее агент распределения действует по правилам, которые хранятся в базе знаний. Для принятия решения об адресе целевого узла агент распределения, наряду с другими правилами, использует, в ча-

стности, правила топологических характеристик ИМ и ВС. При выборе целевого узла сообщение направляется агенту миграции. 4. Агент миграции, получив запрос от агента распределения, выполняет непосредственно перенос выбранных объектов имитационной модели на выбранные целевые узлы сети.

Для повышения гибкости мультиагентной системы балансировки, адаптируемости когнитивных агентов к изменяющимся условиям используют метаправила, более точно – два типа метаправил: метаправила, определяющие структуру правил (их используют агенты распределения и анализа для распознавания нужных правил), и метаправила, определяющие, как можно изменить правила.

Настоящая версия мультиагентной подсистемы динамической балансировки реализована на кластере из 8 вычислительных узлов, работающих под управлением операционной системы Windows HPC Server 2008, которая является специализированной кластерной операционной системы Windows Server 2008. Подсистема балансировки разработана с использованием Net Framework 3.5 и пакета сборок HPC Pack 2008, позволяющего управлять распределённым выполнением приложений. Эксперименты показали (для модели «Клиент-сервер»), что время, затраченное на имитационный эксперимент при использовании подсистемы мультиагентной балансировки, действительно снижается, эффективность применения балансировки увеличивается с увеличением длительности эксперимента.

Надежность. Для обеспечения отказоустойчивости распределенной системы имитации (византийская модель сбоев) также была разработана специальная мультиагентная подсистема. Византийская модель сбоев (или модель произвольных сбоев) является наиболее общей моделью. Она позволяет моделировать такие типы сбоев, как отказы оборудования, потеря связи, неисправность оборудования, программные сбои, ошибки операторов и действия злоумышленников. Мультиагентная подсистема отказоустойчивости состоит из агента-детектора, агента анализа, агента репликаций и агента переноса кода (располага-

ются на каждом вычислительном узле). Аналогично подсистеме балансировки подсистема отказоустойчивости использует знания о модели, знания о состоянии вычислительных узлов и т.д. для выбора наилучшей стратегии в условиях сбоя (в отличие от многих существующих систем отказоустойчивости, которые предлагают ряд оптимизаций, но не указывают области, где их применение наиболее эффективно). Подсистема отказоустойчивости тесно взаимодействует с подсистемой балансировки и подсистемой визуализации.

Итак, в работе представлены программные решения, основанные на мультиагентном подходе и применении экспертных систем, которые позволяют оптимизировать распределенный имитационный алгоритм. Проведенные эксперименты подтвердили эффективность разработанных программных средств.

Список литературы

1. Mikov A.I. Simulation and Design of Hardware and Software with Triad // Proc. 2nd Intl. Conf. on Electronic Hardware Description Languages. – Las Vegas, USA, 1995. – P. 15-20.

2. Fujimoto R.M. Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference Chick S., Sánchez P.J., Ferrin D., Morrice D. J. eds., pp. 124–134.

3. Окольнишников В.В. Представление времени в имитационном моделировании // Вычислительные технологии. Т. 10, № 5; Сибирское отделение РАН. – 2005. – С. 57–77.

4. Вознесенская Т.В. Математическая модель алгоритмов синхронизации времени для распределённого имитационного моделирования // Программные системы и инструменты: темат. сб. факультета ВМиК МГУ им. Ломоносова. № 1. – М.: Изд-во МГУ, С. 56–66.

5. Wilson L.F. and Shen W. Experiments In Load Migration And Dynamic Load Balancing In Speedes // Proceedings of the 1998 Winter Simulation Conference. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds. – Washington, 1998. – P. 483–490.

[1]Д.В.Зимин, [2]В.П. Муленков, [1]Ю.В. Соколкин, [1]В.Я. Модорский

[1]Пермский государственный технический университет
[2]ГП «Возрождение», г.Пермь

МОДЕЛИРОВАНИЕ ВОЗДЕЙСТВИЯ ТРАНСПОРТНЫХ НАГРУЗОК НА КРЫШУ ВАГОНА-ХОППЕРА

В данной работе произведены численные эксперименты для крыши вагона из композиционного материала. Исследования проводились для одной секции, а также для конструкции крыши в целом с использованием кластерных технологий расчета. Задача заключалась в определении полей напряжений и перемещений при различных схемах нагружения. Расчеты произведены в конечно-элементном инженерном многопроцессорном пакете ABAQUS.

Средний срок службы вагона-хоппера составляет 25 лет. При перевозке минеральных удобрений стальная крыша вагона требует замены уже через 5 лет. Установка крыши из композиционного материала позволит добиться того, что конструкция будет работать на протяжении всего срока эксплуатации вагона. Вместе с тем будет обеспечена коммерческая загрузка вагона при его возврате.

Задача решалась в упругой постановке. Конструкция рассматривалась как многослойная оболочка. Материал задавался трехслойным (два слоя стеклопластика, между ними – слой пенопласта). Выбор свойств материала основывался на экспериментальных данных, при этом был введен критерий оценки работоспособности конструкции по максимальным деформациям для пакета. Предполагалось, что разрушение слоя пенопласта приведет к разрушению всего пакета, при этом стеклопластик может не разрушиться. Локальное разрушение пенопласта считалось недопустимым.

Несколько расчетных схем, имитирующих различные эксплуатационные нагрузки на конструкцию, представлены ниже:

Номер схемы нагружения	Характеристики схем нагружения
1	Статическое давление распора насыпного груза Продольное ускорение действующее на крышу 12g Вес конструкции крыши
2	Динамическое давление распора насыпного груза Продольное ускорение, действующее на крышу 1g Вес конструкции крыши с учетом вертикальной динамики
3	Максимальное распределенние по площади крыши давление Вес конструкции крыши Статическое давление распора насыпного груза
4	Вертикальное ускорение 1g, действующее на крышу в вертикальном направлении с частотой A = 3,2Гц
5	Воздействие груза на торцевую стенку крыши при резком торможении с ускорением 3g. Масса воздействующего груза 0,35 от массы груза

Для каждой из вышеперечисленных схем производились расчеты как для одной секции конструкции, так и для целой конструкции. Наибольшую опасность представлял случай, соответствующий схеме 5 (рис. 1). Для этого случая были выполнены расчеты для различного конструктивного исполнения торцевой части крыши и с различными свойствами и толщинами материала.

Вначале были произведены расчеты для одного сегмента крыши. Поскольку результаты по всем схемам были удовлетворительные, расчеты были продолжены для всей конструкции крыши.

Расчеты всей крыши изначально проводились на рабочей станции, но в связи с мелким размером ячеек расчетной сетки и большим количеством элементов расчеты проводились медленно и периодически зависали из-за недостатка оперативной памяти. В связи с этим было решено использовать кластер для проведения расчетов всей крыши.

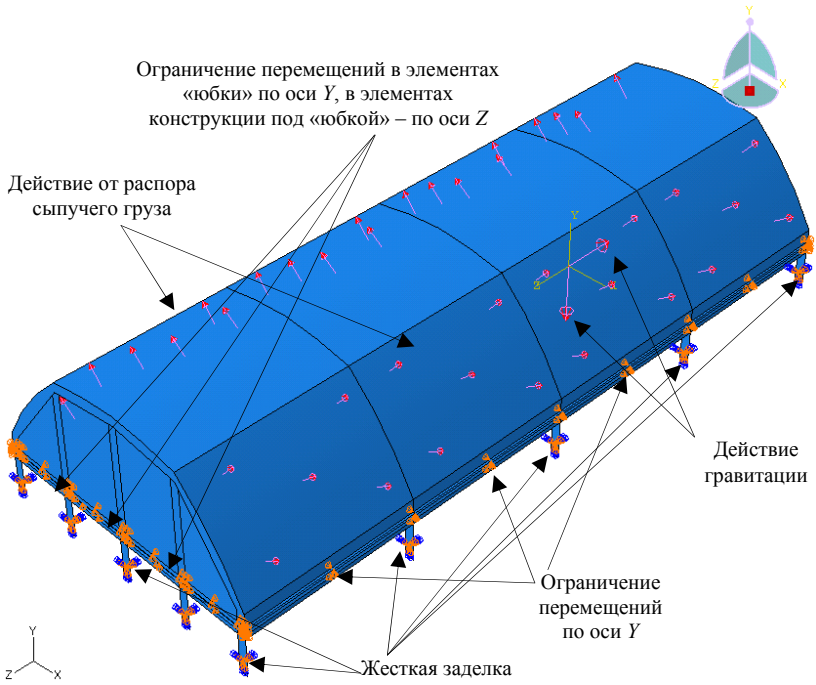


Рис. 1. Модель крыши из четырех секций с граничными условиями (схема 1)

При расчетах на кластере использовался один узел для решения одной задачи (схемы нагружения). Такой подход обеспечил наиболее рациональное использование ресурсов. Как видно из рис. 2, при использовании от 1 до 4 ядер для расчета наблюдается ускорение в 3,3 раза, а при использовании от 5 до 8 ядер ускорение незначительно. Исходя из того, что необходимые ресурсы оперативной памяти значительны, наиболее оптимально использовать один узел под расчет.

Особое внимание стоило уделить расчетной схеме 5. Особенность данного расчета заключалась в том, что из-за воздействия большой нагрузки на торцевую часть возможно ее разрушение. Для исключения разрушения был введен каркас жесткости. Каркас представляет собой арку, идущую по периметру

торцевой части, и три дополнительных ребра, идущих вдоль нее (рис. 3).

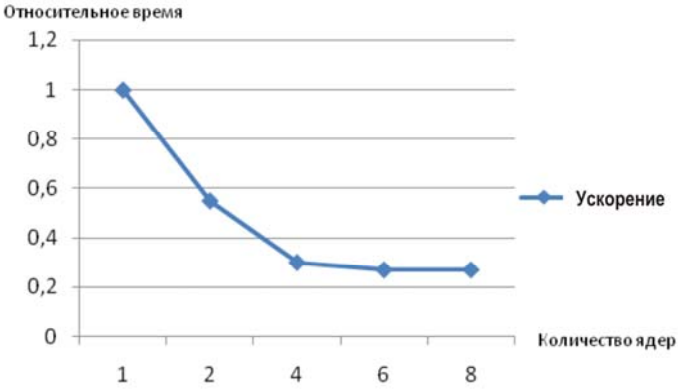


Рис. 2. Зависимость скорости проведения расчетов от количества используемых ядер (ускорение)

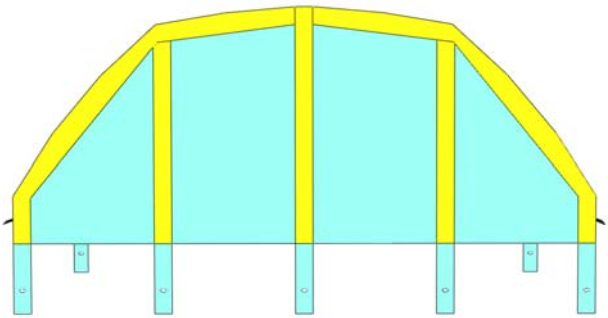


Рис. 3. Конструктивная схема каркаса жесткости с пятью ножками на торцевой панели секции крыши

Данный каркас располагается по периметру всей конструкции. На стыках секции количество продольных ребер сокращено до одного, идущего по центру крыши. Ребра жесткости являются квадратными в поперечном сечении. Принималось, что они изготовлены из стеклопластика, имеющего такие же свойства, как и материал крыши. По периметру боковых граней

крыши есть отвод, так называемая юбка, она используется для установки крыши на вагон, для расчетов это было интерпретировано как ограничение перемещений по оси *Y* для боковых граней и по осям *Y*, *Z* на торцевых гранях. Ограничение перемещений по торцевым граням позволило получить увеличение жесткости на торцевых панелях. Так же на торцевых панелях толщина материала увеличена по сравнению с остальной частью крыши. Все эти модификации привели к тому, что максимальные напряжения при расчете схемы 5 составили 70,95 МПа (рис. 4) и деформации 6,31 (рис. 5).

