МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ВТОРАЯ СИБИРСКАЯ ШКОЛА-СЕМИНАР ПО ПАРАЛЛЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 16–19 декабря 2003 года

Под редакцией доктора физико-математических наук
профессора А.В. Старченко

ИЗДАТЕЛЬСТВО ТОМСКОГО УНИВЕРСИТЕТА
2004

УДК 519.6
ББК 22.19
В 87

Вторая Сибирская школа-семинар по параллельным вычислениям
В 87 /Под ред. проф. А.В.Старченко. – Томск: Изд-во Том. ун-та, 2004. – 108 с.
ISBN 5 – 7511 – 1508 – 2

В сборнике содержатся материалы, представленные во время работы Второй Сибирской школы-семинара по параллельным вычислениям, проходившей 16-19 декабря 2003 года в Томском государственном университете при поддержке Министерства образования РФ.

В докладах участников школы-семинара рассмотрены актуальные проблемы организации параллельных вычислений на многопроцессорных системах, современное состояние и перспективы развития методов параллельных вычислений, методические основы подготовки специалистов в области высокопроизводительных вычислений.

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

УДК 519.6
ББК 22.19

ISBN 5 – 7511 – 1508 – 2 © Томский государственный университет, 2004

РАСПРЕДЕЛЕННАЯ ОБРАБОТКА ДАННЫХ[1]

Вл.В. Воеводин

НИВЦ МГУ

Удивительно быстро развивается вычислительная техника. Еще недавно производительность в один гигафлопс прочно ассоциировалась с суперкомпьютерами, а сегодня это обычная характеристика обычной персоналки. Для того чтобы иметь представление о современных компьютерах с максимальной производительностью, два раза в год формируется список пятисот самых мощных компьютеров мира. В ноябре 2003 года опубликована 22-я редакция списка, определившая лидеров нашего времени. На первом месте находится японский компьютер Earth Simulator производства фирмы NEC с производительностью на тесте Linpack 35,86 Тфлопс. Вторую позицию занимает компьютер ASCI Q производства Hewlett Packard с производительностью на тесте Linpack 13,88 Тфлопс. На третьем месте в новом списке оказался кластер X, состоящий из 2200 процессоров Apple G5 2,0 GHz, соединенных коммуникационной сетью Infiniband. Данный кластер стал третьим компьютером, преодолевшим рубеж 10 Тфлопс, показав на тесте Linpack производительность 10,28 Тфлопс. На данный момент в списке находится 131 система с производительностью на тесте Linpack более 1 Тфлопс (в предыдущей редакции списка было лишь 59 систем). Суммарная производительность 500 компьютеров из списка возросла за полгода с 375 до 528 Тфлопс. Для того чтобы попасть в список, на данный момент требуется производительность более 403,4 Гфлопс (по сравнению с 245,1 Гфлопс в июне 2003 года). Фантастика…

Интересно, что лидер нынешнего списка компьютер Earth Simulator держится на своей позиции уже второй год. В нем все уникально: архитектура, характеристики, возможности. Его создание, безусловно, стало возможным только в результате выполнения проекта действительно общенационального масштаба. Производство компьютера было закончено в феврале 2002 года, после чего он был установлен в Японском центре морских наук и технологий (Japan Marine Science and Technology Center). Сам компьютер и все его технологическое окружение (системы электропитания, кондиционирования, освещения, сейсмозащиты и т.п.) занимают здание размером 50×65×17 метров.

[1] Работа выполняется при поддержке РФФИ, грант № 02-07-90442.

Earth Simulator содержит 640 процессорных узлов, соединенных между собой через высокоскоростной переключатель. В состав узла входит 8 векторных арифметических процессоров, работающих над общей для каждого узла оперативной памятью, коммуникационный процессор и процессор для операций ввода/вывода. Оперативная память каждого узла разделена на 2048 банков и имеет объем 16 Гбайт. Пиковая производительность одного арифметического процессора равна 8 Гфлопс, поэтому пиковая производительность всего компьютера, объединяющего 5120 процессоров, равна 40 Тфлопс. Попутно заметим, что производительность на тесте Linpack составляет 89,6% от теоретического максимума – это также рекордный показатель среди участников списка Top500.

И, наконец, еще одной отличительной особенностью компьютера является то, что его архитектура объединяет все три базовых принципа построения высокопроизводительных систем. Если рассматривать его в целом, то Earth Simulator является массивно-параллельным компьютером с распределенной памятью. Вместе с тем каждый процессорный узел построен на принципах SMP-архитектуры, причем основа каждого процессора – это векторно-конвейерная обработка. Все лучшие качества основных классов современных высокопроизводительных систем были использованы для создания безусловного лидера – суперкомпьютера Earth Simulator.

Но лидер всегда один и уникален, а что же сегодня чаще всего используется на практике? В настоящее время самый динамичный сегмент принадлежит кластерным системам. Общее число кластеров в списке Top500 продолжает быстро увеличиваться и составляет уже 208 систем (149 в июне 2003 года), причем в первую десятку наиболее мощных компьютеров сейчас входит 7 кластеров.

Если попытаться определить коротко, то вычислительный кластер – это совокупность компьютеров, объединенных в рамках некоторой сети для решения одной задачи. В качестве вычислительных узлов обычно используются доступные на рынке однопроцессорные компьютеры, двух- или четырехпроцессорные SMP-серверы. Каждый узел работает под управлением своей копии операционной системы, в качестве которой чаще всего используются стандартные ОС: Linux, NT, Solaris и т.п. Состав и мощность узлов могут меняться даже в рамках одного кластера, давая возможность создавать неоднородные системы. Выбор конкретной коммуникационной среды определяется многими факторами: особенностями класса решаемых задач, доступным финансированием, необходимостью последующего расширения кластера и некоторыми другими. Возможно включение в конфигурацию специализированных компьютеров, например файл-сервера, и, как правило, предоставлена возможность удаленного доступа на кластер через Интернет.

Простор для творчества при проектировании кластеров огромен. Рассматривая крайние точки, кластером можно считать как пару ПК, связанных локальной Ethernet-сетью, так и упоминавшийся выше кластер X, занимающий третье место в списке Top500.

По “кластерному” пути в 1998 году пошли и мы в Научно-исследовательском вычислительном центре МГУ. Именно тогда было принято решение сделать основную ставку на кластерные системы, и время показало, что это решение было абсолютно правильным. Первым мы построили кластер на основе 24 процессоров Intel Pentium-III/500 MHz с использованием сети SCI в качестве вычислительной коммуникационной среды. Через полгода решили провести модернизацию и добавить еще шесть узлов (12 процессоров). И вот здесь, казалось бы, на столь тривиальном действии, нас поджидала неожиданность – узлов на базе использованных процессоров и нужных нам материнских плат в продаже уже не было. Нашли вариант на базе Intel Pentium-III/550 MHz, проблему модернизации кластера введением легкой неоднородности решили, но одновременно появилось четкое представление о серьезной задаче на ближайшее будущее: процессоры быстро совершенствуются, значит, неоднородность кластеров НИВЦ МГУ будет столь же быстро расти. Можно ли будет в таких условиях использовать весь потенциал вычислительных ресурсов суперкомпьютерного центра или придется для каждой задачи выделять лишь некоторое число процессоров какого-либо одного кластера?

Задача оказалась исключительно актуальной. Ниже показана современная структура вычислительного комплекса НИВЦ МГУ, причем в ближайшее время ряды кластеров пополнятся новыми установками:

• Кластер SCI, 36 CPUs, 18 узлов: 2 × Intel Pentium III/500 MHz, 1GB, HDD 3,2 GB, Fast Ethernet + SCI, 18 Gflops,

• Кластер SKY, 40 CPUs, 20 узлов: 2 × Intel Pentium III/850 MHz, 1GB, 2 × HDD 15 GB, Fast Ethernet, 34 Gflops,

• Кластер AQUA, 40 CPUs, 24 узла: 2 × Intel Pentium III/1 GHz, 1GB, HDD 20 GB, Fast Ethernet, 48 Gflops,

• Кластер LEO, 32 CPUs, 16 узлов: 2 × Intel Xeon 2.6 GHz, 2GB, 2 × HDD 40 GB, Fast Ethernet + SCI (D335), 166 Gflops.

Как обеспечить преемственность вычислительной техники для тех пользователей, которым необходимы максимально возможные ресурсы? Как сделать так, чтобы появление новых систем не приводило бы к забвению старых, а просто давало пользователю приращение мощности за счет добавления новых процессоров к уже имеющемуся парку компьютеров?

Заметим, что похожая картина в настоящее время складывается и в сети Интернет: скоростные характеристики сети растут, число процессоров в сети огромно – чем не параллельный компьютер? Причем самый мощный в мире…. Но вот как его использовать для решения одной задачи? В

этом и состоит проблема, которую мы решаем с помощью разработанной в НИВЦ МГУ системы X-Com.

Основные требования к проектируемой системе были сформулированы следующим образом:

• ориентация на вычислительные задачи;

• работа через Интернет, возможность использования всех доступных в сети вычислительных ресурсов различной мощности;

• минимум дополнительных действий и системного вмешательства на используемых ресурсах;

• масштабируемость системы, устойчивость к неоднородности и изменению конфигурации вычислительной среды;

• простота адаптации прикладных программ.

К настоящему времени система прошла апробацию в ходе нескольких масштабных экспериментов с использованием широкого спектра ресурсов: от простых домашних компьютеров до мощных параллельных вычислительных систем, от использования сети до работы на выделенном гигабитном полигоне Московского университета. В каждом случае на подготовку и разворачивание распределенного эксперимента требовалось не более 1 часа.

Совместно с группой специалистов из Центра «Биоинженерия» РАН, в распределенном режиме решалась задача определения скрытой периодичности в генетических последовательностях. Основную проблему в данном случае представляли высокая вычислительная сложность алгоритмов и огромный объем входных данных. По самым скромным подсчетам для обработки материала, имеющего реальное научное значение, на одном процессоре потребовалось бы обработать несколько гигабайт входной информации и многие месяцы, а то и годы, непрерывных вычислений. На вход программы подавались описания структуры генетических последовательностей, закодированные стандартным набором символов a-t-g-c, а на выходе появлялась информация о найденных во входных последовательностях повторах тех или иных фрагментов.

Все входные данные были взяты из специализированного банка данных – центрального хранилища актуальной информации о генетической структуре различных организмов (http://ncbi.nih.gov). Во всех расчетах были задействованы географически удаленные вычислительные ресурсы, объединенные с помощью системы X-Com.

Для решения исходной задачи в среде X-Com все входные последовательности разбивались на порции, содержащие от нескольких десятков до нескольких сотен тысяч символов. Часть порций нарезалась с “перехлестом”. В этом случае конец предыдущей порции совпадал с началом последующей, что было необходимо для нахождения повторов в самой зоне разрезания. Количеством символов в порции можно было регулировать объем

вычислений, необходимый для ее обработки в разумное время. В проведенных экспериментах время обработки одной порции менялось в пределах нескольких десятков минут. Для сокращения сетевого трафика выполнялась online-компрессия всех передаваемых данных. Общий объем написанного для интеграции с системой X-Com дополнительного кода составил менее 200 строк.

Проводя серию экспериментов, одновременно с решением основной прикладной задачи нами преследовались две дополнительные цели: проверить потенциальные возможности X-Com и определить особенности ее работы при пиковых нагрузках. Остановимся подробнее на описании двух наиболее интересных экспериментов.

Цель первого эксперимента – географический размах: 14 удаленных вычислительных систем, принадлежащих 10 организациям, расположенным в 8 городах и 6 различных часовых поясах. В разные моменты времени к эксперименту подключалось 407 процессоров, причем максимальное число одновременно работающих процессоров составило 385, а в среднем по эксперименту эта величина оказалась равной 296. Общее время эксперимента составило около 64 часов. Один средний компьютер справился бы с подобной задачей лишь за два года непрерывной работы. Взаимодействие между компьютерами шло по обычным каналам связи, и суммарно было передано 9 Гбайт данных. Максимальная загрузка исходящего канала данных от центрального сервера составила около 7 Мбит/с. Никаких специальных действий, направленных на обеспечение гарантированной полосы пропускания или увеличение стабильности связи, не предпринималось.

Таблица 1

Вычислительные ресурсы “географического” эксперимента

Организация и кластер	Город	Платформа	Число	
			процессоров	выполненных заданий
ЛИТ ОИЯИ. DN1	Дубна	Linux / Intel	16	3824
ЛИТ ОИЯИ. DN2	Дубна	Linux / Intel	16	3712
ИММ УрО РАН	Екатеринбург	Linux / Intel	16	2108
НИИЯФ МГУ	Москва	Linux / Intel	44	4270
НИВЦ МГУ. AQUA	Москва	Linux / Intel	40	10464
НИВЦ МГУ. SCI	Москва	Linux / Intel	32	3366
НИВЦ МГУ. SKY	Москва	Linux / Intel	40	7404
Офисные компьютеры	Нью-Йорк	Windows / Intel	5	834
ИПС РАН	Переславль-Залесский	Linux / Intel	32	11904
Физфак МГУ	Москва	Linux / Intel	10	488
ИМПБ РАН	Пущино	Linux / Intel	40	3656
ГТУ	Тверь	Linux / Intel	16	45
УГАТУ	Уфа	Linux / Intel	64	490
ВМиК МГУ	Москва	Linux / Intel	36	3067

В табл. 1 показаны вычислительные системы, работавшие в этом эксперименте. Число выполненных каждой системой заданий определялось как их мощностью, так и длительностью участия в эксперименте. Так, кластеры Дубны и Переславля-Залесского работали все время, а системы из Твери и Уфы подключались лишь на несколько часов.

Несмотря на значительную географическую удаленность использованных вычислительных систем, эффективность работы системы в целом превысила 95% (отношение суммарного времени полезных вычислений на узлах к общему времени использования данных узлов в эксперименте).

Задача второго эксперимента – работа в условиях максимальной неоднородности. Организаторы эксперимента: НИВЦ МГУ и Межведомственный суперкомпьютерный центр (МСЦ). В расчете участвовали четыре типа вычислительных ресурсов: отдельные компьютеры, учебный класс, традиционные вычислительные кластеры и суперкомпьютер МВС-1000М. Часть компьютеров работала под управлением Windows, а часть – под Linux (как правило, Red Hat). На одной части компьютеров использовались процессоры Intel, а на другой – Alpha. Одновременно в эксперименте применялись три различных режима использования ресурсов (табл. 2):

• МОНО – монопольное использование, при котором вычислительная система полностью отдавалась под проведение данного эксперимента;

• ОЧЕР – работа через стандартные системы управления очередями заданий. Данный режим не требует менять политику администрирования и использования вычислительной системы при ее подключении и после ее выхода из эксперимента;

• ЗАН – режим “по занятости” позволяет использовать свободные ресурсы вычислительной системы. С помощью специальных алгоритмов система X-Com определяет, что узел в данный момент не используется, и запускает на нем прикладную задачу. Как только любая другая программа на узле проявляет активность, X-Com приостанавливает свою работу.

Эксперимент продолжался 14,5 часов. Среднее число одновременно работающих процессоров составило 125 (максимум зафиксирован на значении 150). Всего к эксперименту подключалось 437 различных процессоров. Казалось бы слишком много, учитывая среднее число 125. Но объясняется это очень просто. Суперкомпьютер МВС-1000М объединяет 768 процессоров, и система управления заданиями при распределении программы на реальные процессоры не обязана учитывать, куда были распределены предыдущие варианты этой же программы.

Всего за время этого эксперимента было передано 26,8 Гбайт данных. Время обработки одной порции данных менялось от 1 секунды до 10 минут. Несмотря на значительную коммуникационную нагрузку и специально созданную неоднородность, эффективность работы всей системы составила 92,9%. В табл. 2 приводятся характеристики вычислительных си-

стем, режимов их использования и агрегированные данные хода вычислений.

Таблица 2

Вычислительные ресурсы “неоднородного” эксперимента

Кластер	Платформа	Режим	Выдано заданий	Обработано заданий	Оборванные сессии	
					число	%
МСЦ. KAPPA	Linux / Intel	МОНО	2485	2476	9	0,36
МСЦ. МВС-1000М	Linux / Alpha	ОЧЕР	3203	2796	407	12,70
НИВЦ. AQUA	Linux / Intel	МОНО	9689	9645	44	0,45
НИВЦ. SCI	Linux / Intel	ЗАН	501	467	34	6,78
НИВЦ. SKY	Linux / Intel	ЗАН	5184	5184	49	0,93
НИВЦ. Уч.классы	Windows / Intel	МОНО	1642	1599	43	2,61

Естественно, что наиболее эффективно используются системы в монопольном режиме. Небольшие накладные расходы, возникшие при использовании режима “по занятости”, во многом объясняются малым изменением вычислительной нагрузки на соответствующих кластерах во время данного эксперимента. Самый большой процент оборванных сессий оказался при использовании стандартных систем управления очередями. Объясняется это тем, что при постановке в очередь система заказывает для программы некоторое время. После того как программу распределяют на реальные узлы, она начинает штатную обработку порций данных и продолжает это до того момента, пока не исчерпает свой лимит по времени, после чего всегда происходит разрыв сессии между узлом и сервером.

Проведенные эксперименты убедительно показали, насколько эффективно можно решать большие задачи с использованием существующих распределенных вычислительных ресурсов. Конечно же, X-Com не является, да и не может быть универсальным решением, одинаково хорошо соответствующим любой задаче и любой вычислительной среде. Но то, что она является одним из эффективных и реально применимых на практике подходов, это точно. Что использовать в каждом случае, определяется прежде всего самим пользователем и стоящими перед ним задачами. Сейчас важно осознать, что в подобных неоднородных распределенных вычислительных средах заложен колоссальный потенциал, и очень важно не опоздать с его освоением.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 609 с.
2. http://www.parallel.ru/ – Информационно-аналитический центр по параллельным вычислениям в сети Интернет.

ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ: ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ЭВОЛЮЦИОННЫЕ СТРАТЕГИИ[1]

Ю.Н. Сидоренко, С.В. Тимченко

Томский государственный университет, Томский государственный университет систем управления и радиоэлектроники

В последние годы при решении различных задач оптимизации и поиска стали широко применяться различные адаптивные процедуры, среди которых особую популярность завоевали эволюционные и в том числе генетические алгоритмы (ГА) и эволюционные стратегии (ЭС). В той или иной мере они основаны на эволюционных процессах в биологических популяциях, развивающихся поколение за поколением, подчиняясь законам естественного отбора и принципу «выживает сильнейший» (точнее, наиболее приспособленный).