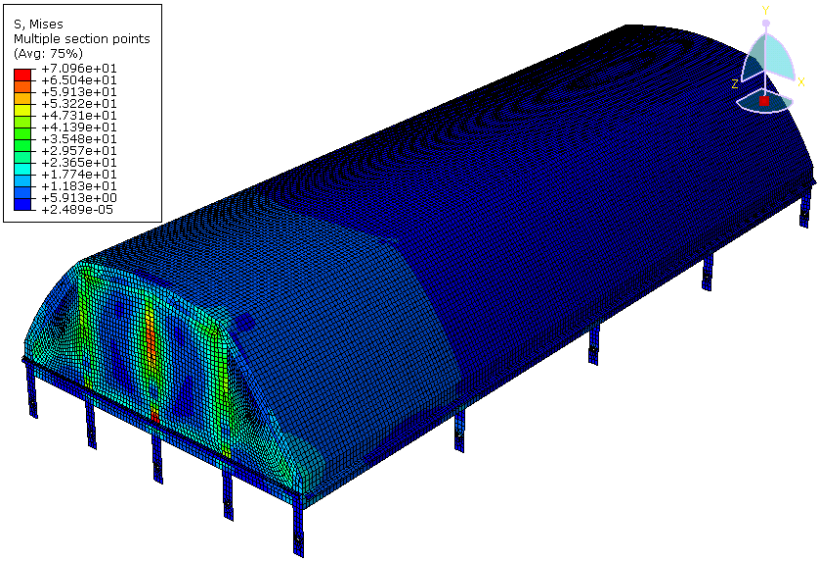


Рис. 4. Распределение полей напряжений (схема 5)

Коэффициент запаса для выбранного критерия оценки составил 2,74.

Для верхней части крыши наиболее важным был расчет по схеме 3. Максимальные напряжения для данного случая 3,054 МПа, а деформации 4,32 при действии распределенной нагрузки в 120кг/м^2. Коэффициент запаса для выбранного критерия оценки составил 63,85.

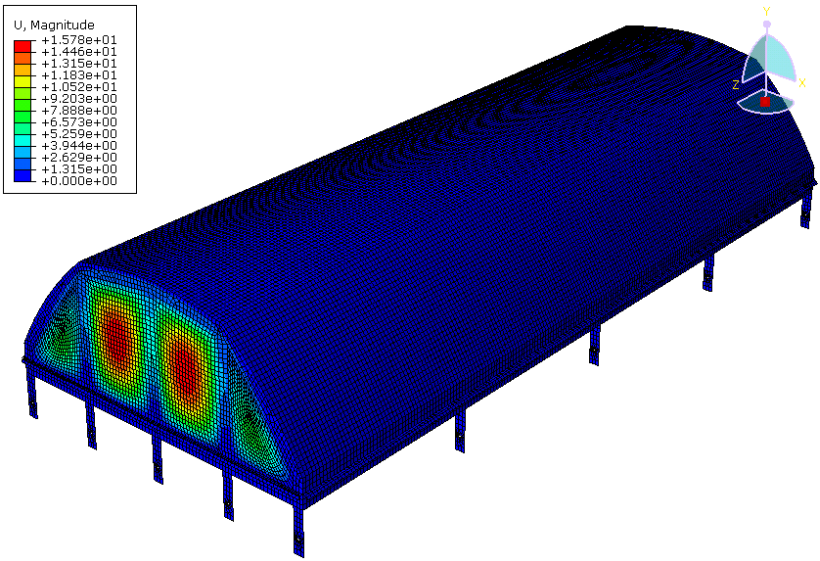


Рис. 5. Распределение полей перемещений (схема 5)

В результате использования кластерных технологий были произведены численные эксперименты конструкции крыши в целой без использования симметрии. Также использование кластерных систем позволило повысить скорость проведения численных экспериментов и их эффективность, предоставив возможность строить более мелкую сетку и увеличить количество точек интегрирования в каждом элементе.

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

Пермский государственный технический университет

СОВЕРШЕНСТВОВАНИЕ МЕТОДИКИ РАСЧЕТА СОЕДИНЕНИЙ ЭЛЕМЕНТОВ ДЕРЕВЯННЫХ АРОЧНЫХ КОНСТРУКЦИЙ НА СТАЛЬНЫХ ЦИЛИНДРИЧЕСКИХ НАГЕЛЯХ

Клееные деревянные арки находят широкое применение в практике строительства большепролетных общественных, спортивных, производственных и складских зданий благодаря относительно высокой долговечности, простоте изготовления и монтажа, наличию обширной сырьевой базы.

Весьма эффективно применение деревянных клееных конструкций эксплуатируемых в химически агрессивных средах. Большой опыт применения деревянных клееных арок накоплен в Пермском крае при возведении складов минеральных удобрений на калийных предприятиях. Построено более 30 складов пролетами 18–60 м и более (за рубежом имеются примеры применения клееных арок пролетом до 250 м), в которых в качестве несущих конструкций использованы стрельчатые арки и распорные конструкции треугольного очертания (А-образные арки).

При проектировании большепролетных арок и рам одной из проблем является устройство узловых соединений. Как правило, для устройства опорных и коньковых узлов арок и рам применяют шарнирные узловые соединения в виде стальных башмаков, прикрепляемых к полуаркам с помощью стальных цилиндрических нагелей (болтов, шпилек, глухарей). Напряженное состояние такого узлового соединения достаточно сложное и характеризуется смятием древесины в зоне контакта торца элемента с опорной плитой башмака, скалыванием и раскалыванием древесины в зоне контакта древесины с нагелем и изгибом самого нагеля, работающего как балка на упругопластическом основании.

Существующая методика расчета соединений такого типа, рекомендуемая СНиП II-25-80, сводится к сравнению минимальной несущей способности нагеля из условия смятия древесины нагельного гнезда или изгиба самого нагеля с усилием, действующим на наиболее нагруженный нагель. Однако *методики определения этих усилий в нормативной литературе не приводится*.

На соединительный башмак в узле действуют продольная и поперечная силы. Поперечная сила вызывает сдвиг башмака по плоскости соприкосновения торца элемента с опорной пластиной, а действие сжимающей продольной силы приводит к смятию древесины в зоне контакта. Из-за конструктивных особенностей соединительных элементов поперечная сила прикладывается с эксцентриситетом относительно плоскости сдвига и создает крутящий момент. В сложившейся практике проектирования для определения равнодействующих усилий, действующих на наиболее нагруженный нагель, используют формулу для расчета болтовых соединений в стальных конструкциях. В этой формуле величина эксцентриситета принимается равной расстоянию от точки приложения поперечной силы до центра узлового соединения. Однако при наличии достаточно длинной опорной плиты свободный поворот башмака невозможен, и в предельной стадии поворот будет происходить относительно крайней точки соприкосновения опорной плиты с торцом деревянного элемента; крутящий момент при этом окажется значительно меньше. В расчетах полностью *игнорируется наличие силы трения по плоскости контакта древесина–металл*, которая также будет препятствовать сдвигу.

При использовании упрощенной методики для расчета опорных и коньковых узлов большепролетных деревянных клееных арок и рам требуемое по расчету количество нагелей оказывается неоправданно большим, что приводит к перерасходу металла. Наличие значительных запасов несущей способности узловых соединений, запроектированных по традиционной методике, подтверждается результатами экспериментов и наблюдениями за длительно эксплуатируемыми большепролетными арками складов калийных солей.

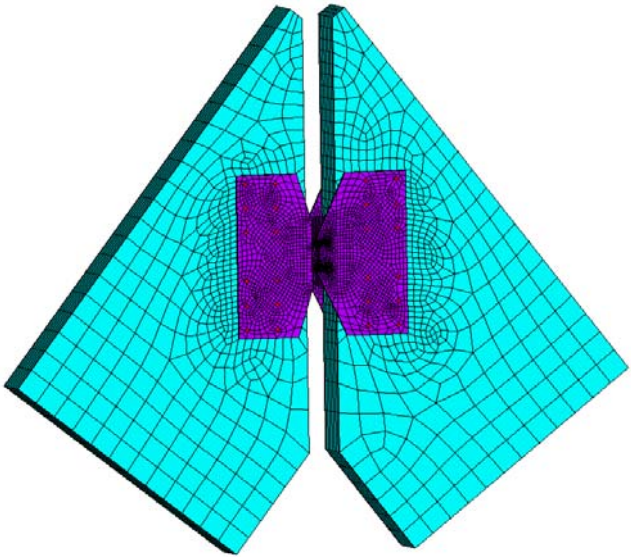


Рис. Конечно-элементная модель соединения

В связи с вышесказанным вопросы рационального проектирования узловых соединений большепролетных деревянных клееных конструкций на стальных цилиндрических нагелях является весьма актуальной. Одним из путей решения данной проблемы может быть использование методов математического моделирования с использованием современных конечно-элементных программных комплексов и ЭВМ.

Эта задача решалась в ПК ANSYS с учетом контакта в плоскости соприкосновения торца полуарки с опорной металлической пластиной и в зонах соединения древесины с нагелями (рисунок). Использовались специальные контактные элементы: TARGE170 и CONTA173. При этом учитывались ортотропные свойства древесины.

Для оценки усилий в нагелях и определения их количества использовалась математическая теория планирования эксперимента. В качестве варьируемых факторов принимались: диаметр нагелей, расстояния между нагелями по вертикали и горизонтали, размер опорной металлической пластины. В качестве функ-

ций отклика рассматривались: максимальные эквивалентные напряжения в нагелях, максимальные контактные напряжения в нагелях, максимальные эквивалентные напряжения в древесине, максимальные контактные напряжения в опорной металлической пластине.

По полученным уравнениям построены зависимости максимальных напряжений в элементах узлового соединения от расстояний между нагелями по вертикали и горизонтали при фиксированных значениях диаметра нагелей и размера опорной пластины.

Полученные зависимости дают возможность рационального подбора параметров соединения при проектировании большепролетных арок и рам.

Н.Ю. Золотых, Е.А. Козинов, В.Д. Кустикова, И.Б. Мееров, А.Н. Половинкин

Нижегородский государственный университет им. Н.И. Лобачевского

ОБ ОДНОМ ПОДХОДЕ К РЕШЕНИЮ ЗАДАЧИ ПОИСКА ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ

Одной из классических задач в компьютерном зрении является задача классификации, заключающаяся в том, чтобы предсказать наличие или отсутствие объекта некоторого класса (машина, птица, человек и т. п.) на изображении или кадрах видео. Наряду с указанной задачей выделяется задача детектирования, в которой необходимо определить конкретное положение объекта. Один из подходов к решению задачи детектирования состоит в использовании алгоритмов машинного обучения для построения модели класса и алгоритмов поиска (вывода) положения объекта на некотором заданном изображении.

В данной работе приводится схема алгоритма вывода, предложенного в [1], описана предлагаемая реализация данного алгоритма, дается анализ точности детектирования объектов на тестовом

множестве изображений PASCAL Visual Object Classes Challenge 2007 (VOC 2007, http://pascallin.ecs.soton.ac.uk/challenges/VOC/), а также рассматриваются вопросы производительности текущей реализации. Разработка ведется в рамках популярной библиотеки компьютерного зрения с открытым исходным кодом OpenCV (http://sourceforge.net/projects/opencvlibrary/).

Постановка задачи. Решается задача детектирования. Требуется выполнить поиск объектов некоторого класса на изображении (сцена может содержать несколько объектов одного класса). Положение объекта определяется прямоугольником, окаймляющим границы этого объекта. Необходимо построить на изображении множество окаймляющих прямоугольников для объектов заданного класса.

Схема решения. Для решения задачи необходимо реализовать алгоритм вывода [1].

Начальным этапом алгоритма является построение пирамиды признаков, которое состоит из нескольких действий:

1. Масштабирование исходного изображения. Данная процедура позволяет получить набор изображений с увеличенным и уменьшенным разрешением – пирамиду изображений.

2. Построение матриц векторов признаков для каждого изображения в пирамиде. Указанная процедура включает два этапа:

а) разбиение изображения на блоки фиксированного размера. Блок состоит из подмножества пикселей, каждый из которых характеризуется интенсивностью цвета;

б) вычисление значений компонент вектора признаков для блока на основании гистограммы ориентации градиентов (histogram of oriented gradients (HOG) features, [1]). Под градиентом понимается разность интенсивностей соседних пикселей.

Построенная математическая модель изображения используется для определения расположения прямоугольника, окаймляющего объект. Для решения поставленной задачи необходимо наличие модели, описывающей класс детектируемых объектов. Модель объекта определяется набором компонент, каждая из которых соответствует одному из ракурсов этого объекта (например, если

осуществляется поиск человека на изображении, под ракурсом можно понимать полную фигуру в фас, в профиль и т.д.). Компонента включает совокупность фильтров:

- грубый фильтр, определяющий множество векторов признаков, наиболее характерных для всего объекта;
- совокупность точных фильтров, описывающих отдельные части объекта.

Поскольку исходная задача состояла в реализации алгоритма поиска и не включала в себя реализацию алгоритма обучения, для проведения экспериментов были использованы модели авторов статьи [1], преобразованные в формат xml в соответствии с разработанной структурой (для построения модели в [1] применялся латентный метод опорных векторов, Latent SVM).

Идея алгоритма определения положения объекта состоит в том, чтобы некоторым образом оценить вероятность нахождения объекта во всех возможных положениях пирамиды изображений и выбрать наиболее вероятные положения.

Данный алгоритм включает следующие этапы:

1. Вычисление значений оценочной функции для каждого возможного положения объекта. Положение объекта определяется расположением левого верхнего угла грубого фильтра в матрице векторов признаков какого-либо уровня. Оценочная функция строится как сумма скалярных произведений векторов признаков грубого и точных фильтров модели с соответствующими векторами матрицы признаков.

2. Выбор положений, для которых значения оценочной функции превышают пороговое значение (является параметром модели). Полученные координаты определяют положение грубого фильтра, описывающего объект, в матрице признаков. Каждый вектор признаков соответствует некоторому блоку изображения, поэтому необходимо выполнить преобразование координат в координаты пикселей.

3. Преобразование блочных координат, соответствующих найденным положениям объекта на различных уровнях пирамиды признаков, в пиксельные координаты исходного изображения. Полученное множество точек – координаты левых верхних

углов окаймляющих прямоугольников. Далее необходимо определить границы окаймляющих прямоугольников.

4. Вычисление координат правых нижних углов окаймляющих прямоугольников на основании размерности грубого фильтра.

Программная реализация

В алгоритме вывода можно выделить два основных модуля:

1. Модуль построения пирамиды признаков (feature pyramid).

2. Модуль вычисления положения объекта заданного класса (matching).

Для проведения экспериментов установления качества распознавания было разработано тестовое приложение, схема функционирования представлена на рисунке

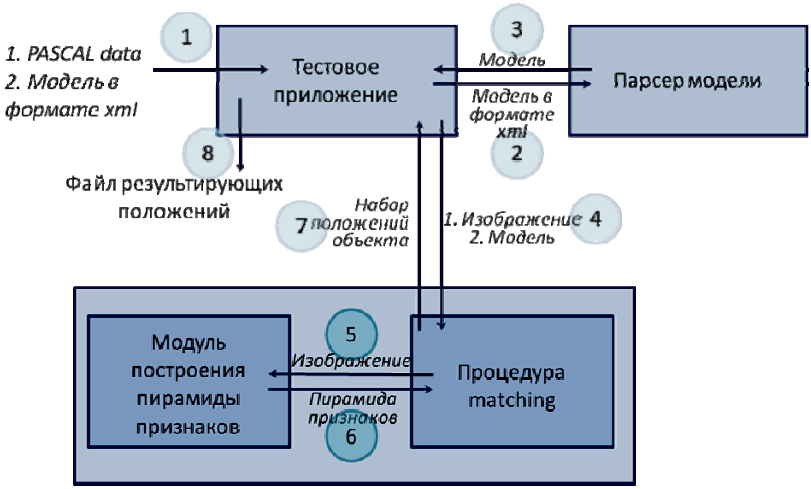


Рис. Схема функционирования тестовой подсистемы

1. Приложение принимает на вход путь до набора изображений и их описаний PASCAL VOC и модель объекта в формате xml (рисунок, п. 1).

2. Xml-файл модели разбирается парсером, который реализован в виде дополнительного модуля (рисунок, п. 2, 3).

3. Для каждого изображения выполняется поиск возможных положений объекта в соответствии с входной моделью (рисунок, п. 4, 7). С этой целью строится пирамида признаков для указанного изображения согласно процедуре, описанной ранее (рисунок, п. 5, 6). Последовательное получение изображений из набора осуществляется с помощью функциональности библиотеки OpenCV.

4. Найденные наиболее вероятные положения объекта сохраняются в результирующем файле (рисунок, п. 8), формат которого приемлем для вычисления метрики качества алгоритма вывода с помощью VOC Development Kit (http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/index.html#devkit). Сохранение осуществляется также средствами OpenCV.

Результаты экспериментов. В качестве тестового набора использовалась база данных конкурса PASCAL VOC 2007, содержащая различные изображения объектов двадцати классов (aeroplane, bicycle, bird, bottle и др.). Представленные фотографии различаются размером изображенных на них объектов, их положением на сцене, ракурсом и степенью освещенности. Указанные факторы оказывают значительное влияние на точность построенной модели.

Оценка качества детектирования объектов с помощью реализованного алгоритма вывода выполнялась средствами VOC Development Kit, в состав которого входит модуль, позволяющий вычислить среднюю точность предсказания (average precision). Указанная метрика определяется как математическое ожидание точностей следующим образом:

$$AP = \frac{1}{11} \sum_{r \in [0;0.1;...;1]} p(r), \quad p(r) = \max_{\bar{r}:\bar{r} \geq r} p(\bar{r}),$$

где $p(\bar{r}) = \frac{a}{a+c}$ – точность, $\bar{r}$ – процент перекрытия детектированного окаймляющего прямоугольника и прямоугольника, который был размечен на исходном изображении как окаймляющий $\bar{r} \in [0;0.1;0.2;...;1]$, a – количество объектов, для которых процент

перекрытия не меньше, чем $\bar{r}$ (т.е. считается, что объект детектирован правильно), c – количество объектов с процентом перекрытия, меньшим, чем $\bar{r}$ (объект найден ошибочно).

Вычислительные эксперименты проводились с использованием следующей инфраструктуры:

- Язык разработки: C.
- Среда разработки: Microsoft Visual Studio 9.0.
- Компилятор: Microsoft C/C++ Compiler Version 15.00.30729 (x64).
- Процессор: 2 двухъядерных процессора Intel Xeon 5150 (2.66 GHz).
- Память: 4 Gb.
- Операционная система: Microsoft Windows Server 2008 Standard SP1 x64.

В таблице приведены численные значения средней точности, полученные на текущей реализации алгоритма вывода. Сравнение с результатами, приведенными в [1], показывает, что практически на всех классах объектов наблюдаются незначительные отклонения, причем как в худшую (наибольшая разница составляет 0,019 на классе bicycle), так и в лучшую сторону (наибольшая разница – 0,023 на классе pottedplant). Значения средней точности, полученные в [1], в настоящее время являются лучшими из известных результатов, о чем свидетельствует первое место в PASCAL VOC Challenge 2009 (detection problem).

Несмотря на практически идентичное качество детектирования, текущая реализация в среднем обрабатывает одно изображение в 2–2,5 раза медленнее, чем реализация авторов алгоритма, которая решает задачу детектирования объектов одного класса на изображении примерно за 2 секунды ([1]). Данный факт объясняется тем, что настоящая версия является последовательной, в отличие от той, с которой проводится сравнение. Дальнейшее распараллеливание для систем с общей памятью позволит достигнуть лучших результатов по скорости.

Средняя точность детектирования объектов, полученная с использованием текущей реализации алгоритма вывода и представленная разработчиками алгоритма, на данных VOC 2007

Название класса объектов	Средняя точность (average precision) текущей реализации	Средняя точность (average precision) реализации авторов статьи [1]	Среднее время обработки одного изображения (с)
aeroplane	0,278	0,280	4,205
bicycle	0,525	0,544	4,412
bird	0,006	0,007	3,746
boat	0,123	0,145	3,623
bottle	0,260	0,262	3,438
bus	0,409	0,398	4,435
car	0,457	0,463	3,215
cat	0,163	0,160	5,032
chair	0,154	0,163	4,890
cow	0,167	0,167	4,696
dining-table	0,238	0,243	5,185
dog	0,070	0,050	4,850
horse	0,437	0,438	5,117
motorbike	0,384	0,382	5,120
person	0,338	0,342	3,747
potted-plant	0,101	0,079	3,417
sheep	0,166	0,172	3,130
sofa	0,214	0,221	4,998
train	0,331	0,340	4,208
tvmonitor	0,380	0,393	4,208

Текущая реализация алгоритма вывода позволяет детектировать объекты с точностью, которая не уступает реализации разработчиков алгоритма, о чем свидетельствуют результаты

проведенных экспериментов. Данная реализация интегрирована в библиотеку с открытыми исходными кодами OpenCV, использование которой не требует дополнительного дорогостоящего программного обеспечения.

Среднее время поиска объектов одного класса на изображении превышает время, достигнутое авторами статьи [1] на тех же классах. Профилировка разработанного приложения показала, что наиболее трудоемким этапом в представленной схеме является вычисление значений оценочной функции. Поэтому в дальнейшем планируется выполнить алгоритмическую оптимизацию и распараллеливание задачи определения положения окаймляющих прямоугольников (matching) для систем с общей памятью (очевидно, что вычисление значений оценочной функции на каждом уровне пирамиды признаков можно производить независимо), что позволит ускорить обработку изображения в несколько раз.

Авторы благодарят В.Л. Ерухимова (компания Itseez) за ценные замечания и постоянное внимание к работе.

Работа выполнена при поддержке федеральной целевой программы «Научные и научно-педагогические кадры инновационной России», госконтракт 02.740.11.5131.

Список литературы

1. Object Detection with Discriminatively Trained Part Based Modes / P.F. Felzenszwalb [et al.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2010. – Vol. 32, No. 9. – P. 1627–1645.

2. Dictionary of Computer Vision and Image Processing / R. Fisher [et al.]. – John Wiley. – 2005.

В.А. Зубарев, В.Н. Терещенко

Киевский национальный университет им. Т.Г. Шевченко

ОБОБЩЕННЫЙ ПАРАЛЛЕЛЬНО-РЕКУРСИВНЫЙ АЛГОРИТМ РЕШЕНИЯ НАБОРА ЗАДАЧ ГЕОМЕТРИЧЕСКОГО D-МОДЕЛИРОВАНИЯ

Работа посвящена созданию эффективных подходов решения задач геометрического моделирования в пространствах E^d. Предлагается новая парадигма создания алгоритмической среды для решения комплекса взаимосвязанных геометрических задач – обобщенный параллельно-рекурсивный алгоритм, использующий набор эффективных параллельных процедур решения. За основу обобщенного алгоритма взята стратегия «разделяй и властвуй», что позволяет эффективно использовать параллельные вычисления на каждом шаге алгоритма. Получено эффективное решение как отдельных задач, так и их совокупности. Установлены оценки сложности разработанных процедур и осуществлена их практическая реализация, подтвердившая их эффективность.

Графическое моделирование является одним из эффективных современных способов исследования и анализа сложных динамических процессов и явлений в науке и технике. В частности, это актуально для моделирования физических и механических процессов, которые происходят при высоких температурах нагревания твердого и жидкого тела, графического рендеринга, моделирования критических процессов реального времени и других сложных динамических процессов и явлений. Современный высокотехнологический уровень науки и техники ставит новые проблемы и определяет направления исследований относительно разработки новых и модификации существующих эффективных методов и алгоритмов. В основе этого лежат новые парадигмы и направления разработки, такие как параллельные и распределенные вычисления, а также архитектурные особенности новейших вычислительных систем.

Предметом исследования являются проблемы, задачи и алгоритмы геометрического моделирования в пространствах большой размерности.

Объектом исследования является *применение параллельных вычислений в задачах геометрического моделирования и решения набора взаимосвязанных задач вычислительной геометрии на одном и том же множестве входных данных*.

Целью исследования является поиск подходов, которые позволяют разработать простую и в то же время эффективную алгоритмическую среду для одновременного решения целого комплекса взаимосвязанных задач вычислительной геометрии в евклидовых *d*-мерных пространствах. Для решения этой проблемы необходимо разработать обобщенный параллельно-рекурсивный алгоритм, использующий общие алгоритмические инструменты и структуры данных, а также соответствующий набор процедур реализации.

К задачам геометрического моделирования можно отнести исследование целевых объектов познания по их модели, построение и изучение моделей реально существующих предметов, процессов или явлений, а также создание моделей новых и еще не изученных явлений. Геометрическая модель – определенный объект, природа которого отображает главные с точки зрения поставленной задачи пространственно-геометрические свойства его моделирования. К таким свойствам можно отнести расположение объекта в пространстве, его форма, внутренняя структура, и др. При построении модели для описания всего комплекса свойств необходимо решать одновременно целую совокупность взаимосвязанных геометрических задач.

Достаточно полно изучен двумерный случай. Но более сложными и важными на практике являются задачи для больших измерений. И здесь существует много открытых и нерешенных проблем, одной из которых является построение единого общего решения для целого комплекса взаимосвязанных задач. В большинстве случаев, даже при том, что каждая отдельная задача имеет эффективный алгоритм, в совокупности

это не всегда дает желаемую эффективную реализацию. В основном это связано с несоответствием структур данных и отсутствием связи между входными данными и результатами вычислений задач, что затрудняет возможность взаимодействия между процедурами алгоритмов и не дает минимального времени решения такого класса задач. Поэтому необходима разработка универсального инструмента, который имел бы общие средства для эффективного решения всего комплекса взаимосвязанных геометрических и прикладных задач. Другими словами, мы должны выбрать стратегию, которая использует общие средства реализации: структуры данных, отдельные этапы алгоритмов, процедуры и некоторые их шаги, а также способы представления результатов. И одной из таких стратегий может быть использование параллельных алгоритмов.