Чрезвычайно важным свойством как ГА, так и ЭС является их естественный внутренний параллелизм. При этом по сравнению с градиентными методами различие во времени расчета целевой функции при различных значениях ее параметров не оказывает влияния на эффективность распараллеливания (эти времена могут отличаться на порядки – например, если можно определить, что точка в пространстве поиска не отвечает наложенным ограничениям, не вычисляя целевую функцию или вычисляя ее только частично).

1. Описание последовательного ГА

Генетические алгоритмы носят итерационный характер и имеют дело с обработкой популяций индивидуумов $P(t) = \left\{x_1^t, x_2^t, ..., x_n^t\right\}$ для итерации t (поколение t). Каждый индивидуум представляет собой потенциальное решение задачи (испытание) и реализуется в некоторой, возможно достаточно сложной, структуре данных S. В этой работе в качестве S рассматриваются строки, составными элементами которых являются вещественные числа (вещественные генетические алгоритмы).

Каждое решение x_i^t оценивается, и определяется мера его «пригодности». Затем формируется новая популяция (итерация или поколение $t+1$). На первом шаге этого формирования – этапе селекции – происходит отбор индивидуумов, обладающих лучшей пригодностью. На следующем шаге

[1] Работа выполнена при поддержке гранта РФФИ 02-01-01022.

некоторые из отобранных таким образом индивидуумов подвергаются преобразованиям с помощью «генетических операторов»: мутации и скрещивания. Оператор мутации создает нового индивидуума путем относительно малого изменения в одном индивидууме, а оператор скрещивания осуществляет более сильные трансформации и создает нового индивидуума путем комбинирования частей из нескольких (двух или больше) индивидуумов. После ряда итерационных шагов алгоритм сходится к лучшему из возможных решений. Остановимся теперь более подробно на трех «генетических операторах» – селекции, скрещивании и мутации.

Селекция. Целью селекции является осуществление выборки индивидуумов в текущей популяции (т.е. из некоторого набора) пропорционально их пригодности. Обычно используют четыре различных механизма селекции – “колесо рулетки”, остаточную стохастическую выборку, равномерную стохастическую выборку и турнирную селекцию. Первые три алгоритма являются вариантами пропорциональной селекции, а последний – непропорциональной.

Наиболее универсальной считается так называемая турнирная селекция, не требующая предварительного ранжирования функции пригодности. При этом последовательно берутся два соседних элемента текущей популяции (первый и второй, третий и четвертый и т.д.) и лучший из них помещается в промежуточную популяцию P'. После первого прохода (пока сформирована только половина промежуточной популяции) исходная популяция случайным образом перемешивается и описанный процесс повторяется еще раз. Здесь лучшие или худшие индивидуумы рассматриваются в смысле их упорядочивания согласно соответствующим значениям целевой функции.

Скрещивание. Наиболее простым является одноточечное скрещивание – каждая выбранная пара строк скрещивается следующим образом: случайно выбирается положение точки сечения (целое число k в промежутке от 1 и $l-1$, где l – длина строки). Затем путем обмена всеми элементами между позициями $k+1$ и l включительно рождаются две новые строки. Например, пусть первая особь – $A=(x_1,x_2,x_3,x_4,x_5)$, а вторая соответственно $B=(y_1,y_2,y_3,y_4,y_5)$ и пусть случайно выбранная точка сечения будет после третьего гена. Тогда в результате скрещивания получим две особи-потомка $A'=(x_1,x_2,x_3,y_4,y_5)$ и $B'=(y_1,y_2,y_3,x_4,x_5)$. После этого потомки замещают родительские особи в промежуточной популяции P'. Схематично этот вариант показан на рис. 1.

Одноточечное скрещивание легко обобщается на *n*-точечное с *n* точками сечения. Предельным случаем является равномерное скрещивание, при котором каждый ген первого из родителей случайным образом переда-

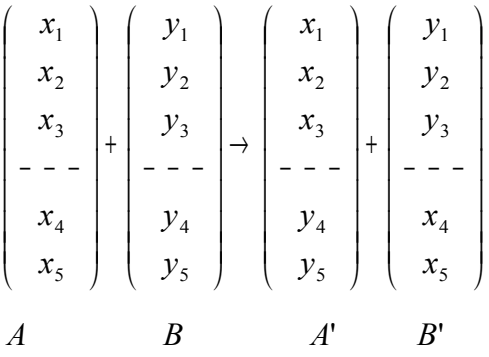


Рис. 1. Скрещивание

ется любому из потомков, при этом другой потомок соответственно получает ген от другого родителя.

Для вещественных ГА наиболее интересно арифметическое скрещивание [2], так как в этом случае сами гены подвергаются модификации. Рассмотрим этот механизм в символьном виде. Пусть у нас есть две родительские особи:

$A = (y_1,...,y_n)$ и $A' = (y'_1,...,y'_n)$.

В результате арифметического скрещивания они заменяются потомками вида

$$A = (\alpha y_1 + (1-\alpha)y'_1,...,\alpha y_n + (1-\alpha)y'_n)$$

и

$$A' = (\alpha y'_1 + (1-\alpha)y_1,...,\alpha y'_n + (1-\alpha)y_n)$$

при некотором случайном значении α из диапазона [–0.5,1.5].

Мутация. Для вещественного ГА использовалась неоднородная мутация [2]. Если ген y_i подвергается мутации, то его новое измененное значение y'_i выбирается внутри интервала $[Min_i, Max_i]$ следующим образом:

$$y_i' = y_i + s(M_i - y_i)\left(1 - \frac{t}{t_{\max}}\right)^b,$$

где s – случайное число из интервала [0,1], M_i случайным образом выбирается из множества $\{Min_i, Max_i\}$, где Min_i и Max_i – нижняя и верхняя границы возможного изменения значения переменной y_i. Такая адаптивная мутация позволяет соблюдать в процессе реализации ГА (эволюции) необходимый баланс между двумя разномасштабными изменениями (мутациями) генов, так как на первоначальных шагах алгоритма в основном преобладали крупномасштабные изменения (обеспечивающие широкую область поиска), в то время как на заключительном этапе (за счет уменьшения масштаба мутаций) происходило уточнение решения.

Полезно рассмотреть выполнение генетического алгоритма как двухстадийный процесс (рис. 2). Начинается он с текущей популяции, к которой применяется оператор выбора, чтобы создать промежуточную популя-

цию. После этого к промежуточной популяции применяются операторы рекомбинации и мутации для того, чтобы создать следующую популяцию. Процесс продвижения от текущей популяции до следующей популяции составляет одно поколение в выполнении генетического алгоритма.

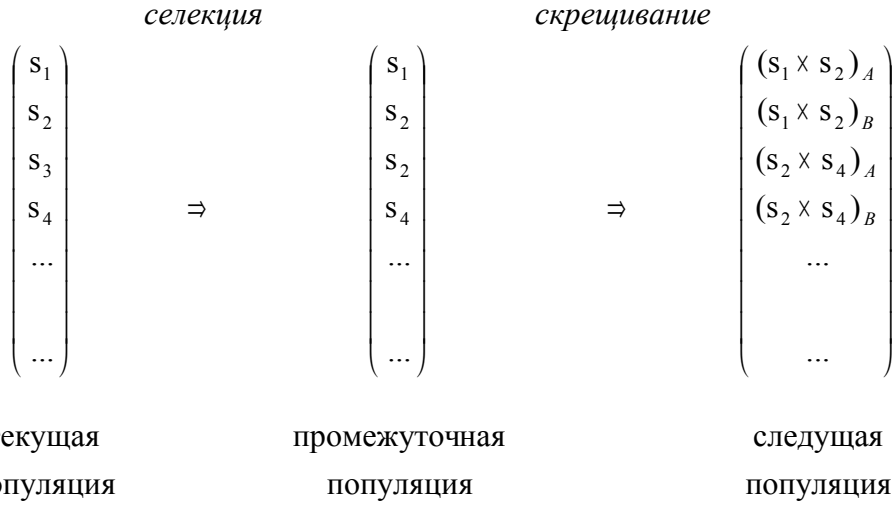


Рис. 2. Генетический алгоритм

2. Параллельные генетические алгоритмы

Как было отмечено выше, у генетических алгоритмов есть чрезвычайно важное свойство. Они обладают естественным внутренним параллелизмом. Причем различие времени расчета целевой функции для разных процессов несущественно. Интересно отметить, что основоположник генетических алгоритмов Холланд [1] задолго до своих первых работ, посвященных ГА, в конце 50-х годов предложил создание мультипроцессора для запуска недетерминированного числа конкурирующих процессов, адаптирующихся во время выполнения (для моделирования эволюции естественных популяций).

Основная идея большинства параллельных алгоритмов заключается в том, чтобы разбить задачу на сравнительно небольшие подзадачи и решать их совместно, используя многопроцессорную вычислительную систему. Стратегия «разделяй и властвуй» может быть реализована большим количеством способов. Одни из них лучше подходят для систем с массовым параллелизмом, другие – для небольших параллельных систем.

2.1. Глобальные ПГА (технология master-slave). В этом случае существует только одна популяция на master-процессоре, как и в последовательном ГА. При этом остальные процессоры (slave-процессоры или workers-процессоры) используются только для расчета целевых функций (рис. 3). Метод очень просто реализуется (так как фактически используется

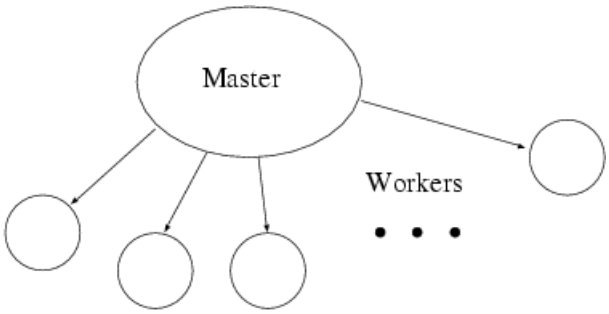


Рис.3. Глобальный ПГА

последовательный ГА) и хорош в тех случаях, когда вычислительная часть задачи доминирует над коммуникационной. При построении глобальных ПГА не делается никаких предположений об архитектуре параллельного компьютера. Число индивидуумов, приходящихся на один процессор, либо константа, либо может динамически меняться с целью обеспечения равномерности загрузки (например, в многопользовательской среде).

Как правило, используются синхронные глобальные ПГА, хотя существуют реализации и асинхронных глобальных ПГА [3]. На компьютерах с MIMD-архитектурой алгоритм, как правило, состоит в размещении популяции на одном процессоре, рассылке индивидуальностей на workers-процессоры, получении на главном процессоре значений целевой функции и применении генетических операторов для следующего шага.

Ниже приводится пример псевдокода асинхронного глобального ПГА.

```
t = 0
initpopulation P(t) /* random or random + initial solution(s) */
Scatter P(t)        /* for evaluation */
while not done do
    P' := ∅
    while size(P') < size(P) do
        Recv_from_j (P_i, f(P_i)) /* from j - th node after evaluation */
        P' = P' ⊕ P_i
        P'' := selectparents P(t)    /* tournament selection */
        recombine P''                /* arithmetical crossover */
        mutate P''  /* nonuniform + distance - dependent mutation */
        Send_to_j(P'') /* to j - th node for evaluation */
    enddo
    P(t + 1) := P'(t) + best(P(t)) /* elitism */
    t := t + 1
enddo
```

2.2. Более сложная модель – крупнозернистый ПГА [4] (другие названия - распределенный ГА или островная модель ГА), в рамках которой

на каждом процессоре располагается изолированная сабпопуляция, обменивающаяся особями с другими сабпопуляциями за счет миграции, регулируемой рядом параметров (рис. 4). Каждая сабпопуляция невелика по размеру, поэтому скорость сходимости весьма высока, хотя качество решения оставляет желать лучшего. Существует критическая скорость миграции, позволяющая получить результат, идентичный последовательному ГА. Один из подходов состоит в том, что на каждом шаге отсылается соседям лучшая особь. Однако в этом случае существует опасность преждевременной сходимости к локальному экстремуму. Другой подход – замена худшего на лучшего от соседей каждые пять (например) итераций. В целом можно отметить, что

• КЗПГА - это достаточно несложное обобщение последовательного ГА, так как фактически имеем несколько последовательных ГА на процессорах, иногда обменивающихся отдельными особями.

• Легко реализуется на кластерах.

• Последовательный ГА легко адаптируется к КЗПГА.

Основными параметрами КЗПГА являются:

• топология связей между сабпопуляциями,

• интенсивность миграции (сколько особей посылается за один раз),

• интервал миграции (количество поколений, после которых происходит миграция).

2.3. *Мелкозернистый ПГА, или клеточный ГА* (рис.5). В этом случае используется большое число очень маленьких сабпопуляций (в пределе - одна особь на процесс), которые интенсивно обмениваются индивидуумами. Эта модель наиболее хороша для MPP-архитектур, но может применяться на любых мультипроцессорах.

2.4. *Гибридные схемы*, объединяющие сразу две из предложенных выше схем (рис.6).

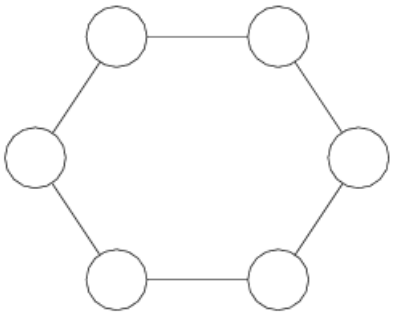


Рис.4. Крупнозернистый ПГА

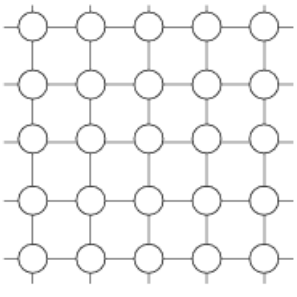
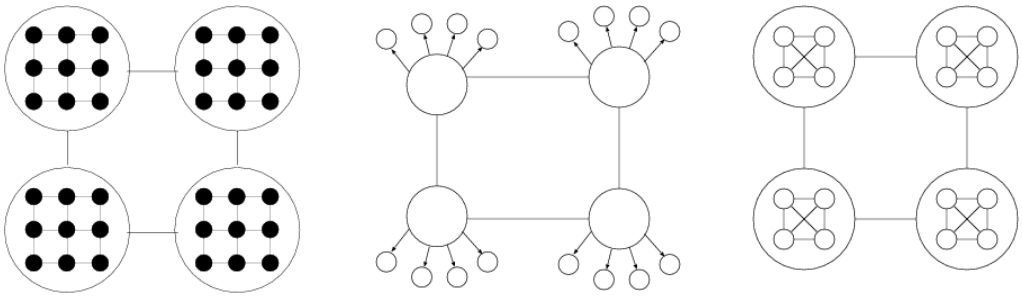


Рис.5. Мелкозернистый ПГА



а) б) в)

Рис. 6. Гибридные схемы:

а) крупнозернистый ПГА+ мелкозернистый ПГА
б) глобальный ПГА + крупнозернистый ПГА
в) крупнозернистый ПГА + крупнозернистый ПГА

Литература

1. Holland J.H. Adaptation in natural and artificial systems. – Ann Arbor: The University of Michigan Press, 1975. – 412 p.
2. Michalewicz Z. Genetic algorithms + data structures = evolution programs. – New York: Springer-Verlag, 1992.
3. Peigin S., Periaux J., Timchenko S. Asynchrone Parallel Genetic Algorithm for Heat Flux Optimization Problem // Parallel Computational Fluid Dynamics. Development and Applications of Parallel Technology. – Amsterdam: Elsevier Science B.V., 1999. – P. 377–384.
4. Grefenstette, J. J. Parallel adaptive algorithms for function optimization (Tech. Rep. No. CS-81-19). – Nashville, TN: Vanderbilt University, Computer Science Department, 1981.

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В ЗАДАЧАХ ОХРАНЫ ОКРУЖАЮЩЕЙ СРЕДЫ[1]

А.В. Старченко

Томский государственный университет

В настоящее время у нас в стране и за рубежом наряду с приборным контролем качества воздуха широко используются методы математического моделирования при анализе характера распространения газодисперсных загрязнителей атмосферы, поступающих вместе с выбросами промышленных предприятий и автотранспорта. Методы математического моделирования позволяют быстро и с малыми затратами выполнять прогноз детальной картины распределения примесей в приземном слое атмосферы. Наряду с решением задач контроля и краткосрочного прогноза математическое моделирование может также использоваться для проведения сценарного анализа по выявлению наиболее неблагоприятных метеорологических условий, а также по установлению факторов (и объектов) повышенной экологической опасности в конкретном районе.

При численном моделировании рассеяния и переноса примеси используются в основном два подхода: эйлерова модель атмосферной диффузии и лагранжева дисперсионная стохастическая модель. В рамках первого подхода на эйлеровой конечно-разностной сетке для концентрации примеси решается трехмерное нестационарное уравнение «адвекции-диффузии» с источниковыми членами, моделирующими поступление, осаждение и химические превращения компонентов. Этот подход требует высокого сеточного разрешения в областях высоких градиентов концентраций, что в случае учета химических реакций между несколькими компонентами делает задачу чрезвычайно трудоемкой в вычислительном плане.

Второй подход представляет пространственный перенос примеси в атмосфере по траекториям движения одиночных частиц-трассеров, поступающих из источников загрязнения в воздушный бассейн, где под действием ветра и турбулентности происходит их рассеяние. Значения концентраций веществ оцениваются на эйлеровой разностной сетке по времени прохода частицей каждой отдельной ячейки. Поскольку в основе этого подхода лежит стохастическая модель, качественные результаты возможны только при запуске большого числа (нескольких тысяч) частиц из каждого источника, что требует привлечения суперкомпьютерной техники.

[1] Работа выполнена при поддержке РФФИ, грант № 04-07-90219.

При использовании эйлерова подхода решается система прогностических уравнений «адвекции-диффузии» [1]:

$$\frac{\partial c_j}{\partial t} + \frac{\partial Uc_j}{\partial x} + \frac{\partial Vc_j}{\partial y} + \frac{\partial W_j c_j}{\partial z} = \\ = \frac{\partial}{\partial x}\left(K_Z^h \frac{\partial c_j}{\partial x} \right) + \frac{\partial}{\partial y}\left(K_Z^h \frac{\partial c_j}{\partial y} \right) + \frac{\partial}{\partial z}\left(K_Z^h \frac{\partial c_j}{\partial z} \right) + Q_j, \; j = 1,...,s. \tag{1}$$

Здесь c_j – концентрация j-й компоненты примеси; Q_j – источниковый член, моделирующий поступление примеси от источников выбросов, изменение концентрации вследствие химических реакций и осаждение составляющей примеси; U,V – горизонтальные компоненты скорости ветра; W_j – скорость вертикального движения j-й компоненты примеси, складывающаяся из вертикальной скорости ветра и скорости подъема или осаждения примеси; s – количество компонент примеси; $K_Z^h = F_h\sqrt{El}$ – коэффициент турбулентной диффузии; E,l – энергия и масштаб турбулентных пульсаций; F_h – функция локальных значений турбулентных корреляций [1]. В качестве граничных условий для уравнения (1) используются простые градиентные соотношения на боковых, верхней и нижней границах области исследования.