На сегодняшний день разработано много эффективных параллельных алгоритмов решения отдельных задач вычислительной геометрии. Так, в работе [1] описаны эффективные алгоритмы с использованием схемы «разделяй и властвуй», решения задач построения выпуклой оболочки для 2- и 3-мерных пространств с оценкой сложности $O(logN)$ и $O(log^3N)$, соответственно, а также построения диаграммы Вороного ($O(log^2N)$). В этой же работе представлен достаточно глубокий анализ других эффективных алгоритмов решения задач пересечение отрезков, триангуляции полигонов, оптимизации ($O(logN)$). В работах М.Т. Гудриха, М.Дж. Атталаха, Р. Коле, Н.М. Амато, Д.З. Чена и других авторов [2–7] получены улучшенные результаты решения вышеуказанных задач для различных моделей вычислений (CREW, EREW) и для высших размерностей ($d \geq 4$). Ряд статей посвящен общим параллельным методам, которые применимы к решению отдельных задач вычислительной геометрии [8–11]. Но после анализа непосредственной реализации многих из этих алгоритмов были выявлены некоторые проблемы, которые не были авторами учтены. В частности, проблемы возникают при анализе границ выпуклых оболочек, когда звенья границ имеют количество точек большее, чем размерность целевого пространства. В то же время идеи, предложеные в работах [1, 4–6, 12, 13],

дают нам представление о структуре необходимой геометрической модели в многомерных пространствах и пути развития необходимых стратегий построения моделей. Особенно следует отметить роботу М. Шеймоса и Д. Хоея, которые впервые в своей работе [14] использовали стратегию «разделяй и властвуй» при построении выпуклой оболочки для случая трехмерного пространства. Они получили наилучшую оценку *O(NlogN)* для однопроцессорной машины. Также необходимо упомянуть работу Т. Чена [15], в которой он элегантно реализовал их идею.

Но при решении целого комплекса задач на одном и том же множестве данных использование отдельных эффективных алгоритмов в общем случае не приносит желаемой эффективности. Дело в том, что каждый алгоритм нуждается в собственной предварительной обработке и создании структуры данных, а также процедур реализации, что не дает минимального времени решения такого класса задач. Поэтому необходимо было выбрать такую стратегию, которая бы позволяла получить самую эффективную реализацию. Учитывая отмеченные особенности и то, что большинство задач вычислительной геометрии имеет внутренний параллелизм и предусматривает рекурсивную природу реализации, наиболее подходящей стратегией, на наш взгляд, может быть та, в основе которой лежит параллельно-рекурсивный алгоритм, использующий технику «разделяй и властвуй». Здесь этапы предварительной обработки и разбиения множества будут общими для всего комплекса задач, а на этапе слияния предлагается использовать общую для всех задач структуру данных – *взвешенную сцепляемую очередь*. Кроме того, результаты отдельных этапов некоторых процедур будут использоваться другими процедурами, что обеспечивает высокую эффективность.

Параллельно-рекурсивный алгоритм. Поскольку нашей целью является создание алгоритмической среды для реализации комплекса задач, в качестве общей стратегии построения решения мы взяли схему «разделяй и властвуй». Эта стратегия позволяет нам построить эффективный обобщенный параллель-

но-рекурсивный алгоритм на уровне управления и использования общей структуры данных, что открывает возможность эффективной разработки не только на машинах с общей памятью, но и с распределенной. Кроме того, выделение и решение задач на подзадачах позволит при необходимости эффективно обмениваться уже полученными результатами между отдельными задачами, что уменьшает алгоритмическую нагрузку на них.

Общая схема. Весь процесс вычисления можно изобразить следующим образом: *предобработка, разбиение множества точек на равномощные подмножества (рекурсивный спуск), решение комплекса задач на подмножествах, начиная с элементарных, и слияние результатов при возвращении к исходному множеству.*

Поскольку предварительная обработка и шаг разбиения общие для всего комплекса задач, мы опишем лишь шаг слияния на примере задачи построения выпуклой оболочки. Процедуры слияния для задач построения триангуляции и диаграммы Вороного аналогичные и отличаются лишь деталями.

Алгоритм построения процедуры слияния выпуклых оболочек

Алгоритм слияния можно схематически разделить на 3 этапа: построение опорных граней моста, удаление лишних вершин и граней, окончательное построение граней моста.

Этап 1. *Построение опорных граней моста*, которые соединяют две оболочки.

1.Выбираем внутреннюю точку p. Пусть это будет точка, которая лежит в серединной плоскости, разделяющей взаимно выпуклые части двух выпуклых оболочек.

2. Находим первую опорную грань. Берем первую точку левой выпуклой оболочки и первую точку правой. На этих точках и векторах b_2, b_{n-1} базиса B пространства E^d строим плоскость. В качестве положительного направления вектора нормали выбираем тот, полуплоскость которого содержит точку p. Перебираем соседей точки первого выпуклого множества. Если новая точка вместе с правой образует плоскость, относительно которой предыдущая

точка вместе с точкой p будет лежать в одной полуплоскости, вместо первой точки берем вновь найденную. Повторяем этот шаг до тех пор, пока можно найти такую точку.

3. Аналогично для следующей грани.

4. Повторяем предыдущие два шага до тех пор, пока не сможем изменить ни одну из точек. Таким образом, мы имеем уже 2 точки, принадлежащие определенной плоскости на выпуклой оболочке. Дальше, нам необходимо найти другие d–2 точки. Для этого исключаем из базиса вспомогательной плоскости по одному вектору и перебираем соседей текущих точек. Если текущая точка лежит по разные стороны от текущего кандидата и центральной точки, переписываем текущего кандидата на новую точку. Первого кандидата выбираем автоматически в качестве первого соседа. В результате получаем грань выпуклой оболочки.

5. Для найденной грани перебираем все ее ребра и для каждого ребра, концы которого принадлежат разным оболочкам, из соседних точек ребра находим такую, относительно которой все другие точки лежат в одной полуплоскости с p:

а) если найденной грани нет в списке уже прибавленных граней к выпуклой оболочке, добавляем ее и выполняем пункт 5;

б) если найдено определенное подмножество точек, которые являются кандидатами на грань на плоскости, но для этого подмножества еще не выполняется пункт 5, то триангулируем это множество по условию принадлежности каждой грани триангуляции опорной грани (дальше «мост»). Каждую полученную грань добавляем к «мосту» и выполняем пункт 5 для нее. Берется начальное ребро из грани, которую проверяем, формируется плоскость с этой гранью и вектором нормали к плоскости, в которой это все строится. Среди точек текущего множества, находящихся по разные стороны относительно плоскости с точками, уже включенными в мост, находим точку с минимальным углом к этой плоскости. При этом рассматриваются только те точки, которые находятся слева и справа, а также смежные с текущими точками;

в) иначе поиск прекращаем. Когда поиск прекращен для всех граней, переходим к следующему пункту.

6. Найденное множество и является мостом между оболочками.

Этап 2. *Удаление вершин с гранями*, которые принадлежат «внутренней части» оболочек подмножеств и не принадлежат результирующей оболочке.

1. Возвращаемся к центральной точке p_1, которую использовали для объединения дочерних множеств с первой. Сначала перебираем точки моста на левой оболочке, а затем – все грани, которым принадлежит эта точка. Для каждой грани проверяем, лежит ли точка p_1 в одной полуплоскости с точками максимума и минимума по каждой координате второй оболочки. Если нет, и все точки грани принадлежат мосту, то просто удаляем эту грань. Если же какая-то точка не принадлежит мосту, то запускаем шаг 2 (такой вызов будет единственен). После проверки всех ребер, переходим к шагу 3.

2. Отмечаем точку как удаленную из оболочки и рекурсивно запускаем этот шаг для всех соседей, которые не принадлежат мосту. На шаге рекурсивного подъема удаляем все ребра и грани, которые выходят из этой точки.

3. Выполняем шаги 1–3 для другой оболочки.

Этап 3. Грани моста добавляем к выпуклым оболочкам и получаем выпуклую оболочку, общую для всего множества.

Теорема 1. *Выпуклую оболочку на множестве S из N точек в евклидовом d-мерном пространстве* E^d $(d > 2)$ *можно построить параллельно-рекурсивным алгоритмом с использованием q процессоров* $(q \le N)$ *за время* $O(\log(N)/(\max(2,\log(q)+1)d(K/2+dK/\max(1,q-\log(q)))))$, *где K – количество вершин выпуклой оболочки), с ускорением процедуры слияния* $R = \log(N) / \max(2,\log(q)+1)$.

Доказательство теоремы выводится из анализа сложности основных этапов алгоритма. Для двумерного случая эта оценка будет $O(\log N/\min(2,\log(q)+1)K)$, поскольку количество точек моста всегда 2 на каждом из подмножеств.

Особенности построения процедур слияния для триангуляции и диаграммы Вороного

Задача триангуляции на ограниченном множестве точек достаточно изучена и для ее решения существует ряд эффектив-

ных последовательных алгоритмов. Здесь следует отметить работы Скворцова, и в частности роботу [16], в которой предложено несколько быстрых алгоритмов триангуляции. В данной работе предлагается процедура триангуляции для рассматриваемого параллельного алгоритма в случае асимптотически больших множеств точек (или множеств точек, представляющих структурированные объекты).

Легко доказать, что триангуляция в евклидовом d-мерном пространстве – это то же самое, что выпуклая оболочка в пространстве $d + 1$. Для этого нужно расширить пространство размерности d до d+1, а точки заданного множества S необходимо поместить на d+1-мерный параболоид, где d+1 координата каждой точки определяется простым соотношением $x_{d+1} = x_1^2 + . + x_d^2$. Далее, необходимо построить выпуклую оболочку для полученных точек, и спроектировать обратно на d-мерное пространство. Полученная нижняя часть оболочки будет искомой триангуляцией.

Теорема 2. *Триангуляцию на множестве S из N точек в евклидовом d-мерном пространстве* E^d $(d > 2)$ *можно построить параллельно-рекурсивным алгоритмом с использованием q процессоров* $(q \le N)$ *за время* $O(\log(N)/\max(2\ \log(q)+1)\cdot d\ K/(\max(k,1q-\log(q))),$ *(K – количество вершин выпуклой оболочки), с ускорением процедуры слияния* $R = \log(N) / \max(2,\log(q)+1)$.

Предложенный метод позволяет значительно улучшить суммарную сложность, заменив в оценке для классического решения N на K. Кроме того, при кажущейся одинаковой эффективности мы выигрываем на значительно меньшем значении коэффициента комплексности оценки в последнем случае (то есть $O(N) = k_2 N$, k_2 = const. $k_2 << k_1$, где κ_2 – коэффициент триангуляции, k_1 – коэффициент построения оболочки). Этот факт очевиден, достаточно сравнить соответствующие шаги выполнения в обоих случаях. Еще одно преимущество такой реализации в том, что не нужно создавать новый массив точек для триангуляции, все можно реализовать на базе основных, на которых строим выпуклую оболочку.

Как и в случае с триангуляцией, для построения диаграммы Вороного используются на основных этапах построения

процедуры слияния результаты предыдущих задач: триангуляции и выпуклой оболочки. Нас здесь больше интересует решение, вытекающее из построения триангуляции. Здесь не нужна перестройка в полном объеме. Достаточно получить триангуляцию в пространстве той же самой размерности и на ее основе построить диаграмму Вороного. В таблице приведен общий анализ оценок сложности для рассмотренных задач.

Общий анализ эффективности алгоритма

Параллельность / Алгоритм	Один процессор	Q процессоров
Предобработка	$O(NlogN)$	$O(logN/max(log(Q)+1,2)N)$
Выпуклая оболочка	$O(Nd^2logK)$	$O(log(N)/(max(2,log(Q)+1)d(K/2+ + dK/max(K,1,Q-log(Q))))$
Триангуляция	$O(logN \cdot d \cdot K)$	$O(log(N)/max(2, log(Q)+1) \cdot d \cdot K/(max(k,1,Q-log(Q)))$
Диаграмма Вороного	$O(N \cdot d^2)$	$O(N \cdot d2/max(q,N))$
Суммарная	$O(N(d^2logK+logN)$	O (предобработка + построение оболочки)

Из сравнительного анализа оценок следует, что для всех задач достигается высокая эффективность, за исключением предобработки. Важно отметить, что предложенный алгоритм в общем способен эффективно использовать количество процессоров даже большее, чем N. Это достигается благодаря, во-первых, использованию процедур слияния, построенных единым способом, что позволяет выполнять ряд расчетов независимо и, во-вторых, наличие нескольких задач позволяет получить дополнительное распараллеливание на межзадачном уровне (пока решается определенная стадия одной из текущих задач, другие могут выполнять следующие стадии).

Реализация алгоритма. Для распараллеливания алгоритма реализована система управления процессами, которая совместима как с системами, которые работают с общей памятью, так и с распределенной. Более сложным по структуре и реализации является второй случай. Там особого выбора нет, система использует MPI на низком уровне. Для первого случая было решено разработать свой механизм управления, но так, чтобы интерфейсы управления в обоих случаях были похожими для обеспечения максимальной простоты перехода от одной парадигмы к другой. Необходимо также отметить, что эффективность использования функциональности реализации лежит на разработчике в рамках используемой им парадигмы. Кроме этого, для «последовательной части» функциональности имеется возможность создавать виртуальные процессоры, которые будут псевдопараллельно нагружать определенный процессор.

На этапе предварительной обработки нужно выполнить начальную обработку входных данных. В нашем случае это лишь сортировка точек. В работах [17–19] предложены эффективные параллельные алгоритмы сортировки. Для реализации разработанного алгоритма мы выбрали сортировку слиянием. Этот алгоритм простой и показывает хорошие результаты на больших объемах данных. Он имеет меньшее ускорение на большом количестве процессоров, но не требователен к архитектуре памяти. Был проведен анализ производительности разработанного алгоритма в сравнении с существующими последовательными реализациями. Вычислительная система реализации алгоритма исследовалась для множества точек от 10^5 по 10^6 с шагом 10^5. На рис. 1 показаны графики временной зависимости работы для последовательного алгоритма (ПА), параллельно-рекурсивного алгоритма (П-РА) и предварительной обработки (ПО). На рис. 2 представлены результаты построения выпуклой оболочки и триангуляции с помощью предложенного алгоритма.

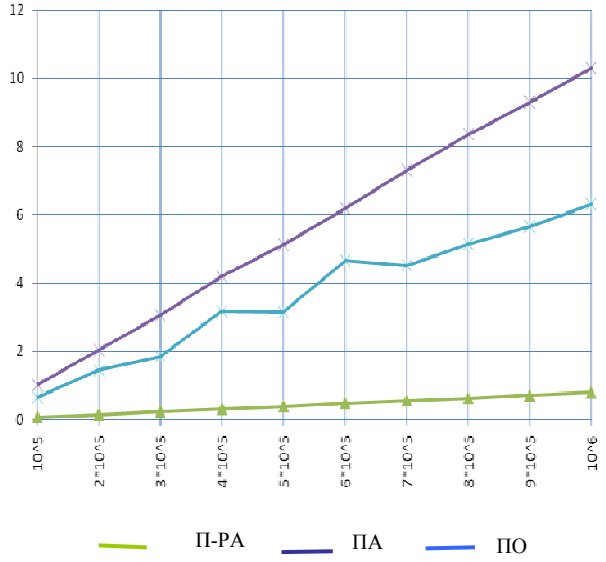


Рис. 1. График временной зависимости от количества точек

Рис. 2. Выпуклая оболочка и триангуляция для точек, равномерно распределенных в кубе

Таким образом, был разработан параллельно-рекурсивный алгоритм построения комплекса взаимосвязанных задач вычислительной геометрии (выпуклой оболочки, триангуляции, диаграммы Вороного и других задач) для пространств произвольной размерности. Получена улучшенная эффективность решения как

отдельных задач, так и их совокупности. Установлена эффективность алгоритма и предложена его практическая реализация для построения выпуклой оболочки и триангуляции в 3-мерном пространстве. Характерной чертой практической реализации предложенного подхода является то, что разработанный алгоритм позволяет одновременно решать как разные шаги одной процедуры задачи на многих процессорах, так и разные процедуры в одном узле алгоритма с помощью технологии MPI. Еще одна особенность этой модели алгоритма заключается в том, что предобработка и процесс разбиения множества точек являются общими для решения рассматриваемого множества задач; отличаться будут лишь процедуры на этапе слияния.

Список литературы

1. Parallel computational geometry / A. Aggarwal [et al.] // Algorithmica. – 1988. – No. 3. – P. 293–327.

2. Atallah M.J., Cole R., Goodrich M.T. Cascading divide-and-conquer: A technique for designing parallel algorithms // SIAM J. Comput. – 1989. – No. 18. – P. 499–532.

3. Cole R., Goodrich M.T. Optimal parallel algorithms for polygon and point-set problems // Algorithmica. – 1992. – No. 7. – P. 3–23.

4. Amato N.M., Goodrich M.T., Ramos E.A. Parallel algorithms for higher-dimensional convex hulls // In Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. – 1994. – P. 683–694.

5. Chen D. Efficient geometric algorithms on the EREW PRAM // IEEE Trans. Parallel Distrib. Syst. – 1995. – No. 6. – P. 41–47.

6. Berkman O., Schieber B., Vishkin U. A fast parallel algorithm for finding the convex hull of a sorted point set // Internat. J. Comput. Geom. Appl. – 1996. – No. 6. – P. 231–242.

7. Goodman J.E., O'Rourke J. Handbook of Discrete and Computational Geometry. – N.Y.: Chapman and Hall/CRC Press, 2004. – 1497 p.

8. Akl S.G., Lyons K.A. Parallel Computational Geometry. – Englewood Cliffs: Prentice-Hall, 1993. – 224 p.

9. JaJa J. An Introduction to Parallel Algorithms. – Amsterdam: Addison-Wesley, 1992. – 566 p.

10. Van Leeuwen J. Handbook of Theoretical Computer Science. – Amsterdam: Elsevier/The MIT Press, 1990. – 1294 p.

11. Reif J.H. Synthesis of Parallel Algorithms. – San Mateo: Morgan Kaufmann, 1993. – 1011 p.

12. Ghouse M. and Goodrich M. Fast randomized parallel methods for planar convex hull construction. Comput. Geom. Theory Appl., 7: 219{236, 1997.

13. Шеймос М., Препарата Ф. Вычислительная геометрия: Введение. – М.: Мир,1989. – 478 с.

14. Shamos M.I., Hoey D. Closest-point problems. In Proc. 16th IEEE Sympos. Found. Comput. Sci pages 151–162, 1975.

15. Chen T.M. A minimalist's Implementation 3-d Divide-and-Conquer Convex Hull Algorithm. School Computer Science University Waterloo. – 2003. – 12 с.

16. Скворцов А.В. Триангуляция Делоне и ее применение. – Томск: Изд-во Томск. ун-та, 2002. – 128 с.

17. Cole R. and Goodrich M.T. Optimal parallel algorithms for polygon and point-set problems. Algorithmica, 7:3–23, 1992.

18. Reif J.H. and Sen S. Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. SIAM J. Comput., 21:466–485, 1992.

19. Akl S.G., Lyons K.A. Parallel Computational Geometry. Prentice-Hall, Englewood Cliffs, 1993.

Г.Г. Исламов, А. Г. Исламов

Удмуртский государственный университет, г. Ижевск

ОБ ОДНОМ КЛАССЕ ОРТОГОНАЛЬНЫХ ПРОЕКТОРОВ

Пусть $B = H_n$ и H_μ – конечномерные вещественные гильбертовы пространства размерности соответственно ν и μ, причём $\nu << \mu$. Пусть, далее, $\Lambda: B \to H_\mu$ – линейное инъективное отображение и $\Lambda^*: H_\mu \to B$ – сопряжённый с ним оператор. Следующее утверждение очевидно.

Лемма. *Самосопряжённый оператор* $S = \Lambda^* \Lambda : B \to B$ *положительно определён и обратим,* $P = \Lambda S^{-1} \Lambda^* : H_\mu \to H_\mu$ – *ортогональный проектор. Кроме того, операторы* $\delta = S^{-1} \Lambda^* : H_\mu \to B$ *и* $\delta^* = \Lambda S^{-1} : B \to H_\mu$ *есть обобщённые обратные отображения к операторам соответственно к* Λ *и* Λ^*, *т.е.* $\Lambda\delta\Lambda = \Lambda$ *и* $\Lambda^*\delta^*\Lambda^* = \Lambda^*$.

Введём каноническое представление произвольного элемента $x \in H_\mu$.

Имеем

$$x = Px + (I - P)x = \Lambda\delta x + \sum_{j=1}^{\mu-\nu} u_j r_j(x).$$

Нетрудно видеть, что система элементов $u_1, \ldots, u_{\mu-\nu}$ в этом представлении принадлежит пространству H_μ, образует базис ядра оператора δ и биортогональна системе функционалов $r_1, \ldots, r_{\mu-\nu}$. Далее, так как $I - P$ есть ортогональный проектор, то в качестве второго слагаемого канонического разложения можно взять спектральное разложение этого проектора, т.е. считать, что система $u_1, \ldots, u_{\mu-\nu}$ ортонормирована и $r_j(x) = (x, u_j)$, $j = 1, \ldots, \mu - \nu$. Всюду $(\cdot,\cdot)$ обозначает скалярное произведение в гильбертовом пространстве H_μ. Ввиду условия $\nu << \mu$ разность $\mu - \nu$ может оказаться достаточно большой. Поэтому целесообразно ввести более узкое пространство

$$D = \{x \in H_\mu : x = \Lambda\delta x + \sum_{j=1}^{n} u_j r_j(x)\},$$

в котором мы будем далее рассматривать линейные уравнения с дополнительными линейными ограничениями. Нетрудно видеть, что это пространство имеет размерность $n + \nu$ $(n \le \mu - \nu)$ и получается в результате расширения образа ΛB оператора Λ за счёт первых *n* элементов ядра оператора δ. При этом элементы $u_1, \ldots, u_m$ образуют базис ядра сужения оператора δ на линей-

ное многообразие D. В пространстве D введём норму $\|x\|_D = \|\delta x\|_B + \sum_{j=1}^{n} |r_j(x)|$. Каноническая форма представления элементов из D позволяет получить канонические представления для линейных операторов и функционалов, определённых на D. Пусть $L: D \to B$ есть линейный оператор. В силу указанной выше факторизации тождественного оператора построенного пространства D имеет место следующее представление:

$$Lx = Q\delta x + \sum_{j=1}^{n} q_j r_j(x),\ x \in D.$$

Здесь $Q = L\Lambda: B \to B$ есть линейный оператор, а элементы $q_j = Lu_j, j = 1,...,n$ принадлежат гильбертову пространству B.

Приведём критерий разрешимости в пространстве D уравнения $Lx = f$ с линейными неравенствами $l_i(x) \geq \beta_i, i = 1,...,m$, где $l_i \in D^*, f \in B$. В терминах канонических представлений оператора L и функционалов

$$l_i(x) = (\delta x, \varphi_i) + \sum_{j=1}^{n} \alpha_{ij} r_j(x),\ \varphi_i = l_i \Lambda \in B^*,\ (\delta x, \varphi_i) = \varphi_i(\delta x),$$

$\alpha_{ij} = l_i(u_j)$ имеет место следующий аналог теоремы Фредгольма о разрешимости уравнения с линейными неравенствами [3]. Пусть вещественные числа $\beta_i, i = 1,...,m$ таковы, что система неравенств $l_i(x) \geq \beta_i, i = 1,...,m$ совместна.

Теорема. *Для того чтобы существовало решение $x \in D$ уравнения $Lx = f$ с линейными неравенствами $l_i(x) \geq \beta_i, i = 1,...,m$, необходимо и достаточно, чтобы для любых $y \in B$ и неотрицательных $\lambda_1,...,\lambda_m$, удовлетворяющих системе уравнений*

$$\begin{cases} Q^* y = \sum_{i=1}^{m} \lambda_i \varphi_i, \\ (q_i, y) = \sum_{i=1}^{m} \lambda_i \alpha_{ij},\ j = 1,...,n, \end{cases}$$

выполнялось неравенство

$$(f, y) \geq \sum_{i=1}^{m} \lambda_i \beta_i .$$

Рассмотрим теперь задачу на собственные значения для проектора $P = \Lambda S^{-1} \Lambda^* : H_\mu \to H_\mu$. При численных расчётах все операторы заменяются представляющими их матрицами. Спектр P состоит из нуля и единицы. Поэтому собственные векторы находятся из уравнений $\Lambda^* u = 0$ и $Pu = u$. Из первого уравнения находим необходимую нам систему линейно независимых элементов $u_1, ..., u_n$ из H_μ. Для этого к транспонированной матрице Λ^* применяется алгоритм поиска базисного минора матрицы [2], в основе которого лежит универсальная операция над матричными структурами [1]. Все решения второго уравнения $Pu = u$ даются формулой $u = Pz$, где z – произвольный вектор из H_μ. Следовательно, для произвольного базиса $f_1, ..., f_\nu$ пространства B решения $x_i = \Lambda f_i$ в пространстве D уравнений $\delta x = f_i, r_j(x) = 0, j = 1, ..., n; i = 1, ..., \nu$ дают базис собственного подпространства, отвечающего единице, самосопряженной матрицы P. Численные расчёты матриц S^{-1}, P и базисных векторов $u_1, ..., u_n$ выполнены на системе графических процессоров, поддерживающих технологию многопоточного программирования CUDA [4].

Список литературы

1. Исламов Г.Г. Универсальная операция над матричными структурами // Современные проблемы вычислительной математики и математической физики: тез. докл. междунар. конф., Москва, МГУ имени М.В. Ломоносова, 16–18 июня 2009 г. – М.: Изд-во МГУ: Макс Пресс, 2009. – 396 с.

2. Исламов Г.Г., Коган Ю.В. Об одном алгоритме поиска базисного минора матрицы // Вестн. Удмурт. ун-та. – 2006. – № 1. Математика. – С. 63 – 70.