Пространственные нестационарные уравнения (1), моделирующие перенос компонентов примеси внутри параллелепипеда с многочисленными линейными, точечными и площадными источниками на подстилающей поверхности и высотными точечными источниками, решались численно. Для этой области исследования строилась конечно-разностная сетка с постоянными размерами ячеек по горизонтальным направлениям и переменными вертикальными, уменьшающимися при приближении к поверхности. Аппроксимация дифференциальных операторов в (1) осуществлена со вторым порядком точности по координатам и первым – по времени с исполь-

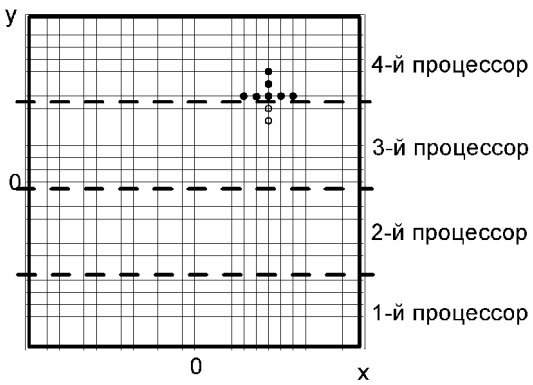


Рис.1. Расчетная сетка в xy-плоскости с указанием распределения подобластей по четырем процессорам. На приграничной сеточной линии приведен разностный шаблон. Открытые кружки указывают на сеточные значения, которые необходимо получить с соседнего процессора

зованием явных разностных схем для всех членов уравнения, за исключением вертикальной диффузии. Такой способ дискретизации дифференциальной задачи позволяет решить проблемы, возникающие из-за нелинейности уравнения (1) и существенно ускорить получение результата вследствие применения экономичного метода прогонки при приемлемом ограничении на величину шага по времени. Адвективные члены прогностических уравнений (1) аппроксимируются с использованием монотонизированной противопотоковой схемы Ван Лира, не допускающей появления «нефизичных» значений концентраций.

В рассматриваемой эйлеровой модели турбулентной диффузии основная вычислительная нагрузка связана с решением 3D прогностических уравнений (1). Требования контроля качества воздуха указывают на необходимость получения детальной картины распределения загрязнений в приземном слое воздуха, поэтому расчетная сетка должна иметь высокую плотность узлов в области исследования. Кроме того, для учета протекания химических реакций, приводящих к появлению вторичных загрязнителей воздушного бассейна (озон), необходимо рассмотрение переноса и рассеяния большого количества компонентов примеси. Такие условия проведения численного анализа заставляют привлекать высокопроизводительную вычислительную технику, в частности имеющиеся в распоряжении многопроцессорные кластерные системы Томского государственного университета (http://cluster.tsu.ru) и Института оптики атмосферы СО РАН (http://cluster.iao.ru), на которых установлен интерфейс передачи сообщений MPI (Message Passing Interface). MPI включает библиотеку функций для программирования обмена информации между процессами параллельной программы и загрузчик соответствующих исполняемых файлов на многопроцессорной вычислительной системе.

Распараллеливание численного метода решения уравнений (1) производилось с использованием геометрического принципа – декомпозиции по данным. Вся область исследования разделялась на одинаковые подобласти. В данной работе параллелепипед области исследования

$$-L_x/2 \le x \le L_x/2;$$
$$-L_y/2 \le y \le L_y/2;$$
$$\delta(x,y) \le z \le L_z$$

($\delta(x,y)$ - функция, описывающая рельеф подстилающей поверхности) разрезался сечениями $y =$ const с распределением данных каждой подобласти соответствующему процессорному элементу. Все сеточные значения концентрации $(c_j)^{n+1}_{k,l,m}$ равномерно распределялись по вычислительным узлам многопроцессорной системы с распределенной памятью. Внутри каждой подобласти сеточные уравнения, полученные в результате принятого подхода дискретизации уравнений (1), решались одновременно методом прогонки. Однако из-за выбранного разностного шаблона при расчете значе-

ний концентраций вдоль двух приграничных сеточных линий (вблизи верхней или нижней границы подобласти; рис.1) необходимы сеточные значения концентрации из соседней подобласти. Поэтому для корректной работы параллельной программы следует организовать межпроцессорные обмены приграничных сеточных значений. Это было выполнено с использованием функций библиотеки передачи сообщений MPI_SendRecV. Кроме того, при подготовке проведения параллельных расчетов применялись следующие функции библиотеки Message Passing Interface: MPI_Bcast, MPI_Scatter (для рассылки исходных данных и информации об источниках выбросов по другим процессорам), MPI_Gather (для «сборки» массивов всех значений концентрации на одном процессоре).

Предложенный метод решения задачи о распределении концентраций основных загрязнителей над городом с использованием высокопроизводительной вычислительной системы был применен для условий г. Томска. Учитывалось 119 линейных, 12 площадных и 338 точечных источников выбросов. Расчеты выполнялись на сетке 100x100x50 с шагом по времени 15 секунд для вычисления распределений концентраций основных загрязнителей городского воздуха.

На рис. 2 и 3 представлены характеристики эффективности и ускорения рассмотренного выше параллельного алгоритма решения задачи переноса примеси, полученные при ее запуске на кластерной установке

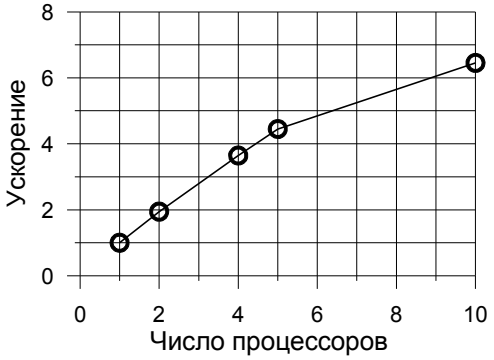


Рис. 2. Ускорение параллельных вычислений

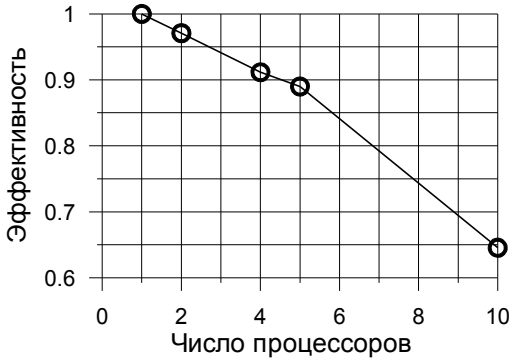


Рис. 3. Эффективность параллельной программы

Томского государственного университета (9 двухпроцессорных вычислительных узлов с процессорами Pentium III 650MHz, RAM 256Mb соединены сетью Fast Ethernet 100Mbit). Ускорение рассчитывается как отношение времени выполнения задачи на одном процессоре к времени решения задачи на p процессорах при неизменных других параметрах. Из рис. 2 видно, что при выполнении параллельной программы на 10 процессорах кластера ТГУ при указанных выше условиях удается получить ускорение вычислений более чем в 6 раз. В то же время величина эффективности параллельной программы (отношение ускорения к числу использованных в расчетах процессоров) с ростом p уменьшается. Это объясняется тем, что при фик-

сированном количестве узлов конечно-разностной сетки отношение объема вычислительной работы к числу совершаемых обменов между процессорами снижается. Тем не менее на 10 процессорах эффективность параллельного алгоритма остается достаточно высокой и для выбранных размеров сетки достигает 65%.

Сравнительные расчеты, выполненные на многопроцессорной вычислительной системе Института оптики атмосферы СО РАН (10 двухпроцессорных элементов Pentium III 1GHz, RAM 1Gb, сеть Gigabit Ethernet 1Gbit), показали, что временные затраты с использованием десяти двухпроцессорных элементов при получении прогноза на двое суток по распределению четырех основных загрязнителей городского воздуха занимают чуть более 3 часов реального времени.

Лагранжева дисперсионная стохастическая модель позволяет аккуратно рассчитывать начальный этап подъема нагретой примеси, осаждение и перенос промышленных выбросов, содержащих взвешенные полидисперсные частицы. В рамках этого подхода распространение загрязнителей оценивается по траекториям большого числа одиночных частиц-трассеров, поступающих из источников в воздушный бассейн, где под влиянием ветра и турбулентных флуктуаций происходит их рассеяние. Траектории представительных частиц рассчитываются из следующих уравнений:

$$\vec{r}_m(t+\Delta t) = \vec{r}_m(t) + \left[\vec{W}_m(\vec{r}_m,t) + \vec{w}(\vec{r}_m,t)\right]\Delta t\,, \qquad (2)$$

где $\vec{r}_m$ – радиус-вектор положения m-й частицы; $\vec{W}_m$ – осредненная скорость примеси, рассчитываемая на основе значений скорости ветра и скорости подъема (осаждения) примеси; $\vec{w}$ – турбулентные пульсации скорости ветра; Δt – шаг интегрирования по времени. Значения компонент пульсационной скорости ветра определялись из конечно-разностного аналога уравнения Ланжевена [2]:

$$\vec{w}(\vec{r}_m(t+\Delta t), t+\Delta t) = \vec{w}(\vec{r}(t),t) + \vec{a}\Delta t + (C_0\varepsilon)^{0,5}\vec{R}\Delta t\,. \qquad (3)$$

Ускорение $\vec{a}$ зависит от стратификации атмосферы, величины скорости диссипации энергии турбулентности ε, нормальных напряжений Рейнольдса $\sigma_1, \sigma_2, \sigma_3$ и определяется выражениями, предложенными в [2]. Последнее слагаемое в (3) представляет турбулентную диффузию, $\vec{R}$ – вектор, компоненты которого суть случайные числа, описываемые нормальным гауссовым распределением с нулевым средним. В случае, когда частица достигнет верхней или нижней границы расчетной области, используются условия идеального отражения. Если частица-трассер пересекает боковую границу, то расчет ее движения прекращается.

Концентрация примеси рассчитывается по траекториям частиц на эйлеровой сетке из следующего соотношения:

$$c_{i,j,k} = \frac{Q}{N * Vol_{i,j,k}} \sum_{m=1}^{N_{i,j,k}} \Delta t_m\,, \qquad (4)$$

где (i, j, k) – номер ячейки сетки; $Vol_{i,j,k}$ – объем ячейки; Q – интенсивность источника (кг/с); N – число частиц, запущенных из источника; $N_{i,j,k}$ – число частиц, прошедших через ячейку; Δt_m – время прохода частицы через ячейку. Поскольку траектории движения каждой отдельной частицы рассчитываются независимо, этот подход идеально распараллеливается. Все частицы-трассеры, поступающие из одного источника, распределялись по процессорам, и расчет их перемещения велся параллельно. После окончания расчетов данные со всех вычислительных узлов собирались на узле-сервере с использованием функции библиотеки Message Passing Interface MPI_Reduce.

Литература

1. Старченко А.В. Моделирование переноса примеси в однородном атмосферном пограничном слое // Материалы Международной конференции ENVIROMIS2000. – Томск, 2000. – С. 77–84.
2. Атмосферная турбулентность и моделирование распространения примесей /Под ред. Ф.Т.М.Ньистадта и Х.Ван Допа. – Л.: Гидрометеоиздат, 1985. – 351 с.

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ СПЛАЙНОВ

В.Н. Берцун

Томский государственный университет

Сплайны дают возможность гладкого восстановления сеточной функции и ее производных, упрощают визуализацию расчетов, используются для конструирования экономичных разностных схем решения многомерных краевых задач [1, 2].

Пусть на отрезке $[a,b]$ вещественной оси x задана сетка

$\omega : a = x_0 < x_1 < ... < x_N = b, \quad h_i = x_i - x_{i-1}, \quad i = \overline{1,N}$.

Функция *S(x)* является интерполяционным кубическим сплайном класса $C^2_{[a,\,b]}$ с узлами на сетке ω , если:

$$S(x) = \sum_{k=0}^{3} a_k^i (x - x_i)^k \ , x \in [x_i, x_{i+1}] , i = \overline{0, N-1} \ ;$$

$$S(x_i) = f_i , i = \overline{0,N} \ ; \ S(x) \in C^2[a,b] .$$

Для построения таких сплайнов необходимо задать два дополнительных условия. На равномерной сетке, если использовать условия отсутствия узла, систему уравнений для определения моментов можно записать в виде

$$\begin{aligned} &- M_0 + 2M_1 - M_2 = 0 , \\ &M_{i-1} + 4M_i + M_{i+1} = 6\Lambda f_i , \ i = \overline{1, N-1} . \\ &- M_{N-2} + 2M_{N-1} - M_N = 0. \end{aligned} \qquad (1)$$

Исключая из первых и последних двух уравнений этой системы соответственно M_0 и M_N , получим систему с трехдиагональной матрицей

$$\begin{aligned} &M_1 = (f_0 - 2f_1 + f_2)/h^2 , M_{N-1} = (f_{N-2} - 2f_{N-1} + f_N)/h^2 , \\ &M_{i-1} + 4M_i + M_{i+1} = 6\Lambda f_i , \ i = \overline{2, N-2}. \end{aligned} \qquad (2)$$

Моменты M_0 , M_N определяются после решения системы (2)

$$M_0 = 2M_1 - M_2 \ , M_N = 2M_{N-1} - M_{N-2} . \qquad (3)$$

На каждом интервале для вычисления сплайна и его производных имеют место формулы

$$\begin{aligned} &S(x) = (1-t) f_i + t f_{i+1} + \frac{t}{6}(1-t) h_{i+1}^2 \left[(2-t) M_i + (1+t) M_{i+1} \right], \\ &S'(x) = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{h_{i+1}}{6} \left[(2 - 6t + 3t^2) M_i + (1 - 3t^2) M_{i+1} \right], \end{aligned} \qquad (4)$$

$S''(x) = (1-t)M_i + tM_{i+1}$, где $t = \dfrac{x - x_i}{h_{i+1}}$, $h_{i+1} = x_{i+1} - x_i$, $x \in [x_i, x_{i+1}]$, $i = \overline{0, N-1}$.

Построение кубических сплайнов через моменты на многопроцессорных вычислительных системах приводит к необходимости распараллеливания метода прогонки [3].

Пусть $N = Pq$, где P – число процессоров, в памяти которых размещены соответственно данные:

$x_0, x_1, \cdots, x_q, x_{q+1}$;

$x_{q-1}, x_q, \cdots, x_{2q}, x_{2q+1}$;

.....

$x_{N-q-1}, x_{N-q}, \cdots, x_{N-1}, x_N$.

Такое расположение данных соответствует декомпозиции области ω с перекрытием на шаг сетки, фрагмент которой изображен на рис. 1.

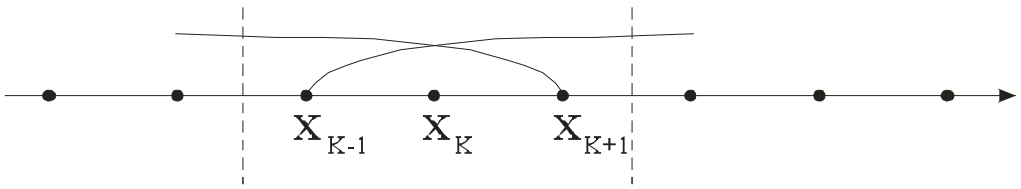


Рис. 1

Процесс построения сплайна в этом случае состоит из трех этапов:

1. Распределение исходных данных в каждом процессоре в соответствии с декомпозицией сетки ω.

2. Вычисление моментов на границах интервала декомпозиции области

$M_k = (f_{k-1} - 2f_k + f_{k+1})/h^2$,

$M_{k+q} = (f_{k+q-1} - 2f_{k+q} + f_{k+q+1})/h^2$,

$k = 1, q, 2q, 3q, \cdots, (P-1)q - 1$.

3. Решение методом прогонки в каждом процессоре трехдиагональной системы для моментов. Определение по формулам (3) моментов M_0 и M_N.

В этом случае погрешность в определении моментов $O(h^2)$, а для их вычисления потребуется $8q+1$ операций на P процессорах. В рассматриваемом алгоритме массив первых прогоночных коэффициентов одинаков для всех процессоров. Поэтому при достаточно больших $q = KL$ в каждом процессоре, содержащем $KL+2$ точек, вводится разбиение на K перекрывающихся подынтервалов. Для первой группы из L точек в каждом процессоре расчеты осуществляются с одновременным определением двух массивов прогоночных коэффициентов. Для всех последующих $(k-1)$ групп в прямой прогонке пересчитывается только второй прогоночный коэффициент.

Тогда в каждом процессоре на вычисления будет затрачено $(8L+1)+(K-1)(5L+3)$ операций. Для вычислений по формулам (4) при заданном x в каждом процессоре предварительно проверяется выполнение условия $x \in [x_{KLi}, x_{KL(i+1)}],\ i = 0,1,...,P-1$. После определения номера процессора, содержащего необходимый интервал и исходные данные, осуществляются вычисления по формулам (2). В рассмотренном алгоритме условие отсутствия узла выполняется только на первых и последних двух интервалах сетки ω.

Отметим, что если при вычислении интеграла $I = \int_a^b f(x)dx$ подынтегральную функцию $f(x)$ заменить кубическим сплайном $S(x)$, то

$$\int_a^b S(x)dx = \frac{1}{2}\sum_{i=0}^{N-1} h_{i+1}(f_i + f_{i+1}) - \frac{1}{24}\sum_{i=0}^{N-1} h_{i+1}^3(M_i + M_{i+1}). \qquad (5)$$

Тогда для вычисления интеграла через моменты на P процессорах получим

$$I = \sum_{i=0}^{P-1} \int_{A_i}^{B_i} S(x)dx,\ A_i = x_{KLi}.\ B_i = x_{KL(i+1)}.$$

Литература

1. Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л. Методы сплайн-функций. – М.: Наука, 1980. – 352 с.
2. Берцун В. Н. Сплайны сеточных функций. – Томск: Изд-во Том. ун-та, 2002. – 124 с.
3. Вшивков В.А. О распараллеливании вычислительных алгоритмов // Сибирская школа-семинар по параллельным вычислениям. – Томск: Изд-во Том. ун-та, 2002. – С. 46–59.

БИБЛИОТЕКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ МОДЕЛИРОВАНИЯ КРИТИЧЕСКОГО ПОВЕДЕНИЯ МАКРОСКОПИЧЕСКИХ НЕУПОРЯДОЧЕННЫХ СИСТЕМ

А.Н. Вакилов, В.В. Прудников, Е.Л. Филиканов

Омский государственный университет

В ОмГУ в научно-исследовательской лаборатории прикладной теоретической физики и параллельных вычислений (руководитель - проф. В.В. Прудников) в течение ряда лет проводятся плодотворные исследования по направлению “Компьютерное моделирование критического поведения неупорядоченных систем”, поддержанные грантами РФФИ (97-02-16124 , 00-02-16455) и Минобразования [1–6].

Известно, что критические свойства многих материалов можно описать с помощью модели Изинга с гамильтонианом:

$$H = -J\sum_{ij} S_i S_j ,$$

где S_i – спиновые переменные, расположенные в узлах d-мерной решетки и принимающие значения +1 и –1. Энергия спин-спинового взаимодействия J ограничена взаимодействием ближайших соседей. Константа обменного взаимодействия J является мерой силы взаимодействия между соседними спинами. Если $J > 0$, то состояния, которые характеризуются одинаковой ориентацией спинов ближайших соседей, энергетически выгоднее по сравнению с состояниями, у которых соседние спины ориентированы в противоположные стороны. Следовательно, можно ожидать, что для $J > 0$ состояние с наименьшей энергией является ферромагнитным, т.е. в среднем суммарное число спинов, ориентированных в одном направлении, не равно нулю. Если $J < 0$, то предпочтительнее с энергетической точки зрения состояния, для которых соседние спины антипараллельны, и можно ожидать, что состояние с наименьшей энергией является антиферромагнитным, т.е. спины сонаправлены через один. Важная роль статистической теории модели Изинга объясняется тем, что она находит применение при рассмотрении самых разнообразных магнитных и немагнитных систем. Сюда входят ферромагнетики, антиферромагнетики, ферримагнетики, бинарные смеси и сплавы, решеточная модель жидкости, адсорбция на поверхности, «плавление» ДНК и т.д. Поэтому статистика модели Изинга занимает видное место среди других вопросов статистической механики.

Макроскопические свойства системы характеризуются следующими величинами. Так, суммарный магнитный момент, или намагниченность, задается выражением

$$M = \sum_{i=1}^{N} S_i \ .$$

Теплоемкость C при постоянном внешнем магнитном поле может быть определена соотношением

$$C = \frac{d < H >}{dT} = \frac{\left(< H^2 > - < H >^2\right)}{kT^2},$$

а изотермическая магнитная восприимчивость при нулевом магнитном поле может быть задана формулой

$$\chi = \frac{\left(< M^2 > - < M >^2\right)}{kT}.$$

Для вычисления средних характеристик системы, например средней энергии или средней намагниченности, нужно сконструировать марковский процесс, когда каждая спиновая конфигурация $\{S_{i+1}\}$ получается из предыдущей конфигурации $\{S_i\}$ с помощью соответствующей вероятности перехода $W(\{S_i\} \to \{S_{i+1}\})$. Метрополис и др. (1953) указали, что можно выбрать W такой, что в пределе большого числа конфигураций функция распределения $P(\{S_i\})$ спиновых состояний, генерируемых марковским процессом, стремится к равновесному каноническому распределению $P_{EQ}(\{S_i\}) = \frac{1}{Z}\exp(-H/kT)$. Достаточным условием для достижения этого распределения является выполнение принципа детального равновесия

$$P_{EQ}(\{S_i\})W(\{S_i\} \to \{S_{i'}\}) = P_{EQ}(\{S_{i'}\})W(\{S_{i'}\} \to \{S_i\}).$$

Данное условие не задает W единственным образом, оставляя произвол в выборе W. Наиболее часто в качестве вероятности перехода используется функция Метрополиса

$$W\left(\{S_i\} \to \{S_{i'}\}\right) = \begin{cases} \exp\left(-\Delta H \big/ kT\right), & при \quad \Delta H > 0, \\ 1, & при \quad \Delta H \le 0, \end{cases}$$

Поведение ряда физических величин для непрерывных фазовых переходов в окрестности T_c характеризуется набором критических индексов. Так, вблизи T_c можно определить поведение намагниченности $M(T)$, восприимчивости $\chi(T)$ и теплоемкости $C(T)$ следующим образом:

$$M(T) \sim (T_c - T)^{\beta}\ ,\ \chi(T) \sim |T_c - T|^{-\gamma}\ ,\ C(T) \sim |T_c - T|^{-\alpha}\ ,$$

где α, β, γ – критические индексы. Теплоемкость и восприимчивость при $T = T_c$ расходятся. Свойства систем при непрерывных фазовых переходах определяются сильными и долгоживущими флуктуациями параметра порядка. Мерой магнитных флуктуаций является линейный размер $\xi(T)$ ха-

рактерного магнитного домена – области с сильно коррелированными спинами. При $T >> T_c$ длина корреляции $\xi(T)$ по порядку величины равна периоду решетки. Поскольку по мере приближения T к T_c сверху корреляция в ориентации спинов увеличивается, $\xi(T)$ будет возрастать при приближении T к T_c. Расходимость $\xi(T)$ в окрестности T_c описывается критическим индексом ν:

$$\xi(T) \sim |T - T_c|^{-\nu}.$$

Из-за долгоживущих флуктуаций намагниченности время релаксации системы τ_p, а также время корреляции флуктуаций τ в окрестности T_c неограниченно возрастают. Можно ввести динамический критический индекс z, определяемый соотношением

$$\tau_p, \tau \sim (T - T_c)^{-z\nu}.$$

Если проведено n последовательных наблюдений M_i $(i = 1,2,...,n,\ где\ n >> 1)$ намагниченности M, то среднеквадратичная статистическая погрешность будет [7]:

$$\langle (\delta M)^2 \rangle \approx 2\frac{\tau_M}{t_{obs}}\left[\langle M^2 \rangle - \langle M \rangle^2\right],$$

τ_M - время релаксации намагниченности; t_{obs} – время наблюдения. При $T = T_c$, $\tau_M \sim L^z$; где L – линейный размер системы; z – динамический критический индекс. Таким образом, очень сложно получить данные с высокой точностью при $T = T_c$, когда погрешность определения намагниченности M максимальна:

$$\langle (\delta M)^2 \rangle \approx 2\frac{\tau_M}{t_{obs}}\left[\langle M^2 \rangle - \langle M \rangle^2\right] \sim \frac{L^{z+\gamma/\nu-d}}{t_{obs}}, \text{ при } T = T_c.$$

Поэтому для “последовательного” компьютера время центрального процессора, необходимое для выполнения одного шага на спин по методу Монте-Карло, растет с увеличением размера системы как $L^{z+\gamma/\nu}$ независимо от d при $d \le 4$. Увеличение в 10 раз линейного размера системы потребует в 10^4 раз больше времени центрального процессора. Поэтому применение компьютеров с параллельной архитектурой, для которых выполнение одного Монте-Карло-шага на спин не столь существенно, становится необходимым для Монте-Карло исследований критического поведения термодинамических систем.

В лаборатории разрабатываются параллельные алгоритмы для проведения моделирования критического поведения неупорядоченных систем. Эти алгоритмы можно разделить на два класса: 1) алгоритмы с переворотом одного спина, включая алгоритм Метрополиса; 2) кластерные алгоритмы Свендсена-Ванга, Вольфа. Имеются следующие возможности использования указанных алгоритмов на кластере: вся программа помещается в процессор и можно набирать статистику независимо, заставляя все процес-

сы работать асинхронно и обрабатывая затем накопленные результаты; для моделирования решеток больших размеров необходимо формировать массивы определенной длины из независимых спинов на решетке с учетом топологии расчетов.

В настоящее время разработаны и реализованы на алгоритмическом языке Си с использованием MPI для двумерной и трехмерной модели Изинга алгоритмы Вольфа, Свендсена–Ванга и Метрополиса. Данный пакет программ позволяет проводить моделирование магнитных систем с большим числом спинов (до 200 миллионов). В дальнейшем предполагается создание пакета параллельных программ с использованием алгоритмов Свендсена–Ванга, Метрополиса и кластерных алгоритмов для систем во внешнем случайном поле и спиновых стекол.

Алгоритм, разработанный Свендсеном–Вангом и Вольфом, совершает крупномасштабные, нелокальные изменения в спиновой конфигурации и значительно уменьшает эффект критического замедления. В этом так называемом кластерном алгоритме целый кластер спинов меняет своё направление для каждого шага процедуры Монте-Карло. Кластеры формируются заданием связи между соседними спинами с помощью вероятностной процедуры, от которой зависит вид модели и алгоритма. В алгоритме Свендсена–Ванга связь между соседними спинами считается всегда разорванной, если спины противоположно направлены, если спины сонаправлены, то связь считается замкнутой с вероятностью $1–exp(–2/T)$ и разорванной с вероятностью $exp(–2/T)$. Эта процедура создаёт кластеры взаимодействующих спинов. Новая конфигурация получается путём присваивания спинам каждого кластера значения +1 или –1 с равной вероятностью. В алгоритме Вольфа создаётся только один кластер. В решётке случайным образом выбирается центральный спин, затем вокруг него строится кластер с использованием вероятностного характера связи, как в алгоритме Свендсена–Ванга. Все спины в этом кластере переворачиваются, изменяя свой знак на противоположный (переворот кластера). Время в методе Монте-Карло удобно измерять в количестве последовательных статистически независимых конфигураций. Главная вычислительная задача алгоритма Свендсена–Ванга – это идентификация и маркировка кластеров, дающая новую конфигурацию. Алгоритмы Метрополиса и Свендсена–Ванга наиболее подходят для распараллеливания, чем алгоритм Вольфа, так как все спины решетки участвуют в переворотах. В этом случае можно использовать стандартное разбиение системы, например, поделить решётку на равные участки между процессорами. Однако этого нельзя сделать в случае алгоритма Вольфа, в котором строится только один кластер вокруг случайного участка решётки. В этом случае стандартное разбиение не дает равномерной загруженности процессоров. Алгоритм Вольфа принадлежит к типу задач, в которых весь вычислительный процесс имеет тенденцию концентрироваться в различные времена в ограниченных областях данных. Для тако-

го рода задач применяется разбиение всех локальных участков данных на равные части среди процессоров. Такое разбиение приводит к увеличению затрат времени на коммуникацию между процессорами, однако это может быть скомпенсировано равномерной загруженностью процессоров.

Для алгоритма Вольфа применялось разбиение решётки линейным размером L на N равных полос шириной W так, чтобы $L = WN$, N/P - было целым числом ($1 \le W \le L/P$), где P – число процессоров. Теперь каждому процессору принадлежит N/P число полос, на которые разбита исходная решётка. Эти полосы составляют подрешётку своего процессора. Вводится набор локальных и глобальных координат спинов. Глобальные координаты спинов – это координаты решётки. Локальные координаты – это координаты отдельной подрешётки $a = i, b = (j \bmod WP) \bmod W + W(j \, div WP)$.

Благодаря такому разбиению вычислительная задача распределяется между процессорами более равномерно, чем при стандартном разбиении. В итоге каждый процессор получает некоторое число полос и спинов. Один из процессоров выбирает случайным образом центральный спин и посылает на все остальные.

Каждый процессор, как и в обычном алгоритме Вольфа, строит кластер в своей подрешётке с использованием специальных граничных условий. Если рост частей кластера на всех процессорах завершился, центральный процесс собирает и накапливает данные. По завершении необходимого числа шагов Монте-Карло 0-процессор вычисляет средние значения физических величин. Применение специальных граничных условий связано с особенностью разбиения задачи. По локальной координате a реализуются обычные циклические граничные условия. Иначе обстоит дело с локальной координатой b, так как на левой и правой границах каждой полосы, входящей в подрешётку, необходим обмен информацией между процессорами.

$b \bmod W = 0$ – спин находится на левой границе полосы,

$(b + 1) \bmod W = 0$ – спин находится на правой границе полосы.

Если координата спина b удовлетворяет первому из этих условий, процессор посылает координаты соседа, находящегося слева, процессору, на подрешётке которого этот соседний спин находится. Для передачи координат спина с целью упрощения локальные координаты снова переводятся в глобальные:

$i = a$,

$j = (b \, div W)PW + pW + b \bmod$, где p – номер процессора.

Параллельный алгоритм Вольфа [8] позволит моделировать решётки большого размера, что, как следствие, снизит эффекты конечных размеров моделируемой системы спинов на критические явления.

Литература

1. Вакилов А.Н., Прудников В.В. Компьютерное моделирование критической динамики разбавленных магнетиков // Письма в ЖЭТФ. – 1992. – Т.55, №12. – С. 709–712.
2. Прудников В.В., Вакилов А.Н. Компьютерное моделирование критической динамики разбавленных магнетиков // ЖЭТФ. – 1993. – Т.103, №3. – С. 962–969.
3. Марков О.Н., Прудников В.В. Компьютерное моделирование критической динамики неупорядоченных двумерных изинговских систем // Письма в ЖЭТФ. – 1994. – Т.60, №1. – С. 24–29.
4. Prudnikov V.V., Markov O.N. Critical dynamics of disordered two-dimensional Ising systems: a Monte Carlo study // J. Phys. A: Math. Gen. – 1995. – Vol.2. – P. 47–49.
5. Prudnikov V.V., Markov O.N. Monte Carlo renormalization group of dilute 2D Ising dynamics // Europhys. Lett. – 1995. – Vol.29, №3. – P. 245–250.
6. Прудников В.В., Марков О.Н., Осинцев Е.В. Особенности фазовых превращений в неупорядоченной антиферромагнитной модели Изинга // ЖЭТФ. – 1999. – Т.116, №3. – С. 953–961.
7. Хеерман Д.В. Методы компьютерного эксперимента в теоретической физике. – М.: Наука, 1990.
8. Bae S. and Ko S.H. Parallel Wolff Cluster Algorithms. School of Computer Science, Syracuse University, Syracuse, NY 13244. U.S.A., 1994.

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ ЗАДАЧ МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ[1]

В.Ю. Казаков, С.В. Тимченко

Томский государственный университет, Томский государственный университет систем управления и радиоэлектроники

При решении большого числа практических задач приходится сталкиваться с необходимостью нахождения решений, удовлетворяющих нескольким, зачастую конфликтующим между собой критериям. В связи с этим решение задачи заключается не в нахождении какого-то одного решения, а в отыскании некоторого множества решений, каждое их которых будет превосходить другие хотя бы по одному критерию. Такие решения, как правило, называются оптимальными по Парето. Необходимость отыскания целого множества решений чрезвычайно усложняет задачу оптимизации и делает практически непригодными большинство классических методов оптимизации. Задача усложняется еще и тем, что надо не только найти решения, максимально близкие к истинному множеству (или фронту) Парето, но и обеспечить максимально возможное различие между такими решениями (т.е. охватить возможно большую часть этого фронта).

Один из наиболее перспективных подходов к решению данной задачи заключается в применении генетических или эволюционных алгоритмов, работающих сразу с целым набором испытаний (точек в пространстве поиска) и позволяющих находить весь фронт Парето за один проход.

1.Введение

Принципы многокритериальной оптимизации существенно отличаются от обычной оптимизации. Во втором случае (один критерий) целью решения задачи является нахождение глобального решения, дающего оптимальное значение для одной целевой функции. В случае нескольких критериев мы имеем соответственно несколько целевых функций, у каждой из которых может быть оптимальное значение при своем собственном наборе значений независимых переменных. Если оптимальные решения для различных целевых функций существенно различны, то невозможно говорить об оптимальном решении всей задачи в целом. В этом случае мы получаем множество оптимальных решений, ни одно из которых не является оптимальным по сравнению с другими во всех смыслах (т.е. по всем критери-

[1] Работа выполнена при поддержке гранта РФФИ 02-01-01022.

ям). Это множество называют множеством решений, оптимальных по Парето. Проиллюстрируем сущность многокритериальной задачи на гипотетическом примере, состоящем в удовлетворении двух критериев оптимальности – цена и качество некоторого изделия (рис.1).

Здесь по оси абсцисс откладывается условная цена некоторого изделия, а по оси ординат – величина, обратная его качеству. На этом рисунке точка *A* представляет решение, близкое к оптимальному по цене, но совершенно неудовлетворительное по качеству. Наоборот, точка *B* представляет собой изделие высокого качества, но и одновременно очень дорогое. Существуют также точки (такие, как точка *D*), принадлежащие фронту Парето, каждая из которых, превосходя какую-либо точку из этого фронта по одному критерию, обязательно уступает ей по другому.

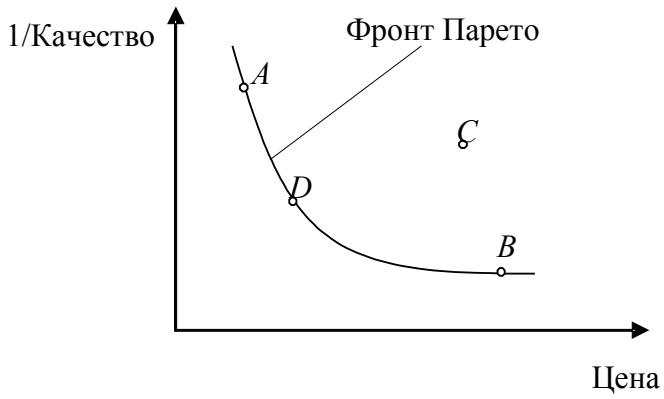


Рис. 1

Рассмотрим теперь точку, не являющуюся оптимальной по Парето (например, точку *C*). Эта точка, превосходя точку *A* по критерию качества, не является оптимальной по Парето, так как существуют решения (например, точка *D*), превосходящие эту точку по обоим критериям.

Для многокритериальной проблемы (пусть f_i – целевые функции, $i = 1,2,...,M,\ M > 1$) для любых двух точек в пространстве поиска существуют две возможности – доминирует одно решение над другим или нет. Говорят, что решение $x^{(1)}$ доминирует над решением $x^{(2)}$, если выполняются два условия:

1. Решение $x^{(1)}$ не хуже решения $x^{(2)}$, т.е. для всех целевых функций $\forall\, i : f_i(x^{(1)}) \le f_i(x^{(2)})$ (в случае задачи минимизации).