3. Исламов Г.Г. Критерий разрешимости уравнений с краевыми неравенствами // Изв. Ин-та математики и информатики. 1994. Вып. 2. – Ижевск: Изд-во Удмурт. гос. ун-та. – С. 3–24.

4. NVIDIA CUDA Programming Guide Version 3.0. – URL: http://www.nvidia.com.

К.С. Исупов

Вятский государственный университет, г. Киров

СОВРЕМЕННЫЙ ПОДХОД К РЕШЕНИЮ «НЕКОРРЕКТНЫХ» МАТЕМАТИЧЕСКИХ ЗАДАЧ

Используемые в современных CAD-системах и программных комплексах математические методы и их алгоритмически-программные решения разрабатывались для вычислительных систем с ограниченным количеством вычислительных модулей. Адаптация применяемых сегодня методов, алгоритмов и их программных реализаций для решения прикладных задач даже на 1000-ядерные вычислительные системы является весьма сложной и в ряде случаев невыполнимой задачей. В связи с этим решение ряда современных прикладных задач высокой размерности и высокой вычислительной сложности становится невозможным.

В последнее время наиболее важным становится требование максимально быстрого выполнения операций над числами, превышающими максимум типового компьютерного диапазона серийной вычислительной техники. Данные числа также принято называть *большими числами*.

Вычисления с *большими числами* являются одной из областей, в которых хорошо разработанные в настоящее время позиционные методы являются неэффективными. Важные для теории и практики математические задачи, требующие таких вычислений и больших вычислительных ресурсов, лежат в областях прикладной и вычислительной теории чисел [1]. Боль-

шинство таких задач содержат вычисления с числовыми величинами, принимающими значения из больших и сверхбольших машинных диапазонов. В настоящее время возникает широкий спектр вычислительных задач [2,3], приводящих к вычислениям, при которых значения числовых данных значительно, в $10^3 \ldots 10^6$ раз, превышают максимум типового компьютерного диапазона серийной вычислительной техники, определяемого длиной аппаратно-поддерживаемого машинного слова, равной в настоящее время 32 или 64 бита.

Ниже приведен ряд задач, часто называемых из-за их сложности вычислительными проблемами [2]:

- тестирование на простоту чисел специального и произвольного вида;
- поиск больших и сверхбольших простых чисел вида $4k+1$;
- поиск псевдопростых чисел и чисел близнецов;
- поиск цепочек простых чисел в арифметических прогрессиях;
- проверка местоположения нулей дзета-функции Римана;
- определение свойств и характера поведения чисел Мерсенна в сверхбольшом компьютерном диапазоне [1];
- обращение и нахождение определителя матриц Гильберта или асимптотически приближающихся к ним матриц;
- решение плохо обусловленных систем линейных алгебраических уравнений (СЛАУ) большого порядка.

Данный перечень не является исчерпывающим. Характерной особенностью указанных в нем задач является невозможность их решения только аналитическими или алгебраическими методами, и по этой причине находят широкое использование вычислительные методы при поиске полного или частичного решения.

В большинстве случаев при решении данных задач даже незначительные возмущения коэффициентов приводят к существенным погрешностям, делающим все решение бессмысленным. Поэтому данные задачи также принято называть *некорректно поставленными* (далее «*некорректными*»).

Решения перечисленных задач имеют большое теоретическое значение. На данный момент даже частные решения, полученные вычислительными методами, находят широкое практическое приложение [3].

Задача решения плохо обусловленных СЛАУ. Наибольший интерес из приведенного списка представляют задачи эффективного решения плохо обусловленных СЛАУ большого порядка и обращение матриц Гильберта. Рассмотрим данные задачи более подробно.

Говоря о численном решении СЛАУ, важно затронуть вопрос о точности используемых алгоритмов. Например, в широко известном математическом пакете Mathcad 12 в случае хорошо обусловленных систем точность функций, решающих СЛАУ, очень высока. Ответ обычно верен до 14–15 знака мантиссы, что близко к предельной теоретической точности численных методов на машинах с 64-битовыми числами с плавающей точкой. Однако с увеличением размеров системы точность обычно падает. Также крайне важно, насколько разнородные уравнения образуют систему. Если коэффициенты двух или нескольких уравнений близки, то при малых возмущениях в матрице коэффициентов A или векторе правых частей B произойдут большие изменения в векторе неизвестных $X = A^{-1} \cdot B$. Это означает, что численный метод будет неустойчив и, следовательно, резко повысится вероятность получения ответа, содержащего большую погрешность ξ [4].

Таким образом, возмущение коэффициентов системы линейных уравнений может не просто немного исказить ее решение, а иметь более серьезные последствия. Например, система линейных уравнений

$$\begin{cases} X + 0{,}99\,Y = 1{,}01 \\ X + 1{,}01\,Y = 0{,}99 \end{cases}$$

имеет единственное решение, но за счет изменения коэффициентов и свободных членов всего лишь на 1% можно получить как противоречивую систему, так и систему, имеющую бесконечно много решений.

Системы, содержащие практически линейно зависимые уравнения, называются *плохо обусловленными*. Количественной

мерой обусловленности может являться определитель: чем он ближе к нулю, тем хуже обусловлена система. Однако есть и более объективная характеристика, называемая числом обусловленности (condition number, *CN*). Число обусловленности определяется как произведение нормы матрицы коэффициентов *A* на норму обратной ей матрицы. Чем больше число обусловленности, тем хуже обусловлена система и тем выше вероятность получения ответа со значительной погрешностью ξ.

Классической плохо обусловленной системой линейных уравнений является система, матрица коэффициентов которой является *матрицей Гильберта*. Матрица Гильберта *A* – это матрица, значение элементов которой определяется формулой согласно выражению $A_{ij}=1/(1+i+j)$, где *i* и *j* – индексы элемента.

Очевидно, что чем больше будет матрица Гильберта, тем в меньшей степени будут отличаться нижние ряды. Соответственно, тем хуже будет обусловлена матрица, а следовательно, и сама СЛАУ.

В табл. 1 приведена зависимость определителя и числа обусловленности *CN* матрицы Гильберта от ее порядка.

Таблица 1

Зависимость определителя и числа обусловленности *CN* матрицы Гильберта от ее порядка

Порядок матрицы, *N*	Определитель, $\lvert A(N)\rvert$	Число обусловленности, $CN(A(N))$
2	$8,\ 3\cdot10^{-2}$	19,333
3	$4{,}63\cdot10^{-4}$	526,159
4	$1{,}653\cdot10^{-7}$	$1{,}561\cdot10^{4}$

С помощью пакета Mathcad 12 было произведено решение ряда плохо обусловленных СЛАУ с матрицами Гильберта в качестве коэффициентов. Результаты решения представлены в табл. 2. На основании табл. 1 был построен график зависимости определителя матрицы от ее порядка, представленный на рис. 1.

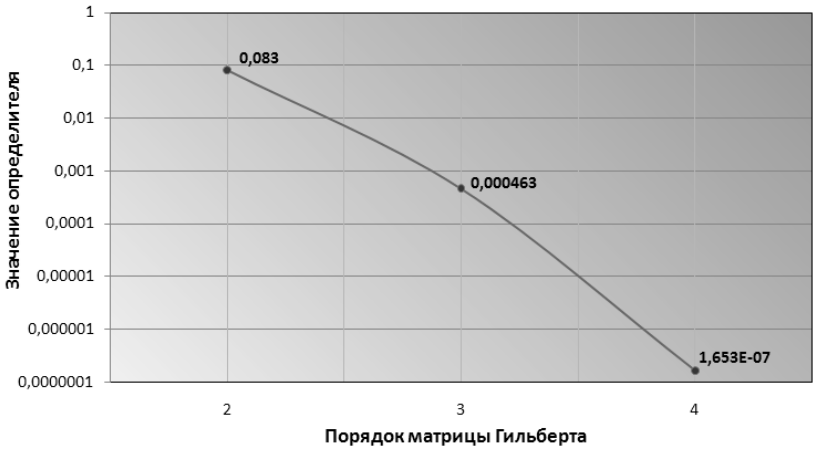


Рис. 1. График зависимости определителя матрицы Гильберта от ее порядка

Таблица 2

Зависимость погрешности ξ и числа обусловленности *CN* от порядка СЛАУ

Порядок матрицы Гильберта, N	Число обусловленности, $CN(A(N))$	Погрешность определения корней, ξ
3	$5{,}3\cdot10^{2}$	$2{,}5\cdot10^{-13}$
5	$4{,}8\cdot10^{5}$	$7{,}2\cdot10^{-11}$
7	$4{,}8\cdot10^{8}$	$3{,}3\cdot10^{-7}$
13	$5{,}7\cdot10^{17}$	$7{,}3\cdot10^{2}$

Как видно из табл. 2, с увеличением порядка матрицы происходит резкое увеличение числа обусловленности и погрешности работы численного метода пакета Mathcad 12. Погрешность ξ определения корней может составлять сотни и тысячи процентов, тем самым делая результат решения СЛАУ абсолютно бесполезным. В рассмотренном случае это наблюдается, когда порядок СЛАУ достигает 13. Задача решения плохо обусловленных СЛАУ – классическая «*некорректная*» задача.

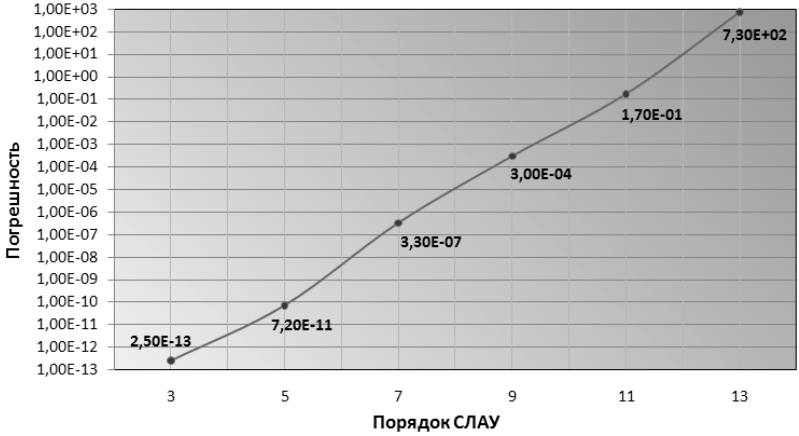


Рис. 2. Зависимость погрешности ξ определения корней СЛАУ от ее порядка

На основании табл. 2 построен график зависимости погрешности ξ определения корней от порядка матрицы Гильберта, наблюдаемой при решении соответствующей данной матрице СЛАУ. График данной зависимости представлен на рис. 2.

Анализируя представленный график можно заметить, что при незначительном увеличении порядка решаемой СЛАУ происходит экспоненциальное уменьшение определителя матрицы ее коэффициентов, а следовательно, такое же увеличение погрешности ξ определения корней (на графике представление зависимости искажено ввиду использования логарифмической шкалы). Это, в свою очередь, позволяет объяснить неэффективную работу в данной области соответствующих методов современных математических пакетов, например Mathcad 12.

Важно подчеркнуть, что матрица Гильберта не определяет однозначно класс плохо обусловленных СЛАУ, который является значительно более объемным. В общем случае, для того чтобы охарактеризовать СЛАУ как плохо обусловленную систему, достаточно асимптотического приближения матрицы ее коэффициентов к матрице Гильберта [5].

Задача обращения матрицы Гильберта. В данном контексте следует также рассмотреть задачу обращения матрицы Гильберта. Точная обратная матрица Гильберта – это матрица с очень большими целочисленными значениями. Данная задача наряду с задачей нахождения корней плохо обусловленных СЛАУ является сложно решаемой современными CAD-системами.

Например, в пакете Matlab функция invhilb(N) возвращает матрицу, обратную матрице Гильберта порядка N (N<15). Для N>15 функция invhilb(N) возвращает приближенную матрицу. Значения обратной матрицы могут быть представлены как числа с плавающей запятой без погрешности округления до тех пор, пока порядок матрицы Гильберта N не превышает 15. Далее, с увеличением порядка матрицы погрешность ξ решения задачи возрастает согласно зависимости, изображенной на рис. 2.

Современный подход к решению «некорректных» задач. Одним из перспективных направлений в решении поставленных задач является внедрение методов вычислений на основе нетрадиционных способов кодирования и соответствующих им вариантов машинной арифметики. Особую роль в развитии указанного направления играют числовые системы с параллельной структурой, и в первую очередь модулярные системы счисления, в которых целые числа представляются наборами остатков от деления на выбранные модули (основания).

Одной из самых известных таких систем является полиномиальная система классов вычетов, также известная как система остаточных классов (СОК). Здесь в отличие от обобщенной позиционной системы счисления образование цифр каждого разряда проводится независимо друг от друга.

Каждый разряд в системе остаточных классов представляет собой наименьший неотрицательный остаток от деления на соответствующее основание самого числа, а не предыдущего частного, как это имеет место в позиционной системе.

Если же касаться задач матричной алгебры, то здесь каждый массив, представленный в позиционной системе, в СОК будет представляться множеством независимых друг от друга массивов, элементы которых представляют собой наименьшие

неотрицательные остатки от деления на выбранные основания соответствующих позиционных элементов.

Остаток от деления меньше делителя, поэтому если в качестве оснований СОК выбрать числа, не выходящие за пределы разрядной сетки ЭВМ, то и вычеты, взятые по этим основаниям, не будут выходить за пределы разрядной сетки. К тому же выполнение операций в полиномиальной системе практически ничем не отличается от выполнения операций в традиционных позиционных системах счисления.

Таким образом, становится возможным параллельное решение «*некорректных»* задач без потери точности.

В результате данного обзора справедливо будет сказать, что заложенные в наиболее распространенные современные математические системы позиционные методы для выполнения вычислений над полем больших чисел способны лишь частично решать поставленные перед ними задачи, т.е. выдавать корректное решение при ограниченной, как правило, небольшой размерности входных данных. С учетом богатых возможностей распределения вычислений при решении рассмотренных задач, обуславливаемых свойствами матричных операций, ставится задача разработки эффективных параллельных методов, алгоритмов и средств на их основе, обеспечивающих решение «*некорректных»* задач в более широком диапазоне представления их размерности.

Предложен метод решения рассмотренных задач, основанный на использовании в качестве базиса для расчетов мощного механизма для выполнения высокоточных параллельных вычислений системы остаточных классов.

Список литературы

1. Рибенбойм П. Рекорды простых чисел // Успехи математических наук. 1987. – Т. 42. Вып. 5, 1987. – С. 119–176.

2. Lenstra H.W., Tijdeman R.J. Computational Methods in Number Theory. –Amsterdam: Math. Cent., 1982. –198 p.

3. Инютин С.А. Модулярные вычисления в сверхбольших компьютерных диапазонах // Изв. вузов. Электроника. – 2001. – № 6. – С. 34–39.

4. Гурский Д., Турбина Е. Вычисления в MATHCAD 12. – СПб.: Питер, 2006. – 544 с.

5. Петров И.Б., Лобанов А.И. Введение в вычислительную математику. – URL: http://www.intuit.ru/department/calculate/calcmathbase.

А.В. Карзов

Запорожский национальный технический университет, Украина

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ

Неотъемлемыми атрибутами современной вычислительной техники становятся программно-аппаратные средства цифровой обработки изображений. Высококачественные оцифрованные изображения в несжатом виде представляют собой огромные массивы данных, для хранения которых требуются значительные объёмы памяти ЭВМ. Для уменьшения объёмов хранимых и передаваемых данных используются различные методы их сжатия.

Алгоритм фрактального сжатия изображений является одной из наиболее перспективных альтернатив алгоритмам, использующим дискретное косинусоидальное преобразование (ДКП), таким как JPEG. В силу своих преимуществ именно фрактальное сжатие применяется в ряде узкоспециализированных задач, важнейшей из которых является передача изображений со спутников (в этом случае преимущество достигается за счет лучшего качества изображений, сжатых с коэффициентом компрессии больше 100, по сравнению с ДКП-алгоритмами). Еще одно преимущество – возможность применения к сжатым изображениям фрактального масштабирования.

Алгоритм фрактального сжатия изображений. Основа метода фрактального кодирования – обнаружение самоподобных участков в изображении. Впервые возможность применения теории систем итерируемых функций (IFS) к проблеме сжатия изображения была исследована Майклом Барнсли [1] и Аланом Слоуном, которые запатентовали свою идею в 1990 и 1991 гг. Джеквин представил метод фрактального кодирования, в котором используются системы доменных и ранговых блоков изображения, блоков квадратной формы, покрывающих все изображение. Этот подход стал основой для большинства методов фрактального кодирования, применяемых сегодня.

Алгоритм данного метода:

1. Изображение разбивается на ранговые и доменные блоки, причем сторона домена в два раза больше стороны ранговой области. Ранговые блоки должны покрывать все изображение целиком, а доменные могут располагаться произвольным образом и не покрывать всего изображения.

2. Домены уменьшаются до размеров ранга путем усреднения яркости соседних пикселей.

3. Для каждого ранга подбирается наиболее похожий на него домен после преобразования, минимизирующего разницу между рангом и доменом, путем перебора всех доменов или их части. Преобразование включает изменение контраста и яркости домена.

Для домена, уменьшенного до размеров ранга, преобразование имеет вид

$$T(D) = sD + oI ,$$

где $|s|<1$ – коэффициент контраста; $o \in [-255;255]$ – коэффициент яркости; I – единичная матрица.

Налагаемые ограничения необходимы для того, чтобы гарантировать сходимость при восстановлении изображения.

4. Вычисляются коэффициенты преобразования контраста s и яркости o по формулам

$$s = \frac{n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j} - \sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j} \sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j}}{n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}^2 - \left(\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}\right)^2},$$

если

$$n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}^2 - (\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j})^2 = 0,$$

то $s = 0$;

$$o = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j} - s\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}}{n^2},$$

где D и R – матрицы, представляющие собой соответственно домен, уменьшенный до размера ранга, и ранг; n – размер стороны рангового блока.

5. На основе рассчитанных коэффициентов преобразования оценивается соответствие домена и ранга по формуле

$$r = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j}^2 + s\left(s\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}R_{i,j} + 2o\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}\right) + o\left(on^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j}\right)}{n^4}.$$

Наиболее оптимальный домен для данного ранга определяется путем перебора всех доменных областей.

Данный алгоритм выполняется для всех ранговых блоков изображения. В результате для каждого ранга определяются координаты наиболее похожего на него домена вместе с коэффициентами преобразования доменного блока в ранговый, минимизирующими разницу между ними.

Ускорение фрактального сжатия изображений. Основным недостатком фрактального алгоритма является то, что требуются значительные вычислительные ресурсы для подбора доменного блока соответствующего ранговому блоку.

Для решения данной проблемы предлагается разбить исходное изображение размера на подизображения, состоящие из

определенного количества ранговых блоков. Эти подизображения вычисляются на отдельных процессорах, при этом производится поиск доменных блоков только для тех ранговых блоков, которые входят в подизображение.

Подизображения создаются исходя из среднего количества ранговых блоков, приходящихся на один процессор:

$$A = N / m ,$$

где N – количество ранговых блоков в изображении; m – количество процессоров.

В случае если не удается разделить изображение поровну на все процессы, то количество ранговых блоков, получаемых последним процессом, определяется по формуле

$$A_m = N - \left| N / m \right| * (m - 1) ,$$

для остальных

$$A_{1..m-1} = \left| N / m \right| .$$

Основной процесс загружает изображение и создает структуру данных, в которой каждый ранговый блок отмечается как обработанный и необработанный. Делит ранговые блоки для всех процессов и рассылает их по соответствующим процессам. В дальнейшем основной процесс ждет сигнализации от других процессов о выполнении своей части работы. Если приходит такая сигнализация, основной процесс проверяет структуру данных на наличие необработанных ранговых блоков и в случае наличия таких отправляет отработавшему процессу новый ранговый блок, предварительно отмечая его как пройденный.

Когда все процессы сигнализируют, что они отработали и нет необработанных ранговых блоков, основной процесс производит составление выходного сжатого изображения.

Рассматривая каждый новый ранговый блок, дочерний процесс проверяет структуру данных, т.е. выясняет, обрабатывался ли данный блок, и если не обрабатывался, производит поиск соответствующих ему доменных блоков. Обработав все свои

ранговые блоки, дочерний процесс сигнализирует об этом основному процессу.

В работе представлен параллельный алгоритм фрактального сжатия изображений, который позволяет значительно ускорить выполнение сжатия изображения, особенностью которого является равномерное разбиение задачи фрактального сжатия для последующей параллельной (распределенной) обработки.

Область применения разработанного алгоритма включает в себя предварительное сжатие изображений для передачи каналам связи с ограниченной пропускной способностью (например, спутниковая съемка), уменьшение объема графической информации, передаваемой по локальным сетям и сети Интернет, сжатие с возможностью восстановления изображения большего размера с меньшим количеством шумов и артефактов заметных для человека (полноформатная полиграфия).

Список литературы

1. Barnsley M. Fractals everywhere – Boston, Academic Press, 1993. – P. 103–114.

2. Уэлстид С. Фракталы и вейвлеты для сжатия изображений в действии: учеб. пособие. – М.: Триумф, 2003. – 320 с.

3. Hammerle J., Uhl A. Fractal Image Compression on MIMD Architectures II: Classification Based Speed-up Methods // Journal of Computing and Information Technology. – 2000. – P. 71–82.

4. Jackson D.J., Mahmoud W. Parallel Pipelined Fractal Image Compression using Quadtree Recomposition // The computer journal. – 1996 – No. 1. Vol. 39. – P. 1–13.

5. Илюшин С.В., Свет С.Д. Фрактальное сжатие телемедицинских изображений // Электросвязь. – 2009. – № 4. – С. 36–40.

С.Н. Карпенко, А.В. Дергунов

Нижегородский государственный университет им. Н.И. Лобачевского

АВТОМАТИЗАЦИЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ MPI-ПРИЛОЖЕНИЙ

Цель параллельных вычислений – решить задачу быстрее. Чтобы определить, где именно тратится время на вычисления в параллельной программе, т.е. найти узкие места производительности, используются инструменты для измерения и визуализации производительности. Пример такого инструмента – Intel Trace Analyzer and Collector, который обеспечивает визуализацию трассы выполнения программы. Но трасса параллельной программы зачастую бывает очень большой, поэтому ее сложно анализировать вручную. Разработанная автором система позволяет автоматизировать повышение производительности в MPI-программах.

Повышение производительности MPI приложений. Процесс повышения производительности MPI-программ с использованием измерительных инструментов, собирающих трассу выполнения приложения, состоит из следующих задач:

1. Выбор метрик или параметров оценки производительности программы. В данной работе в качестве такой метрики рассматривается общее время работы программы.
2. Сбор данных производительности.
3. Анализ данных производительности программы на предмет выявления проблем производительности.
4. Выявление причин возникновения проблем производительности.
5. Принятие решения о способе устранения проблем производительности
6. Устранение причин возникших проблем производительности.

Этот процесс может повторяться до достижения требуемой производительности. Разработанная автором система предназначена для автоматизации 3, 4 и 5 задач.

Для повышения производительности MPI-приложений система анализирует коммуникационный профиль приложения [1]. Длина сообщений, интенсивность передачи сообщений в разные моменты времени работы программы, топология передач сообщений между процессорами, накладные расходы на организацию взаимодействия, масштабируемость коммуникационной структуры программы, структура синхронных и асинхронных посылок – эти и ряд других характеристик определяют коммуникационный профиль приложения. Повышенные накладные расходы MPI-программы могут быть препятствием к достижению хорошей масштабируемости и часто являются следствием плохо реализованного коммуникационного профиля.

Ниже перечислены некоторые способы улучшения коммуникационного профиля приложения и, как следствие, повышения производительности MPI-программ:

- сокращение количества пересылаемых данных и по возможности пересылка одного большого сообщения вместо нескольких маленьких;
- отправка данных как можно раньше, а также маскирование ожидания приема данных с использованием неблокирующих операций MPI;
- использование топологий MPI, чтобы эффективно распределить процессы между процессорами параллельной вычислительной системы;
- распараллелить вычисления и назначить процессам процессоры так, чтобы сбалансировать вычислительную нагрузку;
- в некоторых случаях имеет смысл продублировать вычисления определенных данных вместо его пересылки и т.д.

Общая архитектура системы. Схема работы системы представлена на рис. 1. Исходными данными является трасса выполнения MPI-приложения (полученная, например, с помо-

щью Intel Trace Collector). Трасса конвертируется в специальный формат, в котором выполнение MPI-приложения представлено набором операций и их параметров. Входные данные анализируются модулем автоматического анализа проблем производительности. Для этого используется описание проблем производительности на специальном языке. Результат анализа – список обнаруженных проблем производительности с указанием степени их влияния на общее время работы приложения.

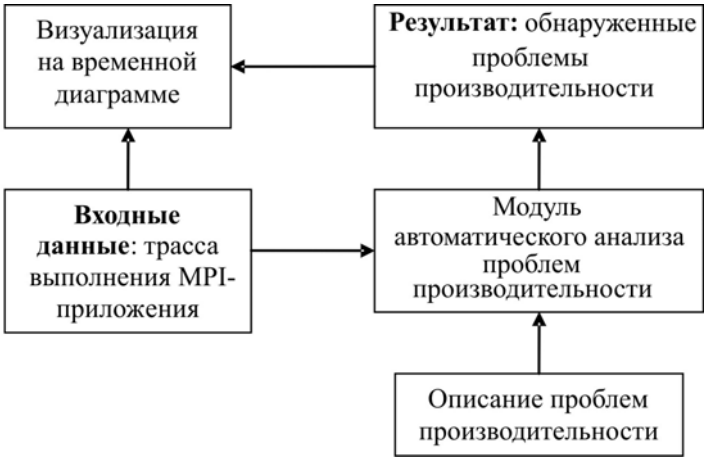


Рис. 1. Схема работы системы

Описание трассы выполнения MPI-приложения. В рамках разработанной системы результат запуска MPI-программы представляется с помощью общей статистики (общее время работы программы, количество процессов и т.д.) и набором коммуникационных операций, выполненных при ее работе, и параметров этих операций.