2. Решение $x^{(1)}$ строго лучше решения $x^{(2)}$ хотя бы для одной функции, т.е. $\exists\, i : f_i(x^{(1)}) < f_i(x^{(2)})$.

Таким образом, множество оптимальных по Парето решений *P* определяется таким образом, что в пространстве поиска нет решений, доминирующих над решениями из множества P.

2. Постановка задачи

Рассмотрим задачу определения параметров траектории входа в атмосферу Земли, минимизирующих величину интегрального конвективного теплового потока в точке торможения затупленного тела при одновременной минимизации толщины теплозащитного покрытия. В качестве исходной математической модели для расчета теплового потока используются уравнения тонкого (гиперзвукового) вязкого ударного слоя с учетом неравновесного характера протекания химических реакций, многокомпонентной диффузии и сопряженного характера теплообмена на поверхности тела [1-3].

Задача оптимизации формулируется следующим образом: в пространстве непрерывных функций $V(t)$ и $H(t)$ (где $V(t)$ и $H(t)$ – соответственно скорость и высота полета в зависимости от времени) необходимо найти такие функции $V(t)$ и $H(t)$, чтобы достигался минимум функционала

$$Q(V, H, t_m) = \int_0^{t_m} q_w(H(t), V(t), R^*, k, ...)dt$$

при минимально возможной толщине теплозащитного покрытия. Здесь подынтегральная функция $q_w(H(t), V(t), R^*, k, ...)$ представляет собой тепловой поток в критическую точку тела, который определяется из численного решения сопряженной задачи при помощи метода [9]. Искомые функции $V(t)$ и $H(t)$ ищутся в классе линий Безье m-го порядка.

Задача решается при ограничениях на равновесную температуру поверхности $T_w(t)$, которая не должна превышать некоторого заданного значения $T_w^{\max}(t)$, и температуру внутренней поверхности оболочки $T_i(t)$, которая также не должна превышать некоторого заданного значения $T_i^{\max}(t)$

Очевидными являются требования на крайние точки искомых функций:

$$V(0) = V_0,\ H(0) = H_0,\ V^{TM} = V^*,\ H^{TM} = H^*.$$

Другие ограничения на область допустимых значений поставленной задачи могут быть получены из физического анализа задачи и являются следствиями системы уравнений динамики полета. В частности, необходимыми являются ограничение на максимально допустимую перегрузку и дополнительное ограничение на ускорение, смысл которого в том, что торможение не может быть больше торможения, вызываемого максимальной силой сопротивления для данного тела на данной высоте:

$$\left|\dot{V}(t)\right| < ag,\ \left|\dot{V}(t)\right| < S^* \rho_\infty(t) V^2(t)/(2m).$$

Для учета ограничений на допустимые решения при реализации алгоритма поиска оптимального решения использовалась модифицированная целевая функция Q^*:

$$Q^* = \begin{cases} q_1 + q_2\left(\left|\dot{V}(t)\right| / g - a\right) & if \left|\dot{V}(t)\right| > ag \\ q_3 + q_4\left(T_w^{\max} - T_w\right) / T_w^{\max} & if\ T_w > T_w^{\max} \\ q_5 + q_6\left(T_i^{\max} - T_i\right) / T_i^{\max} & if\ T_i > T_i^{\max} \\ q_7 + q_8\left(\left|\dot{V}(t)\right| / g - A(t) / g\right) & if \left|\dot{V}(t)\right| > A(t) \end{cases}$$
$$Q$$

3. Метод решения

Для решения поставленной задачи в данной работе предлагается многокритериальный асинхронный параллельный генетический алгоритм (МАПГА), являющийся развитием однокритериального алгоритма, предложенного в [2–6]. В его основе лежит ранжированная турнирная селекция, основанная на частичном упорядочивании точек в пространстве поиска. На первом этапе ранжирования (частичного упорядочивания) в текущей популяции выбираются точки, лежащие во фронте Парето, им присваивается нулевой ранг, после этого они исключаются из рассмотрения. На втором этапе в оставшейся части популяции выбираются особи, для которых нет доминирующих решений. Далее этим особям присваивается первый ранг, и они также исключаются из рассмотрения. Процесс повторяется до исчерпания популяции. После этого в процессе селекции используются не значения целевых функций, а присвоенные особям ранги.

МАПГА состоит из следующих основных шагов:

1. На основе случайного поиска на master-процессе определяется первоначальная популяция.

2. Для вычисления целевой функция индивидуумы рассылаются slave-процессам.

3. Промежуточная популяция инициализируется множеством наилучших по Парето решений из предыдущей популяции.

4. Master-процесс принимает сообщения от готовых slave-процессов и включает соответствующее решение в промежуточную популяцию. Важно отметить, что вследствие асинхронности алгоритма потомки особей из k-й генерации могут быть включены как в $k+1$-ю генерацию, так и в $k+m$-ю ($m>1$) генерацию в зависимости от времени расчета целевой функции. После этого, если промежуточная популяция полностью сформирована, она подвергается ранжированию, копируется в текущую популяцию и, если не выполняется некоторый критерий завершения итерационного процесса, то алгоритм возобновляется, начиная с 3-го шага.

5. Если популяция сформирована еще не полностью, то из предыдущей популяции при помощи оператора селекции выбираются две новые родительские особи, к которым применяются операторы скрещивания и мутации для производства новых пробных точек в пространстве поиска.

При этом, как правило, используются оператор арифметического скрещивания [8] и оператор мутации с вероятностью пропорциональной расстоянию между родительскими особями [7].

6. Для вычисления целевой функции полученные индивидуумы посылаются свободным slave-процессам, и алгоритм возобновляется, начиная с 4-го шага.

Ниже приводится краткий псевдокод описанного алгоритма.

```
t := 0
initpopulation P(t)          /* random or random + initial solution(s) */
Scatter P(t)                 /* for evaluation */
while not done do
P' := Pareto_set(P)          /* Pareto elitism */
while size(P') < size(P) do
Recv_from_j (Pi, f(Pi))
P' := P' + Pi
Pi:= selectparents P(t)      /* ranking tournament selection or ranking
                                tournament selection with sharing*/
recombine Pi          /* single point, uniform or arithmetical crossover */
mutate Pi               /* nonuniform + distance-dependent mutation */
Send_to_j(Pi)                /* for evaluation */
enddo
P(t+1) := ranking P'(t)
t := t + 1
enddo
```

Предложенный параллельный алгоритм для решения задач многокритериальной оптимизации использовался для решения описанной в п. 2 задачи. На примере задачи определения параметров оптимальной траектории входа в атмосферу Земли показано, что задача параметрического исследования может быть сведена к поиску фронта Парето в задаче много-

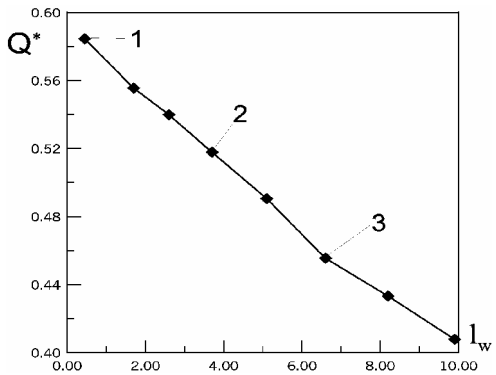


Рис. 2

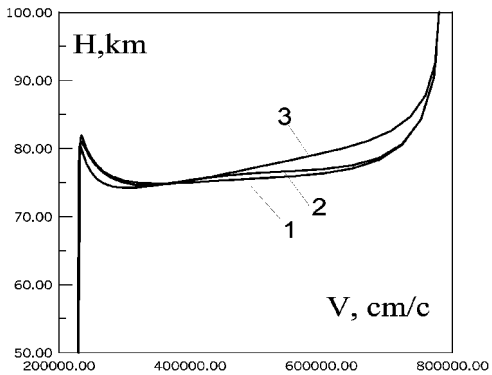


Рис. 3

критериальной оптимизации, в которой в качестве дополнительного критерия выступает ее варьируемый параметр. Проведенные расчеты показали, что данный подход позволяет существенно сократить время решения задачи. При этом параллельность алгоритма позволяет не только эффективно использовать имеющиеся вычислительные мощности, но и предотвратить стягивание фронта Парето в точку. Некоторые результаты расчетов приведены на рис. 2-3 (кривые 1–3 на рис. 3 соответствуют точкам 1–2 на рис.2).

Литература

1. Peigin S.V.,Tirskii G.A. et al. Super- and hypersonic aerodynamics and heat transfer. – New-York: CRC Pres, 1993.
2. Казаков В.Ю., Пейгин С.В., Тимченко С.В. Оптимизация траектории входа в атмосферу Земли по интегральному тепловому потоку // ЖПМТФ. – 2000. – Т.41, № 4. – С. 112–121.
3. Peigin S.V., Desidery J.-A., Timchenko S.V. Reentry Trajectory Optimization Using Genetic Algorithms // Computational Fluid Dynamics'2000. Proc. of the Fifht European CFD Conference. – 2000. – P. 14.
4. Peigin S., Periaux J., Timchenko S. Asynchrony Parallel Genetic Algorithm for Heat Flux Optimization Problem // Parallel Computational Fluid Dynamics. Development and Applications of Parallel Technology. Elsevier Science B.V., 1999. – P. 377–384.
5. Peigin S., Mantel B., Timchenko S. et al. Application of a genetic algorithm to a heat flux optimization problem // Surveys on Mathematics for Industry. – 2000. – Vol.9, № 3.
6. Hoffmeister F., Back T. Genetic algorithms and evolution strategies: similarities and differences // Parallel Problem Solving from Nature - Proceedings of 1st Workshop. – 1991. – P. 455–469.
7. Sefioui M., Periaux J., Ganascia J.-G. Fast convergence thanks to diversity // Proc. of the 5th Annual Conference on Evolutionary Programming. – MIT Press, 1996.
8. Michalewicz Z. Genetic algorithms + data structures = evolution programs. – New York: Springer-Verlag, 1992.
9. Петухов И.В. Численный расчет двумерных течений в пограничном слое // Численные методы решения дифференциальных и интегральных уравнений и квадратурные формулы. – М.: Наука, 1964. – С. 305–325.

ОЦЕНКА ЭФФЕКТИВНОСТИ РАСПАРАЛЛЕЛИВАНИЯ ЯВНЫХ РАЗНОСТНЫХ СХЕМ ДЛЯ РЕШЕНИЯ УРАВНЕНИЙ ГАЗОВОЙ ДИНАМИКИ

Л.Л. Миньков, Е.В. Пикущак

Томский государственный университет

В данной работе проводится оценка эффективности и ускорения параллельных алгоритмов решения задачи о распаде произвольного разрыва в идеальном газе, описываемого системой нестационарных газодинамических уравнений в одномерном приближении, и решения задачи о стационарном течении невязкого газа в сопле Лаваля, получаемого на основе метода установления, в двумерном осесимметричном приближении.

1. Задача о распаде произвольного разрыва

Система одномерных уравнений газовой динамики имеет вид

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} = 0, \qquad (1)$$

где U – вектор консервативных переменных; F – вектор потоков:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}; \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u H \end{pmatrix};$$

ρ - плотность; u – скорость; p – давление; E – полная энергия, $E = e + \frac{1}{2}u^2$;

H – энтальпия, $H = E + \frac{p}{\rho}$.

Для замыкания системы уравнений (1), представляющей собой дифференциальную форму законов сохранения, привлекается уравнение состояния, которое для совершенного газа имеет вид: $e = \frac{p}{\rho(k-1)}$, k – отношение удельных теплоемкостей при постоянном давлении и постоянном объеме.

В начальный момент времени задаются следующие условия:

$\rho = 1,\ u = 0,\ p = 1$ для $x < 0$;

$\rho = 0.125,\ u = 0,\ p = 0.1$ для $x > 0$.

Численное решение данной задачи проводится с использованием явной разностной схемы. Область решения по пространству разбивается на

N отрезков (ячеек). Разностный аналог системы (1) записывается в следующем виде:

$$\frac{U_i^{n+1} - U_i^n}{\tau} + \frac{F_{i+1/2} - F_{i-1/2}}{h} = 0. \qquad (2)$$

Система (2) решается в два этапа. Сначала по известным на нижнем временном слое t_n значениям параметров в ячейках U_i^n вычисляются потоки $F_{i-1/2}$ на гранях ячеек, где i – номер ячейки, $i = 1,\ldots,N$, затем с помощью (2) вычисляют U_i^{n+1} на верхнем временном слое t_{n+1}.

Для вычисления потоков на гранях ячеек используется либо метод Ван Лира [1], либо метод С.К.Годунова [2].

Для распараллеливания полученных последовательных алгоритмов применяется геометрическая декомпозиция, т.е. каждый процесс, получая исходные данные, находит решение поставленной задачи в отведенной ему геометрической подобласти (на отрезке области интегрирования по пространству). Чтобы определить значения потоков на границах отрезка, на каждом процессе вводятся две дополнительные граничные ячейки, куда соседние процессы посылают известные значения величин U_1^n и U_M^n (рис.1).

Здесь M – количество ячеек, отводящихся одному процессу. Переда-

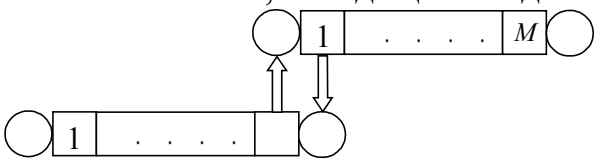


Рис. 1. Схема обмена данными между процессами

ча данных основана на концепции передачи сообщений с использованием стандарта MPI (Message Passing Interface) [3].

Полученные параллельные алгоритмы для решения исходной системы (1) были протестированы на вычислительном кластере ТГУ.

Ускорение $S_p = t_1/t_p$ и эффективность $E_p = S_p/p$ параллельных алгоритмов (t_1 – время выполнения алгоритма на одном процессоре, t_p – время выполнения алгоритма на системе из p процессоров) определяются в зависимости от числа процессов и числа разностных ячеек, приходящихся на всю область интегрирования. Число процессов изменялось от 1 до 16. Расчеты проводились на трех разностных сетках с числом ячеек N=1680, 3360, 6720.

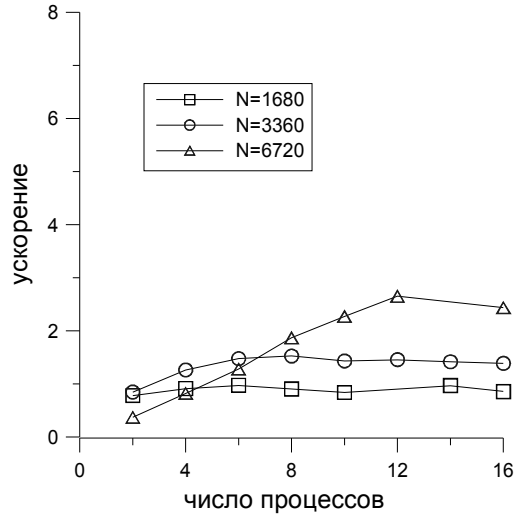


Рис. 2. Зависимость ускорения параллельного алгоритма для метода Ван- Лира от числа процессов

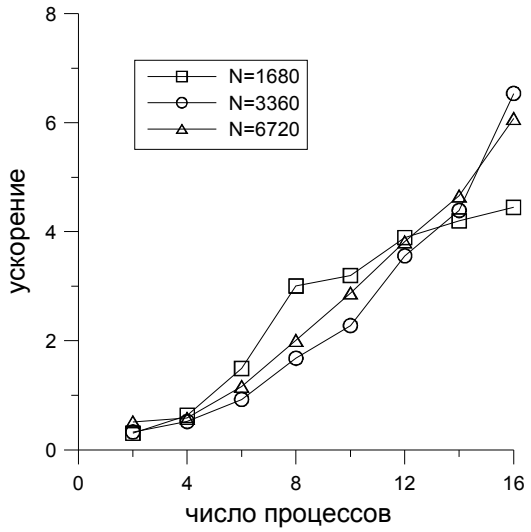


Рис. 3. Зависимость ускорения параллельного алгоритма для метода Годунова от числа процессов

На рис.2 и 3 показано ускорение параллельных алгоритмов для метода Ван Лира и метода Годунова соответственно. Для обоих методов ускорение примерно одинаково, если $p \leq 6$. При большем числе процессов параллельный алгоритм для метода Годунова имеет большее ускорение. Так, например, при $p = 16$ ускорение для метода Годунова получилось примерно в 2.5 раза выше, чем для метода Ван Лира. Из вышеуказанных рисунков видно, что при $p > 6$ увеличение числа разностных ячеек ведет к увеличению ускорения параллельного алгоритма.

Эффективность распараллеливания последовательных алгоритмов решения системы (1) показана на рис.4 и 5. Из рисунков видно, что эффективность алгоритмов невысока и не превышает 41%. Увеличение числа процессов ведет к снижению эффективности распараллеливания для метода Ван Лира (рис.4), в отличие от метода Годунова (рис.5). Такое поведение объясняется тем, что в методе Ван Лира число арифметических операций на одном шаге интегрирования гораздо меньше, чем в методе Годунова, что ведет к увеличению относительной доли временных затрат на выполнение операций обмена между процессами.

Очевидно, что при большем числе операций на одном шаге интегрирования относительная доля временных затрат на выполнение операций обмена между процессами будет уменьшаться. Поэтому при распараллеливании алгоритмов решения уравнений газовой динамики с помощью явных схем большей эффективности следует ожидать в задачах большей размерности.

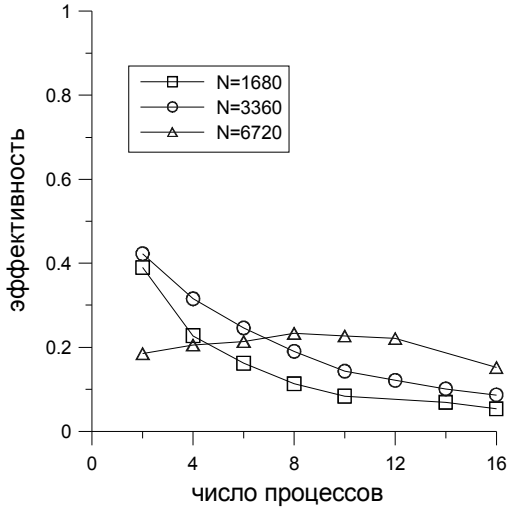


Рис. 4. Зависимость эффективности параллельного алгоритма для метода Ван Лира от числа процессов

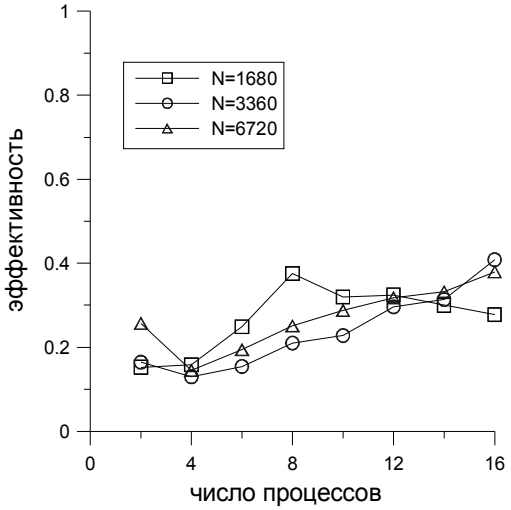


Рис. 5. Зависимость эффективности параллельного алгоритма для метода Годунова от числа процессов

2. Задача о течении газа в сопле Лаваля

Стационарное решение задачи о течении газа в сопле Лаваля находится методом установления из решения следующей нестационарной системы уравнений Эйлера:

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0, \tag{3}$$

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho \upsilon \\ \rho E \end{pmatrix}; \; F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho \upsilon u \\ \rho H u \end{pmatrix}; \; G = \begin{pmatrix} \rho \upsilon \\ \rho \upsilon u \\ \rho \upsilon^2 + p \\ \rho H \upsilon \end{pmatrix}.$$

Начальные условия задаются из решения соответствующей одномерной задачи:

$$\rho = \rho_0 \left(1 - \frac{k-1}{k+1}\lambda^2\right)^{\frac{1}{k-1}}; \; p = p_0 \left(1 - \frac{k-1}{k+1}\lambda^2\right)^{\frac{k}{k-1}}; \; V = \lambda \sqrt{\frac{2k}{k+1}\frac{p_0}{\rho_0}};$$

$$u = V\cos\theta(x), \; \upsilon = V\sin\theta(x),$$

где $\theta(x) = \dfrac{y}{R(x)}\theta_R(x)$; $\left(\dfrac{k+1}{2}\right)^{\frac{1}{k-1}}\lambda\left(1 - \dfrac{k-1}{k+1}\lambda^2\right)^{\frac{1}{k-1}} = \dfrac{1}{R^2(x)}$.

Для профиля сопла использовалась зависимость [4]:

$$R(x) = \begin{cases} 3.13, & \text{если } x \in [0, 1] \\ \sqrt{1^2 - (x-1)^2} + 2.13, & \text{если } x \in (1, 1.7071] \\ -x + 4.5442, & \text{если } x \in (1.7071, 3.3612] \\ -\sqrt{0.625^2 - (x - 3.8031)^2} + 1.625, & \text{если } x \in (3.3612, 3.9649] \\ 0.2679x - 0.0409, & \text{если } x \in (3.9649, 5]. \end{cases}$$

Здесь $\theta_R(x)$ – угол между касательной к профилю сопла и осью x; ρ_0, p_0 – параметры торможения газа.

Граничные условия:

на оси симметрии: $v = 0$; $\frac{\partial u}{\partial y} = \frac{\partial p}{\partial y} = \frac{\partial \rho}{\partial y} = 0$;

на непроницаемой поверхности: $\vec{V} \cdot \vec{n} = 0$ – условие непротекания;

на выходной границе: $p = p_a$, если $M < 1$;

во входной границе: $\frac{p}{\rho^k} = \Phi_0$; $v = 0$; $\frac{k}{k-1}\frac{p}{\rho} + \frac{u^2}{2} = H_0$.

Давление торможения принималось равным 6.2 МПа, плотность торможения 2.7 кг/м^3, отношение теплоемкостей $k = 1.4$.

Система уравнений (3) решается методом Годунова. Схема распараллеливания применяется такая же, как и для решения одномерной задачи (рис.1). Расчетная область разбивается на подобласти вдоль оси Ox. Но в отличие от одномерной задачи для обмена данными между процессами слева и справа от расчетной области, приходящейся на один процесс, вводятся не вспомогательные ячейки, а одномерные массивы, размер которых равняется количеству ячеек поперек области.

Условием окончания численного счета двумерной задачи является достижение невязки некоторого заданного значения, которая представляет собой следующую сумму:

$$\frac{1}{N*M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\left|\rho_{ij}^{n+1} - \rho_{ij}^{n}\right| + \left|u_{ij}^{n+1} - u_{ij}^{n}\right| + \left|v_{ij}^{n+1} - v_{ij}^{n}\right| + \left|p_{ij}^{n+1} - p_{ij}^{n}\right|\right).$$

Здесь N – количество разбиений в направлении оси Ox на одном процессе; M – количество разбиений вдоль оси Oy.

Из рис.6 видно, что ускорение возрастает при увеличении числа процессов и измельчении расчетной сетки, тогда как для одномерной задачи это практически не фиксируется (рис.2,3). Это можно объяснить тем, что в одномерном случае доля временных затрат на обмен информацией между процессами может быть соизмерима с долей времени простого численного счета, т.е. не намного его меньше. Для двумерной же задачи они не настолько соизмеримы.

Пунктирными линиями на рис.6 показана теоретически полученная зависимость ускорения от числа процессов по закону Амдаля для параллельного алгоритма двумерной задачи с соответствующей долей последовательных операций f=0.1 и f=0.2. Видно, что с измельчением разностной сетки доля последовательных операций уменьшается, так как увеличивается доля параллельных операций, выполняемых каждым процессом.

Из рис.7 видно, что эффективность параллельного алгоритма возрастает с увеличением числа расчетных узлов и падает при увеличении числа процессов, за исключением p=4. Падение эффективности при увеличении числа процессов можно объяснить тем, что операции обмена на вычислительном кластере происходят медленно. Еще одной причиной падения эффективности с увеличением числа процессов является несбалансированная нагрузка процессоров в полученном параллельном алгоритме. Нулевой процессор оказывается более нагруженным, чем все остальные процессоры. Все обменные операции проходят через него. Это объясняется физической топологией кластера ТГУ – «звездой». Процессоры связаны между собой только через центральный процессор (нулевой). Также считывание из файлов разностной сетки и начальных условий и их дальнейшее разбиение и рассылка другим процессам осуществляется именно нулевым процессом.

Значение эффективности для одномерной задачи не превышает 42 %. Из этого можно сделать вывод о том, что одномерную задачу распараллеливать нерационально. В ней все-таки значителен объем обмениваемой информацией по сравнению с последовательно выполняемыми на каждом

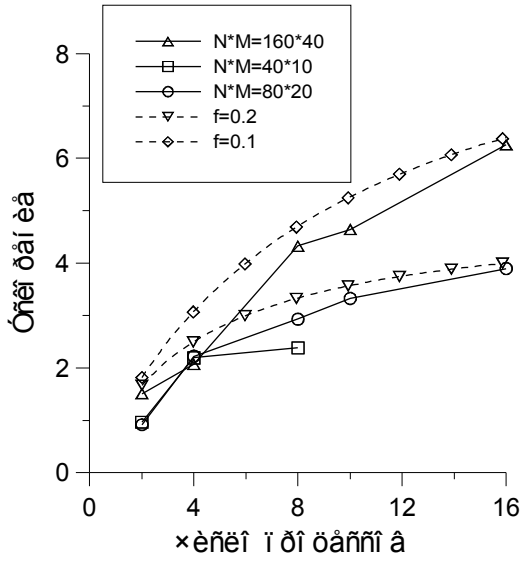


Рис. 6. Зависимость ускорения параллельного алгоритма для двумерной задачи от числа процессов

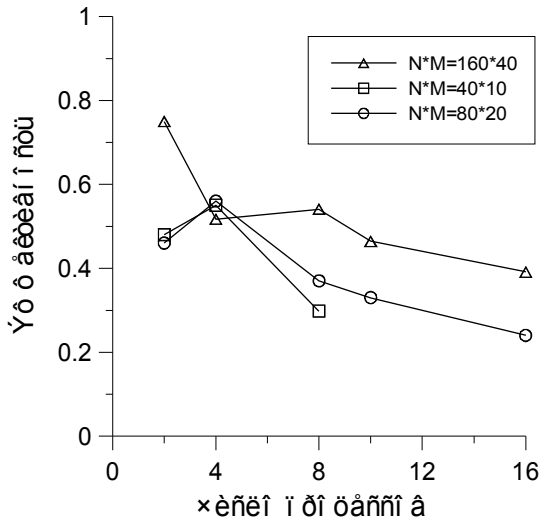


Рис. 7. Зависимость эффективности параллельного алгоритма для двумерной задачи от числа процессов

процессоре операциями.

Максимальная эффективность для двумерной задачи равна 76% . Она достигается на самой мелкой расчетной сетке 160*40 и на двух процессах.

Литература

1. Van Leer B. Towards the Ultimate Conservative Finite Difference Scheme III. Upstream centered Schemes for Hyperbolic Conservation Laws // J. Comp. Phys. – 1977. – Vol.7, №23. – P. 263–275.
2. Численное решение многомерных задач газовой динамики / Под ред. С.К.Годунова. – М.: Наука, 1976. – 400 с.
3. Корнеев В.Д. Параллельное программирование в MPI. – Новосибирск: Изд-во СО РАН, 2000. – 213 с.
4. Численные методы в механике жидкостей / Под ред. О.М. Белоцерковского. – М.: Мир, 1973. – 304 с.

ОПТИМИЗАЦИЯ КЛАСТЕРНЫХ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ КРИТЕРИЕВ СТРУКТУРНОЙ СЛОЖНОСТИ

В.Е. Подольский, С.С. Толстых

Тамбовский государственный технический университет

Эффективность использования вычислительных кластеров во многом зависит от выбора структуры взаимодействия компьютеров в сети при решении конкретных задач. Использование вычислительных кластеров для решения прикладных задач тем легче (т.е. требует меньшей квалификации программиста), чем уже проблема, и, в особенности, проще структура задачи. В нашей работе основное внимание уделяется решению структурно–сложных задач. Рассматривается комплексная математическая модель, состоящая из специализированных блоков, предназначенных для решения типовых задач (системы уравнений, численное интегрирование, комбинаторика, адаптированные под эмпирику датчики случайных чисел и др.). Считается, что каждый блок сложной модели процедурно обеспечен и работает, в свою очередь, с использованием параллельных вычислений. Вместе с тем соединение блоков в единую модель остается проблематичным. Необходимость решения подобных задач становится тем очевиднее, чем выше производительность кластера. Примерами задач могут служить многие уже разработанные математические модели крупных производственных процессов (имитационное моделирование работы электростанций, химических цехов с большим числом одновременно протекающих процессов и др.). Именно с развитием вычислительных кластеров связаны комплексные решения задач большой размерности, которые до недавнего времени считались безнадежными в практическом плане. При этом значительную роль играют вычисления с повышенной разрядностью. Так, при решении, например, задач, связанных с моделированием химических реакций, часто возникают проблемы жесткости системы дифференциальных уравнений. Несмотря на то, что «жесткую» задачу удается эффективно перенести на вычислительный кластер, точность ее решения вызывает большие сомнения, если использовать стандартные представления чисел на периферийных компьютерах. Таким образом, при решении блочных задач проблемы размерности и точности могут быть взаимосвязаны: чем больше блоков участвует в моделировании, тем априори выше требования к точности вычислений с вещественными числами.

В нашей работе мы стремимся рассматривать структуру вычислений как адаптируемый ориентированный граф с параметричностями дуг в виде фреймов знаний, включающих в себя количественную и качественную информацию о передаче информации из блока в блок. Адаптация графа подразумевает смену структуры и/или уточнение фреймов с последующим пересчетом схемы кластерного расчетного модуля. В основе оптимизации расчетов – два конкурирующих критерия структурной сложности.

Итак, имеется система S, структура которой представима в виде сильно связного графа. Рассмотрим ряд модификаций классических матричных представлений сильно связного ориентированного графа без петель $G = (V, D, \Gamma)$, где $V = \{v_1, v_2, ..., v_n\}$ – множество вершин графа; $D = \{d_1, d_2, ..., d_m\}$ – множество дуг с параметричностями $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. В данной работе будем считать, что дуги $d_k(i,j) = (v_i \to v_j), k = 1..m; i, j \in \{1..n\}$ имеют вещественные параметричности $\gamma_k(i,j) \in R_1^+$. В частности, это могут быть целые числа, как в работах Г.М. Островского [1]. Нумерация дуг подчиняется правилу Ξ, о котором будет сказано ниже. Граф G считается конденсированным по правилу Ψ так, что в нем изначально нет транзитивных вершин:

$$\nexists i, j, l, i_1, i_2 \in \{1..n\}, k, k_1, k_2, k_3, k_4 \in \{1..m\} : \left(\exists \left(d_{k_1}(i,j), d_{k_2}(j,l)\right), \exists \left(d_{k_3}(i_1,j), d_{k_4}(j,i_2)\right)\right).$$

1. Взвешенная матрица смежности

$$\mathbf{X} = (x_{ij})_{n,n};\ x_{ij} = \begin{cases} \exists d_k \in D \Rightarrow x_{ij} = \gamma_k(i,j), k \in \{1..m\}, \\ \nexists d_k \in D \Rightarrow 0, k \in \{1..m\}. \end{cases}$$

2. Взвешенная матрица инцидентности

$$\mathbf{B} = (b_{ij})_{n,m};\ b_{ij} = \begin{cases} \exists d_j(l,k), i \neq l, i \neq k \Rightarrow 0; k, l \in \{1..m\}, \\ \exists d_j(i,k) \Rightarrow x_{ik}; k \in \{1..m\}, \\ \exists d_j(k,i) \Rightarrow 2x_{ki}; k \in \{1..m\}. \end{cases}$$

3. Взвешенная матрица контуров

$$\mathbf{C} = (c_{ij})_{r,m};\ c_{ij} = \begin{cases} \underline{c}_{ij} = 1 \Rightarrow x_{uk} \sum_{l=1}^{r} \underline{c}_{lj}; \exists d_j(u,k), j \in \{1..m\}, \\ \underline{c}_{ij} = 0 \Rightarrow 0. \end{cases}$$

Здесь $\underline{\mathbf{C}}$ – матрица элементарных контуров:

$$\underline{\mathbf{C}} = (\underline{c}_{ij})_{r,m};\ \underline{c}_{ij} = \begin{cases} \exists \kappa_i = \left(d_{k_1}(i_1, j_1), d_{k_2}(i_2, j_2), ..., d_{k_{t_i}}(i_{t_i}, i_1)\right) \Rightarrow 1; \\ \nexists \kappa_i \Rightarrow 0, \end{cases}$$

где r – число контуров графа G; κ_i – i–й элементарный контур.

Правило Ξ нумерации дуг устанавливает в качестве более приоритетных дуги, входящие в большее число элементарных контуров:

$$\forall\, d_{k_1}, d_{k_2} \in D : \sum_{i=1}^{r} \underline{c}_{ik_1} \geq \sum_{i=1}^{r} \underline{c}_{ik_2} \Rightarrow k_1 < k_2 .$$

В рассмотрение вводится понятие «матрица сложности» графа G:

$$\mathbf{W}(G) = \left(\mathbf{XBC}^{\mathrm{T}}\right)\left(\mathbf{XBC}^{\mathrm{T}}\right)^{\mathrm{T}} = \left(w_{ij}\right)_{n,n} \tag{1}$$

Аксиома абстрактной сложности: граф G_1 сложнее графа G_2, если $\|\mathbf{W}(G_1)\|_F > \|\mathbf{W}(G_2)\|_F$.

Критерий предпочтительности используется нами в анализе структурной сложности графа G и вычисляется рекурсивно:

$$P(G) \Leftarrow \frac{\left\| W\left(G\middle|\gamma_1^* \Leftarrow \gamma_1^* + \delta\right)\right\|_F - \left\| W\left(G\middle|\gamma_1^* \Leftarrow \gamma_1^* - \delta\right)\right\|_F}{2\delta} \times \frac{1}{\gamma_1^*} \times$$
$$\times \left[1 + \sum_{i=1}^{\beta_1^*} P\left(Dec\left(G \Leftarrow \Psi\left(X, D \setminus \left\{d_1^*\right\}, \Gamma \setminus \left\{\gamma_1^*\right\}\right)\right)[i]\right)\right] \to \frac{}{d_1^*} \to \max, \tag{2}$$

где d_1^* – первая дуга в иерархии разрывов при итерационном расчете системы S; γ_1^* – ее параметричность; $G\left|\gamma_1^* \Leftarrow \gamma_1^* \pm \delta\right.$ – граф G, в котором параметричность дуги d_1^* получает приращение, равное $\pm\delta$; δ – параметр численного дифференцирования нормы Фробениуса матрицы $\mathbf{W}$.

Рекурсивность критерия (2) подразумевает, что после разрыва дуги d_1^* сначала проводится конденсация графа G без дуги d_1^*, а затем структурная декомпозиция: формально говоря, оператор $Dec(G)$ выявляет массив из β –го числа сильно связных подсистем (запись β_1^* означает, что число подсистем обусловлено разрывом дуги d_1^*; выражение $Dec(G)[i]$ формально означает «i–я сильно связная подсистема»), а затем к каждой i–й подсистеме применяется та же процедура вычисления критерия предпочтительности, что и к исходному графу. Для нахождения массива сильно связных подсистем оператор структурной декомпозиции использует матрицу достижимости $U = [sign(X)]^n$, в которой каждой отдельно взятой подсистеме соответствует набор одинаковых строк. Для исходного графа характерно равенство $\sum_{i=1}^{n}\sum_{j=1}^{n} sign(u_{ij}) = n(n-1)$. Нумерация подсистем соответствует нумерации подматриц, образующих квазидиагональ матрицы смежности после установки в нуль элемента $x_{ij} \Leftrightarrow d_1^*$ в ее квазитреугольной форме, если она существует. Возможна ситуация, когда после разрыва дуги d_1^* граф продолжает оставаться сильно связным. В этом случае $\beta_1^* = 1$.