Ниже приведен простой пример описания работы MPI-приложения:

```
program
    total_time               2.471200748267274
    total_communication_time 2.003308519420723;
```

```
operation point_to_point
    send_op_name          "MPI_Ssend"
    send_op_type          "blocking"
    receive_op_name       "MPI_Recv"
    receive_op_type       "blocking"
    send_start_time       0.4687189432899993
    send_finish_time      2.4709433821505122
    receive_start_time    2.4695157509991628
    receive_finish_time   2.469596837040292
    send_process_rank     0
    receive_process_rank  1
    send_source_code      "latereceiver.cpp:20"
    receive_source_code   "latereceiver.cpp:31";
```

Здесь описывается программа с двумя процессами. За время работы приложения была осуществлена одна операция двухточечного обмена данными («operation point_to_point»). Процесс номер 0 переслал данные с помощью процедуры MPI_Ssend процессу номер 1, который принял эти данные с использованием процедуры MPI_Recv.

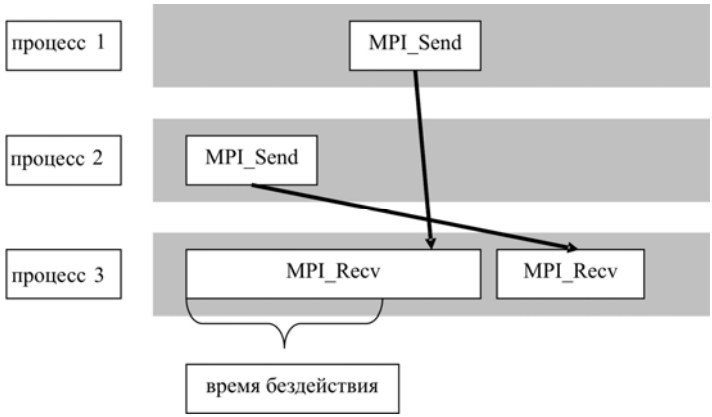


Рис. 2. Прием сообщений в неверном порядке

Язык описания типов проблем производительности в MPI-приложениях. В рамках системы разработан язык для описания типов проблем производительности, встречающихся в MPI-приложениях при использовании коммуникационных

операций. Рассмотрим в качестве примера проблему производительности «Прием сообщений в неверном порядке». Эта проблема возникает, когда порядок посылки и приема сообщений не совпадают. Из-за этого блокирующие операции приема затрачивают лишнее время на ожидание приема соответствующего сообщения. Схематично эта проблема представлена на рис. 2.

Рассмотрим, как описывается эта проблема производительности на разработанном языке:

```
    problem "Прием сообщений в неверном порядке"
        description "Порядок посылки и приема сообщений
не совпадают"
        find op1 type point_to_point
        find op2 type point_to_point
        where
            op1.receive_op_type eq "blocking" and
            op2.receive_op_type eq "blocking" and
            op1.receive_process_rank                    =
op2.receive_process_rank and
            op1.receive_start_time                      <
op2.receive_start_time and
            op1.send_start_time  >  op2.send_start_time
and
            op1.send_start_time                         >
op1.receive_start_time
        severity
            (op1.send_start_time                        -
op1.receive_start_time) /
            total_communication_time;
```

Здесь осуществляется поиск двух операций типа «point_to_point», которые удовлетворяют следующим условиям:

- тип принимающей процедуры обеих операций – блокирующий (т.е. процедура приема данных обязательно блокируется до начала посылки данных);
- прием данных в этих операциях осуществляется на одном и том же процессе;
- прием данных во время первой операции произошел раньше приема данных для второй операции;
- посылка данных во время первой операции, напротив, произошла позднее посылки данных для второй операции;

• в рамках первой операции посылка данных произошла позднее начала их приема, что обеспечивает ожидание.

Если проблема обнаружена, то для нее вычисляется степень ее влияния на производительность операций обмена в MPI-программе как разница между временем начала посылки данных и временем начала приема данных, поделенное на суммарное время, затраченное всеми процессами приложения при вызове коммуникационных операций MPI. Это значение указывает, в какой мере можно улучшить производительность программы, если избежать этой проблемы.

Предложены модель выполнения MPI-приложения и язык описания типов проблем производительности при взаимодействии процессов. Приведен пример описания на этом языке одной проблемы производительности: прием сообщений в неверном порядке. Использование предложенного языка позволяет автоматизировать анализ трассы MPI-программы и находить часто встречающиеся проблемы производительности. В разработанной на основе предложенной модели системе повышения производительности MPI-приложений предусмотрена возможность расширения количества анализируемых проблем с помощью описания новых типов операций и типов проблем производительности. Анализ степени влияния на производительность позволяет ранжировать обнаруженные проблемы и позволяет начать с решения наиболее важных из них. В [2] перечислены типы проблем производительности, которые были описаны на рассмотренном языке.

Список литературы

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

2. Карпенко С.Н., Дергунов А.В. Модели и программные средства повышения производительности MPI-приложений // Технологии Microsoft в теории и практике программирования: материалы конф. – Н.Новгород: Изд. Нижегород. гос. ун-та, 2009. – С. 188–192.

СОДЕРЖАНИЕ

Пленарные доклады

Петров В.Ю., Шевелев Н.А., Модорский В.Я.
Решение инженерных задач на высокопроизводительном вычислительном комплексе ПГТУ 4

Матвеенко В.П., Масич А.Г., Масич Г.Ф.
Региональная научно-образовательная оптическая сеть УрО РАН – фундамент киберинфраструктуры.....11

Снытников В.Н., Кукшева Э.А., Маркелова Т.В., Снытников Н.В., Стадниченко О.А., Стояновская О.П.
Суперкомпьютерные вычисления с экзафлопной производительностью в астрофизике и астрокатализе 21

Секционные доклады

Андреев Д.Ю., Джосан О.В.
Метод оптимизации ввода-вывода в приложениях для массивно-параллельных вычислительных комплексов 32

Андреев М.В., Галеев Э.Р., Штангеев А.Л., Юлдашев А.В., Яковлев А.А.
Опыт портирования геостатистического симулятора на архитектуру GPU средствами технологии PGI Accelerator 37

Багаутдинов Т.А., Гергель В.П., Горшков А.В., Фикс И.И., Кириллин М.Ю.
Моделирование распространения света в многослойной среде методом Монте-Карло 41

Балк П.И., Деменев А.Г., Долгаль А.С., Леденцов О.В., Мичурин А.В.
Эффективность применения высокопроизводительных вычислительных кластеров с целью оценки достоверности решения обратной задачи гравиметрии.....47

Барановский Н.В.
О подготовке специалистов по высокопроизводительным вычислениям для нужд МЧС.....55

Баркалов К.А., Рябов В.В., Сидоров С.В.
Оценки эффективности параллельного индексного метода глобальной оптимизации 62

Безгодов А.А., Иванов С.В., Косухин С.С.
Высокопроизводительный программный комплекс моделирования экстремальной динамики морских плавучих объектов.....70

Белобородов С.М.
Прикладная задача параллельных расчетов локальных дисбалансов динамически неустойчивых роторов с использованием кластерных систем 77

Бибиков С.А., Никоноров А.В., Фурсов В.А., Якимов П.Ю.
Рекуррентная обработка изображений в распределенной массивно-многопоточной CUDA-среде 80

Бояршинов М.Г., Балабанов Д.С.
Вычислительное моделирование движения сжимаемой среды с использованием высокопроизводительного вычислительного комплекса 88

Бражкин Д.Б.
Решение задач глобальной оптимизации на графическом процессоре.....96

Бутымова Л.Н., Модорский В.Я., Шмаков А.Ф., Гайнутдинова Д.Ф.
Анализ колебательных режимов акустического трансформатора 104

Бутымова Л.Н., Модорский В.Я.
Анализ динамического напряженно-деформированного состояния акустического трансформатора.....110

Воеводин Вад.В.
Характеристики работы с памятью и эффективность работы программ 114

Волохов В.М., Пивушков А.В., Волохов А.В., Варламов Д.А.
Реализация нескольких независимых ресурсных грид-сайтов на едином физическом пространстве кластера.....119

Гаврилов Н.И., Белокаменская А.А.
Организация потоковых вычислений на GPU
в задаче стереовизуализации томограмм 124

Газизов Р.К., Фатхулисламов И.Р.
GPU и Matlab для решения задачи автоматической
географической привязки спутниковых изображений
с пиксельной точностью на графических процессорах.....133

Галягин К.С., Ипанов А.С., Ошивалов М.А., Селянинов Ю.А.
Оптимизация проточного тракта многофазного эжектора
для перекачки продукции нефтедобывающих скважин 138

Галягин К.С., Ошивалов М.А., Селянинов Ю.А., Вахрамеев Е.И.
Моделирование процесса парофазного осаждения.....145

Гергель А.В.
Параллельные методы многоэкстремальной оптимизации
с адаптивными решающими правилами.....152

Гергель А.В., Гнатюк Д.В.
Сравнение адаптивных параллельных алгоритмов
для многомерной многоэкстремальной оптимизации.....160

Гергель В.П., Ахматнуров Д.В., Калачев А.В., Царев М.С.
Учебно-исследовательская библиотека параллельных
методов для систем с разделенной общей памятью.....166

Гергель В.П., Сиднев А.А.
О применении макромодульной технологии
разработки программ 172

Гергель В.П., Чабан К.А.
Система распределенных вычислений DCS 178

Гиркин М.В., Крыжановский Д.И., Фёдоров М.А.
Моделирование работы резистивной памяти RRAM
на кластерных системах методом Монте-Карло.....183

Горбик В.В., Панюков А.В.
Техника реализации симплекс-метода на кластерных
системах 189

Городецкий Е.С.
Организация многопоточных вычислений на графическом процессоре в задаче расчёта кинематики рычажных механизмов.....198

Дектерева Ю.А., Зимин Д.В., Модорский В.Я.
Использование высокопроизводительных компьютеров для расчета напряженно-деформированного состояния ферменной оправки для непрерывной намотки труб из композиционных материалов 208

Димашова М.П.
Реализация алгоритма сегментации изображений Mean shift на GPU.....214

Добряев Д.Н., Данилова Н.В.
Параллельные алгоритмы глобального поиска с обобщенной характеристикой квадратичного типа 221

Дружков П.Н., Золотых Н.Ю., Мееров И.Б., Половинкин А.Н.
Реализация алгоритма градиентного бустинга деревьев решений.....231

Джосан О.В.
О перспективах и проблемах экзафлопсных вычислений в России 238

Евсеев А.В., Ясинский Ф.Н.
О решении уравнения Навье-Стокса в переменных «Функция тока – Вихрь» с использованием системы с несколькими графическими ускорителями.....245

Ермаков С.А., Замятина Е.Б., Козлов А.А.
Оптимизация распределенного имитационного эксперимента по времени и по надежности.....251

Зимин Д.В., Муленков В.П., Соколкин Ю.В., Модорский В.Я.
Моделирование воздействия транспортных нагрузок на крышу вагона-хоппера 261

Зобачева А.Ю., Кашеварова Г.Г., Фаизов И.Н.
Совершенствование методики расчета соединений элементов деревянных арочных конструкций на стальных цилиндрических нагелях.....267

Золотых Н.Ю., Козинов Е.А., Кустикова В.Д., Мееров И.Б., Половинкин А.Н.
Об одном подходе к решению задачи поиска объектов на изображениях 270

Зубарев В.А., Терещенко В.Н.
Обобщенный параллельно-рекурсивный алгоритм решения набора задач геометрического D-моделирования 278

Исламов Г.Г., Исламов А.Г.
Об одном классе ортогональных проекторов 290

Исупов К.С.
Современный подход к решению «некорректных» математических задач 294

Карзов А.В.
Параллельный алгоритм фрактального сжатия изображений 302

Карпенко С.Н., Дергунов А.В.
Автоматизация повышения производительности MPI-приложений 307

Научное издание

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ (HPC–2010)

Материалы X Международной конференции

В двух томах

Том 1

Редактор и корректор *И.А. Мангасарова*

Подписано в печать 25.10.2010. Формат 60×90/16.
Усл. печ. л. 20,0.
Тираж 150. Заказ № 218/2010.

Издательство
Пермского государственного технического университета.
Адрес: 614990, г. Пермь, Комсомольский пр., 29, к. 113.
Тел. (342) 219-80-33.