На рис. 1 показано, как организуются параллельные вычисления с использованием критерия предпочтительности. Без ограничения общности будем полагать, что граф системы после разрыва дуги d_1^* является совокупностью из двух гамаков. Через дуги, обозначенные 2 и 3, проходит наибольшее число контуров, и предположительно именно они наиболее

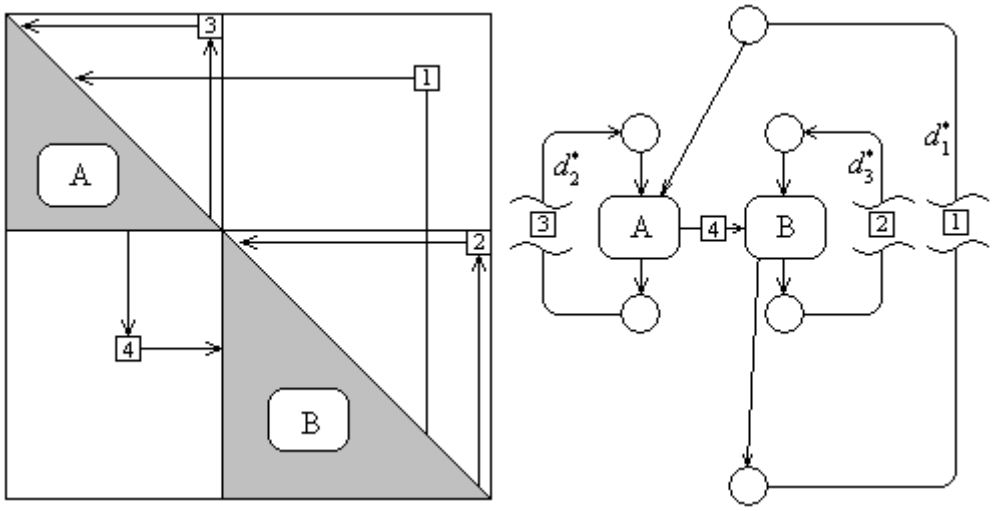


Рис. 1. Иллюстрация к параллельному расчету системы по критерию

предпочтительны к разрыву; производная нормы матрицы сложности по параметричности этих дуг больше производной для других дуг (производные всегда неотрицательные). Дуга d_1^* – первая дуга в иерархии разрывов, именно с разрыва этой дуги начинается итерационный расчет всей системы. Разрыв предполагает, что переменным, сопоставленным с этой дугой, производится задание начальных приближений с последующим уточнением, например методом Ньютона–Рафсона. Расчет подсистем A и B производится параллельно. Переменные, сопоставленные с дугой 4, находятся в критической секции, и доступ к ним можно регулировать (тривиальный случай – общая область памяти между двумя процессами). Внутри цикла по переменным разрыва дуги d_1^* находятся еще два итерационных цикла; после их параллельного прохождения в подсистеме B формируются уточненные значения переменных, сопоставленных дуге d_1^*. Разность заданных и полученных значений является, по сути дела, левой частью СНУ, решаемой в разрыве 1.

Следует отметить, что шкалирование сильно связных систем по критерию предпочтительности более конструктивно, нежели по норме матрицы сложности.

Рассмотрим другой критерий структурной сложности, по которому можно производить итерационный расчет больших систем. В работе [2] впервые появился термин «пролонгатор»; сам критерий претерпел с того времени значительные изменения [3–5], но название мы решили сохранить. В настоящее время он подкреплен аксиоматикой и выводом операторных уравнений, позволяющих расширить применение пролонгатора к фреймовым параметричностям. Пока же для сопоставления с критерием предпочтительности мы оставим параметричности вещественными:

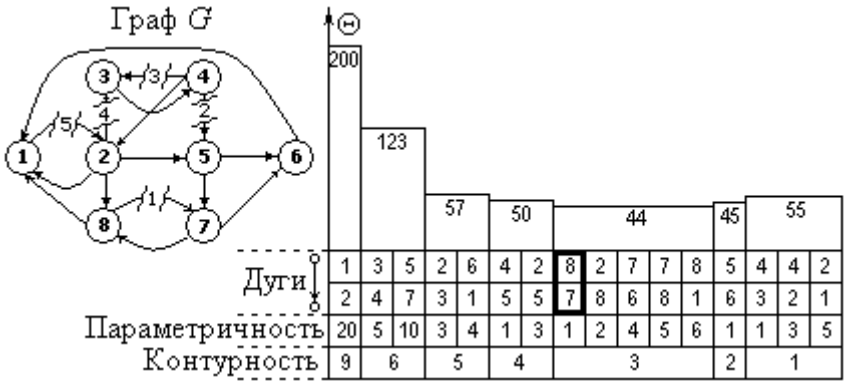


Рис. 2. Иллюстрация к вычислению пролонгатора

$$\theta(G) \Leftarrow \gamma_1^* \left[1 + \sum_{i=1}^{\beta_1^*} \theta\left(Dec\left(G \Leftarrow \Psi\left(X, D \setminus \{d_1^*\}, \Gamma \setminus \{\gamma_1^*\}\right)\right)[i]\right) \right] \xrightarrow[d_1^*]{} \min . \qquad (3)$$

На рис. 2 приведен пример графа из 8 вершин и показаны оптимальные разрывы, найденные в результате минимизации пролонгатора (номера разрывов). На этом же рисунке приведена диаграмма, иллюстрирующая вычисление пролонгатора на верхнем уровне рекурсии. Дуга $8 \to 7$ обведена жирной линией: именно с разрыва этой дуги начинается итерационный расчет системы, представленной графом G.

На рис. 2 видно, что наиболее предпочтительной к разрыву является дуга $1 \to 2$ – она разрывает сразу 9 контуров, но имеет большую параметричность, слишком большую, чтобы было выгодно начинать с ее разрыва итерационный расчет. Как видно из этого рисунка, правило Ξ нумерации дуг дополняется необходимостью упорядочения дуг в порядке возрастания параметричностей внутри каждой подматрицы матрицы элементарных контуров. На шкале сложности показана выпуклая оболочка значений пролонгатора. От особенностей ее построения во многом зависит вычислительная сложность алгоритма вычисления пролонгатора. Выведен ряд особенностей поведения функции $\theta(G)$, позволяющих находить d_1^* за время, которое зависит от размерности системы почти линейно.

После удаления из графа дуги d_1^* проводится конденсация графа, иллюстрация к которой приводится на рис. 3. Показано, что транзитивные дуги заменяются обобщенной дугой с минимальной параметричностью

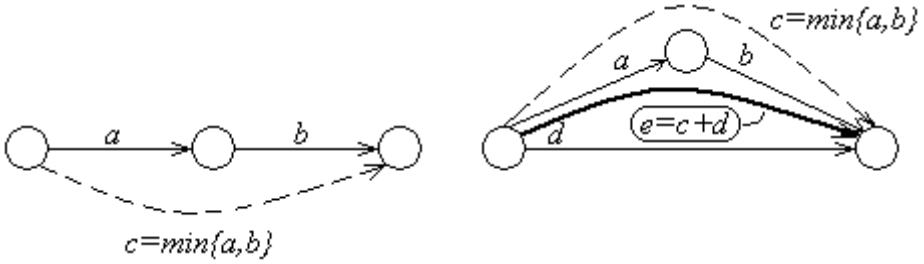


Рис. 3. Иллюстрация к конденсации графа

(показана пунктиром). Этот процесс мы называем стягиванием. После стягивания могут появиться «параллельные» дуги, т.е. возникает тенденция перехода к рассмотрению мультиграфа. Но мы отказываемся от этого и заменяем две «параллельные» дуги на одну обобщенную с суммарной параметричностью $e = \min\{a,b\} + d$.

В таком виде конденсация графа отвечает цели анализа (построения оптимального расчетного модуля с возможностью параллельного процессирования). Другие цели анализа порождают не только иной алгоритм конденсации, но и другие принципы пролонгирования сложности. Так, например, иначе устроен пролонгатор для оценки вычислительной сложности циклических алгоритмов с шаговыми циклами. Разработан также пролонгатор для оценки вычислительной сложности итерационных алгоритмов линейной алгебры. Данные оценки принципиально согласуются с выводами, полученными в теории вычислений совсем другим путем. Оценивание сложности алгоритмов может проводиться как с учетом распараллеливания, так и без него.

В настоящее время ведется работа по отысканию Парето-оптимальных областей с использованием критерия предпочтительности (2) и пролонгатора (3).

Вычисление критериев (2), (3) оформлено в виде программы на языке Visual C++, имеющей дружественный, графический интерфейс. Протокол осуществления разрывов записывается в HTML-файл, удобный для просмотра.

Литература

1. Островский Г.М., Волин Ю.М. Моделирование сложных химико-технологических схем. – М.: Химия, 1975. – 311 с.
2. Толстых С.С., Чаузов А.Г. Метод структурного анализа больших систем //Деп. в ВИНИТИ редколлегией журнала «Известия АН СССР. Техническая кибернетика». – 1985. – № 6581–85. – 36 с.
3. Толстых С.С., Толстых Т.Н. Эвристический алгоритм построения функций структурной сложности //Вестник ТГТУ. Тамбов, 1996. Т.1, № 1–2.
4. Толстых С.С. Матричный критерий сложности замкнутых систем //Вестник ТГТУ. Тамбов, 1998. Т. 4, № 2–3.
5. Толстых Т.Н., Толстых С.С. Структурный анализ в региональной экономике. – Воронеж: «Истоки», 1998. – 68 с.

ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА ЗАГРУЗКИ В ЗАДАЧЕ МОДЕЛИРОВАНИЯ ДИНАМИКИ ПРОТОПЛАНЕТНОГО ДИСКА

А.В. Снытников

Институт вычислительной математики и математической геофизики СО РАН

Протопланетный диск описывается системой уравнений звездной динамики [1], состоящей из уравнения Власова и уравнения Пуассона:

$$\begin{cases} \dfrac{\partial f}{\partial t} + \vec{v}\nabla f - \vec{a}\dfrac{\partial f}{\partial \vec{v}} = 0; \\ \dfrac{1}{r}\dfrac{\partial}{\partial r}\left(r\dfrac{\partial \Phi_2}{\partial r}\right) + \dfrac{1}{r^2}\left(\dfrac{\partial^2 \Phi_2}{\partial \varphi^2}\right) + \dfrac{\partial^2 \Phi_2}{\partial z^2} = 4\pi G\rho, \end{cases}$$

где f – зависящая от времени t одночастичная функция распределения по координате r и скорости $\vec{v}$; $\vec{a}$ – ускорение частиц единичной массы; F – гравитационный потенциал.

Для решения выбрана цилиндрическая система координат, которая наилучшим образом соответствует симметрии изучаемого объекта. Уравнение Власова решается методом частиц в ячейках [2,3]. При аппроксимации уравнения Пуассона получается система линейных алгебраических уравнений. Для решения этой СЛАУ используется комбинированный метод, состоящий из преобразования Фурье и итерационной процедуры последовательной верхней релаксации. Это позволяет сочетать высокую производительность прямых методов и нечувствительность к ошибкам округления, присущую итерационным методам. Структура метода показана на рис. 1. Более подробное описание можно найти в работе [3].

После преобразования Фурье получается набор двумерных систем линейных алгебраических уравнений, описывающих отдельные гармоники потенциала. Первоначально эти уравнения равномерно распределяются по процессорам. Скорость сходимости метода ПВР зависит от диагонального преобладания матрицы системы, которая имеет следующий вид:

$$A(k) = \left\{ \frac{1}{h_R^2}\frac{r_{i-3/2}}{r_{i-1/2}}, \ldots, \frac{1}{h_Z^2}, 2\left[\frac{1}{h_Z^2} + \frac{1}{h_R^2} + \frac{2\sin^2\dfrac{\pi k}{N_\varphi}}{h_\varphi^2 r_{i-1/2}^2} \right], \frac{1}{h_Z^2}, \ldots, \frac{1}{h_R^2}\frac{r_{i+1/2}}{r_{i-1/2}} \right\},$$

$$0 \le k \le N_\varphi .$$

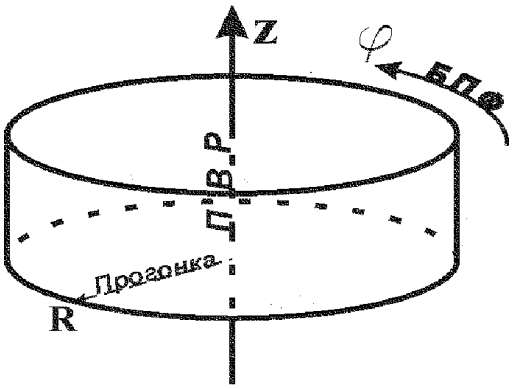


Рис. 1

Рис. 1 Структура итерационного метода

Из формулы видно, что для коротковолновых гармоник *(k~Nf)* есть выраженное диагональное преобладание, но в случае небольших k диагональное преобладание слабое. Таким образом, скорость сходимости также будет неодинакова, и необходимо найти оптимальное распределение гармоник по процессорам.

Число итераций N_I, требуемое для достижения сходимости на некотором временном шаге, пропорционально скорости изменения гармоники потенциала и, таким образом, не может быть предсказано заранее:

$$\Phi(r,\varphi,z,t) = \sum_{k=0}^{N_\varphi} \hat{\hat{\Phi}}(r,z,t)\exp[ik\varphi];$$

$$N_I(k) \propto \frac{\partial}{\partial t}\hat{\hat{\Phi}}(r,z,t).$$

Отсюда следует, что необходима динамическая балансировка загрузки процессоров.

Под загрузкой процессора подразумевается суммарное число итераций на текущем шаге по всем гармоникам потенциала, принадлежащим

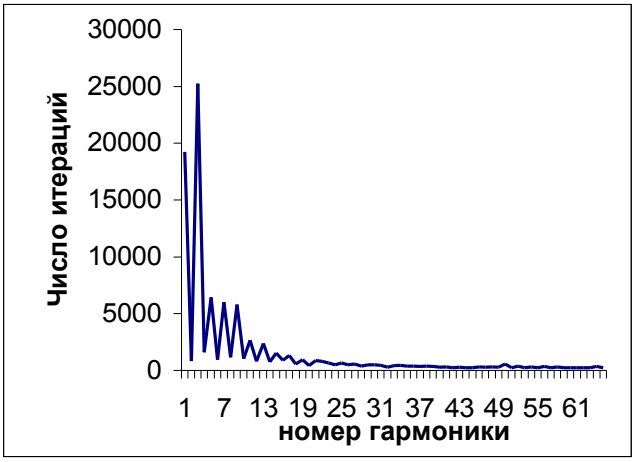


Рис. 2 Зависимость числа итераций от номера гармоники

данному процессору. На рис. 2 показана типичная зависимость числа итераций от номера гармоники. Из физических соображений следует, что может быть только один процессор, загрузка которого заметно превышает загрузку любого другого процессора.

Процессор считается перегруженным, если его загрузка превышает порог. Некоторые гармоники должны быть перемещены с перегруженного процессора. Вначале гармоника, требующая наименьшего числа итераций, пересылается на наименее загруженный процессор. Этот процесс повторяется, пока балансируемый процессор остается перегруженным или пока на нем не останется только одна гармоника.

Существенной проблемой являются длинноволновые гармоники, для расчета которых необходимо намного больше итераций, чем для расчета всех остальных гармоник. Распределение загрузки процессоров в этом случае показано на рис. 3 (для 8 процессоров).

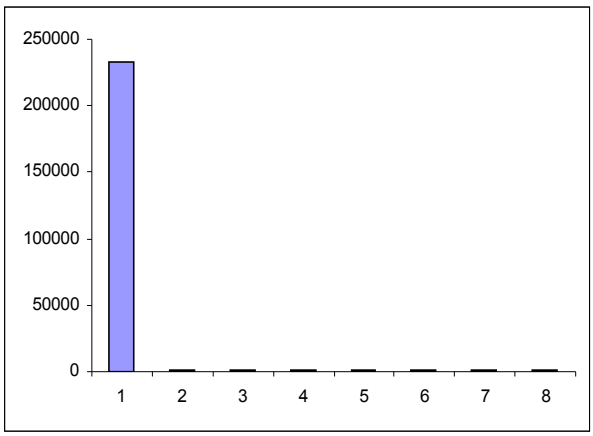


Рис. 3 Распределение загрузки процессоров

Процессор, содержащий одну такую гармонику, остается наиболее загруженным. В таких случаях производится разделение гармоники между процессорами.

Разработанный алгоритм балансировки обеспечивает почти равномерное распределение загрузки между процессорами. На рис. 4 показано отношение суммарной загрузки всей системы W к максимальной загрузке отдельного процессора:

$$W = \frac{N_{total}}{N_{\max}}.$$

В идеале это отношение равно числу процессоров, тогда загрузка распределена совершенно равномерно. Измерения проводились в течение 1000 временных шагов, в расчете использовано 8 процессоров, размер сетки 1000x512x512 узлов.

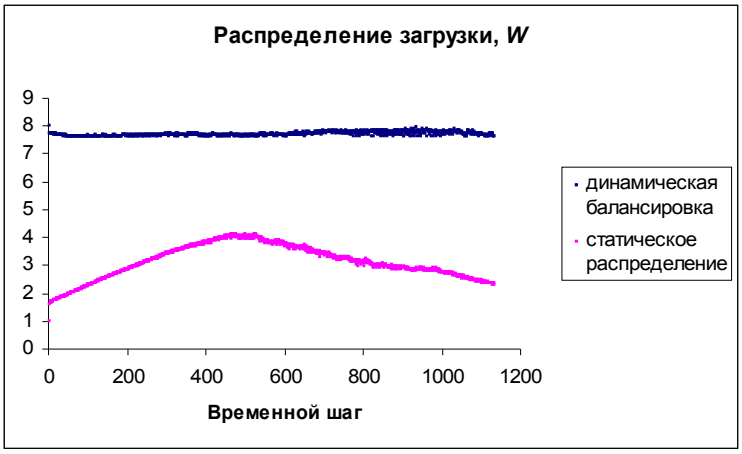


Рис. 4

Видно, что в результате использования динамической балансировки распределение загрузки оказывается близким к идеальному. Это исключительно важно, так как цель работы заключается в проведении больших расчетов за приемлемое время.

Литература

1. Bertin G. Dynamics of Galaxies. – Cambridge University Press, 2000.
2. Хокни Р., Иствуд Дж. Численное моделирование методом частиц. – М.: Мир, 1987.
3. Вшивков В.А., Кукшева Э.А., Никитин С.А., и др. О параллельной реализации численной модели физики гравитирующих систем // Автометрия. – 2001. – Т. 39. – С. 115–123.

ПАРАЛЛЕЛЬНАЯ ГЕНЕРАЦИЯ СОЧЕТАНИЙ И ПЕРЕСТАНОВОК

Н.Е. Тимошевская

Томский государственный университет

Предлагается метод параллельной генерации перестановок и сочетаний, ориентированный на использование многопроцессорных систем. В основе метода лежит возможность нумерации перечисляемых объектов так, что по номеру можно легко восстановить сопоставленный ему объект. Построенная таким образом нумерация позволяет разбить все множество объектов на классы, мощности которых отвечают производительности процессоров в системе, и генерировать объекты различных классов независимо и параллельно на соответствующих процессорах.

1. Введение

При решении многих задач возникает необходимость перебора или упорядоченного перечисления комбинаторных объектов определенного типа – подмножеств заданного множества, перестановок его элементов, размещений, подстановок, разбиений множеств и других подобных объектов. Порождение объектов указанных типов, как правило, трудоемко по времени, так как в реальных задачах их число очень велико. Применение параллельных вычислений позволяет сократить время перебора.

2. Генерация сочетаний

Здесь под сочетанием понимается всякое упорядоченное по возрастанию k-элементное подмножество n-элементного множества $N = \{1,2,...,n\}$, т.е. любой такой вектор $(b_1, b_2, ..., b_k)$ длины $k \le n$ с компонентами в N, называемый также сочетанием из n по k, в котором $b_i < b_{i+1}$ для $i = 1,...,k-1$, а под генерацией сочетаний – перечисление всех таких векторов. Известны алгоритмы, в которых сочетания порождаются в лексикографическом порядке [1], т.е. так, что сочетание $(b_1, b_2, ..., b_k)$ предшествует сочетанию $(a_1, a_2, ..., a_k)$, если для некоторого $i, 1 \le i \le k$, имеет место $b_1 = a_1, b_1 = a_1, ..., b_i = a_i, u\ b_{i+1} < a_{i+1}$.

Например, все сочетания из 5 по 3 перечисляются в следующем порядке: (1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5), (3, 4, 5).

С целью распараллеливания генерации сочетаний введем понятие индекса сочетания и покажем, как сочетание строится по его индексу. Индексом сочетания назовем порядковый номер I этого сочетания в последовательности всех сочетаний, расположенных в лексикографическом порядке, $1 \le I \le C_n^k$. Каждому индексу I из указанного диапазона соответствует одно сочетание и наоборот.

Рассмотрим теперь, как сочетание $(b_1, b_2, ..., b_k)$ строится по его номеру I. Пусть $1 \le t \le k$ и $0 = b_0 < b_1 < b_2 < ... < b_{t-1}$. Для любого j в $\{b_{t-1} + 1, b_{t-1} + 2, ..., n - k + t\}$, через s_j^t обозначим число всех сочетаний с префиксами $(b_1, b_2, ..., b_{t-1}, r)$, где $r \in \{b_{t-1} + 1, b_{t-1} + 2, ..., j - 1\}$. В каждом из этих сочетаний недостающие $k - t$ элементов берутся из $n - r$ элементов $r + 1, ..., n$, поэтому

$$s_j^t = \sum_{r=b_{t-1}+1}^{j-1} C_{n-r}^{k-t}.$$

Положим также $s^t = \sum_{i=1}^{t-1} s_{b_i}^i$. Это есть число всех сочетаний с префиксами $(b_1, b_2, ..., b_{i-1}, a_i)$ для $i \le t - 1$ и $a_i < b_i$, т.е. всех сочетаний, предшествующих сочетаниям с префиксом $(b_1, b_2, ..., b_{t-1})$. Отметим, что величины s_j^t и s_t не убывают с ростом j и t соответственно.

Например, для сочетаний из 5 по 3 при $t = 1$: s_1^1=0, s_2^1=6, s_3^1=9; при $t = 2$ и $b_1 = 1$: s_2^2=0, s_3^2=3, s_4^2=5; при $b_1 = 2$: s_3^2=0, s_4^2=2; при $t = 3$ и (b_1, b_2) = (1, 2): s_3^3=0, s_4^3=1, s_5^3=2. Кроме того, s^1=0; при $t = 2$ и $b_1 = 1$: s^2=0, при $b_1 = 2$: s^2=6, при $b_1 = 3$: s^2=9; при $t = 3$ и (b_1, b_2) = (1, 2): s^3=0, при (b_1, b_2) = (1, 4): s^3 =5, при (b_1, b_2) = (2, 3): s^3=6.

Теорема 1. Пусть $1 \le t \le k$ и $(b_1, b_2, ..., b_{t-1})$ есть префикс сочетания с индексом I; тогда t-й элемент b_t этого сочетания равен наименьшему из таких j, что $b_{t-1} < j < n - k + t$ и $I \le s^t + s_{j+1}^t$, если такие j существуют, и $b_t = n - k + t$ в противном случае.

Пример. Следуя методу данной теоремы, построим сочетание из 5 по 3 с индексом I=9.

$t = 1$: s^t=0; j=1, $s_{J+1}^t = s_2^1$=6, $s^t + s_{J+1}^t$=6< I; j=2, s_3^1=9, $s^t + s_{J+1}^t$=9≥ I; следовательно, b_1=2.

$t = 2$: $s^t = s_2^1$=6; j=3, $s_{J+1}^t = s_4^2$=2, $s^t + s_{J+1}^t$=8< I; j=4=$n - k + t$; следовательно, b_2=4.

$t = 3$: не существует j для 4=b_{t-1}< j <$n - k + t$=5, поэтому $b_3 = n - k + t$=5.

В результате получаем сочетание (2, 4, 5).

Данный метод вычисления сочетания из n по k по его индексу I можно выразить следующим алгоритмом.

Алгоритм 1

1. $b_0=0$, $s=0$, $t=1$.

2. Если $t \le k$, то п.3, иначе п.8.

3. $j = b_{t-1}+1$.

4. Если $j<n-k+t$ и $s+C_{n-j}^{k-t}<I$, то п.5, иначе п.6.

5. $s = s+C_{n-j}^{k-t}$; $j=j+1$; п.4.

6. $b_t=j$.

7. $t=t+1$; п.2.

8. $(b_1,b_2,...,b_k)$ – искомое сочетание.

Реализация алгоритма 1 предполагает использование «больших» чисел (целых чисел, двоичное представление которых требует большего числа разрядов, чем в стандартных типах) для задания индекса и подсчета числовых значений C_n^k. Вместе с тем, опираясь на него, можно предложить следующий алгоритм параллельной генерации сочетаний.

Пусть требуется построить все сочетания из n по k на однородной многопроцессорной вычислительной системе, состоящей из p одинаковых процессоров с общей или распределенной памятью. Максимальную загруженность процессоров можно обеспечить, разделив последовательность всех сочетаний на блоки по $s = [C_n^k / p]$ элементов в каждом. Блок с номером m начинается с сочетания, соответствующего индексу $I = s(m-1)+1$, и генерация сочетаний в нем возлагается на m-й процессор. Если число всех сочетаний не кратно p, то процессор, генерирующий сочетания последнего блока, строит также $C_n^k \bmod p$ оставшихся сочетаний, число которых меньше p. При выполнении этого алгоритма отсутствует взаимодействие между процессорами и все процессоры одинаково загружены, что позволяет добиться ускорения, близкого к p. В случае использования неоднородной системы равномощные блоки сочетаний генерируются не за одинаковое время, что приводит к простаиванию процессоров и падению эффективности алгоритма. В этом случае все множество сочетаний следует разбивать на блоки, мощности которых отвечают производительности процессоров в системе.

3. Генерация перестановок

Требуется построить все перестановки $(a_1,a_2,...,a_n)$ из элементов множества $N=\{1,2,...,n\}$, т.е. перестановки степени n. Известны алгоритмы, перечисляющие все перестановки в лексикографическом порядке, начиная с перестановки $(1,2,...,n)$ [2], т.е. так, что перестановка $(a_1,a_2,...,a_n)$ предшествует перестановке $(b_1,b_2,...,b_n)$, если для некоторого $i, 1 \le i \le k$, имеет место $b_1 = a_1, b_1 = a_1, ..., b_i = a_i, u\ b_{i+1} < a_{i+1}$.

Например, перестановки степени 3 перечисляются в следующем порядке: 123, 132, 213, 231, 312, 321.

Назовем индексом перестановки ее порядковый номер I в последовательности всех перестановок степени *n*, расположенных в лексикографическом порядке, $1 \le I \le n!$. Так перестановка 123 имеет индекс 1, перестановка 132 – индекс 2, перестановка 321 – индекс 6. Покажем, каким образом перестановку можно вычислить по ее индексу.

Пусть $0 \le t \le n$ и $A = (a_1, a_2, ..., a_n)$ есть префикс перестановки с индексом I. Обозначим J_t число перестановок, предшествующих всем перестановкам с префиксом A, и положим $M_t = I - J_t$. Назовем M_t кратностью префикса A. По определению $M_0 = I$ и M_t есть число перестановок с префиксом A, индексы которых не превосходят I. Пусть $Y = \{y_1, ..., y_{n-t+1}\} = N - \{a_1, ..., a_{t-1}\}$ и $y_1 < ... < y_{n-t+1}$. Тогда по свойству перестановки $a_t \in Y$. Допустим, что $a_t = y_k$, $1 \le k \le n - t + 1$. Тогда $A = (a_1, a_2, ..., a_{t-1}, y_k)$ и $J_t = J_{t-1} + (k-1)(n-t)!$, поэтому $M_t = I - J_t = I - J_{t-1} - (k-1)(n-t)!$. Таким образом, зная индекс перестановки и любой ее префикс, можно вычислить кратность последнего.

Пример. Пусть n=3, I=4. Тогда

1) если $t = 0$ (A пусто), то M_0=4;

2) если $t = 1$ и A=2, то $Y = \{1,2,3\}$, $k = 2$ и $M_1 = M_0 - (k-1)(n-1)! = 2$;

3) если $t = 2$ и A=23, то $Y = \{1,3\}$, $k = 2$ и $M_2 = M_1 - (k-1)(n-2)! = 1$.

Теорема 2. Пусть $1 \le t \le n-1$, $(a_1, ..., a_{t-1})$ – префикс перестановки с индексом I и M_{t-1} – его кратность. Пусть также $\{y_1, ..., y_{n-t+1}\} = N - \{a_1, ..., a_{t-1}\}$ и $y_1 < ... < y_{n-t+1}$. Тогда t-й элемент a_t в перестановке с индексом I есть y_k для

$$k = \left[\frac{M_{t-1} + (n-t)! - 1}{(n-t)!} \right].$$

При известном префиксе $(a_1, ..., a_{n-1})$ перестановки с индексом I ее последний член находится как единственный элемент множества $N - \{a_1, ..., a_{n-1}\}$.

Пример. Построим перестановку (a_1, a_2, a_3) по индексу $I = 4$.

$t = 1$: $M_0 = 4$, $Y = \{1,2,3\}$, $k = 2$, следовательно, $a_1 = 2$.

$t = 2$: $M_1 = M_0 - (k-1)(n-1)! = 2$, $Y = \{1,3\}$, $k = 2$,

следовательно, $a_2 = 3$.

$t = 3$: $Y = \{1\}$, следовательно, $a_3 = 1$.

В результате получаем перестановку (2, 3, 1).

Алгоритм 2 (вычисление перестановки $(a_1, ..., a_n)$ по индексу I)

1. $t = 1$, $Y = \{y_1, ..., y_n\} = \{1,2,...,n\}$.

2. Если $t = n$, то п.5; иначе $k = \left[\frac{I + (n - t)!}{(n - t)!} \right]$ и

3. $a_t = v_k$, $Y = \{y_1, ..., y_{n-t}\} = Y - \{y_k\}$.

4. $I = I-(k-1)(n-t)!$, $t = T + 1$, перейти к п.2.

5. $a_n = v_1$.

Параллельная генерация перестановок осуществляется так же, как и сочетаний, а именно все множество перестановок делится на блоки, мощности которых отвечают производительности процессоров в системе; в каждом блоке по алгоритму 2 с помощью индекса вычисляется первая перестановка, начиная с которой один из процессоров системы генерирует перестановки всего блока в лексикографическом порядке. Как и в случае сочетаний, этот параллельный алгоритм обладает высокой эффективностью благодаря равномерному распределению вычислений и отсутствию обменов данными между процессорами.

Литература

1. Липский В. Комбинаторика для программистов. – М.: Мир, 1988. – 200 с.
2. Дейкстра Э. Дисциплина программирования / Под ред. Э.З. Любимского. – М.: Мир, 1978. – 280 с.

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ОДНОГО МЕТОДА РЕШЕНИЯ УРАВНЕНИЙ НАВЬЕ-СТОКСА

Н.Н. Богословский, А.В. Старченко

Томский государственный университет

Рассматривается двумерное нестационарное рециркуляционное ламинарное течение несжимаемой вязкой жидкости в прямоугольной полости (каверне) с верхней стенкой, движущейся со скоростью W_0 (рис. 1).

Теплофизические свойства жидкости постоянны. На стенках каверны скорость жидкости равна скорости ограничивающих течение поверхностей.

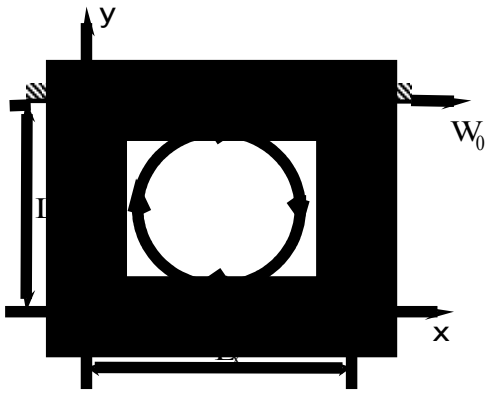


Рис.1 Область исследования

1. Математическая постановка задачи

Исследуемое течение описывается уравнениями Навье-Стокса [1], которые включают:

а) уравнение неразрывности

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0; \qquad (1)$$

б) уравнения движения

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} + \frac{\partial (uv)}{\partial y} = \vartheta \frac{\partial^2 u}{\partial x^2} + \vartheta \frac{\partial^2 u}{\partial y^2} - \frac{1}{\rho}\frac{\partial p}{\partial x}; \qquad (2)$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (v^2)}{\partial y} = \vartheta \frac{\partial^2 v}{\partial x^2} + \vartheta \frac{\partial^2 v}{\partial y^2} - \frac{1}{\rho}\frac{\partial p}{\partial y},$$

а также уравнениями состояния:

$$\rho = const; \vartheta = const \tag{3}$$

и граничными условиями (условиями прилипания):

$$\begin{aligned} &u\big|_{x=0} = v\big|_{x=0} = 0;\ u\big|_{x=L_x} = v\big|_{x=L_x} = 0; \\ &u\big|_{y=0} = v\big|_{y=0} = 0;\ v\big|_{y=L_y} = 0; u\big|_{y=L_y} = W_o. \end{aligned} \tag{4}$$

Здесь u,v – горизонтальная и вертикальная компоненты скорости соответственно; ϑ – коэффициент кинематической вязкости; ρ – плотность; p – давление.

В качестве начальных условий выбираются нулевые значения скорости жидкости во всей каверне.

2. Метод расчета

Система уравнений (1)–(3) с граничными условиями (4) решалась численно методом расщепления по физическим процессам. Решение задачи на шаге $t_j \le t \le t_{j+1}$ получим как результат последовательного решения двух более простых систем.

а) Система уравнений, описывающих перенос субстанций вдоль траекторий и молекулярный обмен при известном поле давления:

$$\frac{\tilde{u} - u^t}{\Delta\tau} + \frac{\partial}{\partial x}\left(u^t\tilde{u} - \vartheta\frac{\partial\tilde{u}}{\partial x} + \frac{p^t}{\rho}\right) + \frac{\partial}{\partial y}\left(v^t\tilde{u} - \vartheta\frac{\partial\tilde{u}}{\partial y}\right) = 0;$$

$$\frac{\tilde{v} - v^t}{\Delta\tau} + \frac{\partial}{\partial x}\left(u^t\tilde{v} - \vartheta\frac{\partial\tilde{v}}{\partial x}\right) + \frac{\partial}{\partial y}\left(v^t\tilde{v} - \vartheta\frac{\partial\tilde{v}}{\partial y} + \frac{p^t}{\rho}\right) = 0.$$

б) Система уравнений, обеспечивающих согласование поля давления и компонент скорости на слое t_{j+1}. Давление запишем в виде $p^{t+\Delta\tau} = p^t + p'$, $\Delta\tau = t_{j+1} - t_j$.

$$\frac{u^{t+\Delta\tau} - \tilde{u}}{\Delta\tau} = -\frac{1}{\rho}\frac{\partial p'}{\partial x};\quad \frac{v^{t+\Delta\tau} - \tilde{v}}{\Delta\tau} = -\frac{1}{\rho}\frac{\partial p'}{\partial y};\quad \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^{t+\Delta\tau} = 0.$$

Дискретизация, т.е. получение конечно-разностного аналога исходной системы дифференциальных уравнений, производится методом контрольного объема [2]. В данной работе использовались неравномерные шахматные сетки с узлами, сгущающимися к границе расчетной области. Построенная разностная схема является неявной и имеет первый порядок точности относительно $\Delta\tau$. При аппроксимации конвективных членов применяется противопотоковая схема, а для диффузионных – центрально-разностная. Полученная система сеточных уравнений является системой положительного типа с диагональным преобладанием.

Рассмотренный выше алгоритм можно представить в виде следующей последовательности шагов:

1) Задание начального поля давления p^0 и полагаем $p^t = p^0$, $t = 0$.

2) Задание начального поля скорости u^0, v^0.

3) Решение уравнений движения для получения $\tilde{u}, \tilde{v}$.

4) Решение уравнения для p' и расчет $p^{t+\Delta\tau}$ путем добавления p' к p^t.

5) Расчет $u^{t+\Delta\tau}, v^{t+\Delta\tau}$ с помощью формул для поправки скорости.

6) Полагаем $t = t + \Delta\tau$.

7) Если $t < T_{\max}$, то возврат к шагу 3.

Системы алгебраических уравнений для компонент скорости и поправки давления, получающиеся при аппроксимации исходной дифференциальной задачи, решаются с помощью явного метода Н.И. Булеева [3].

3. Параллельная реализация

На рис. 2 схематично представлено распределение данных между процессами. Нижняя часть выделенного круга представляет область, рассчитываемую 0-процессом, а верхняя – область, рассчитываемую 1-процессом. Кружками обозначены точки, в которых вычисляется давление, а штрихи на гранях конечных объемов соответствуют точкам вычисления горизонтальной и вертикальной компонент скорости. На разных этапах расчетов требовалась синхронизация вычислений между соседними процессами, т.е. обмен данными, расположенными в прилежащих к границам разрезов областях. Например, для расчета давления вблизи разрезов требуются четыре смежных значения (шаблон «крест»), три из которых рассчитываются и расположены в данном процессе, а четвертое, вычисленное соседним процессом, необходимо получить.

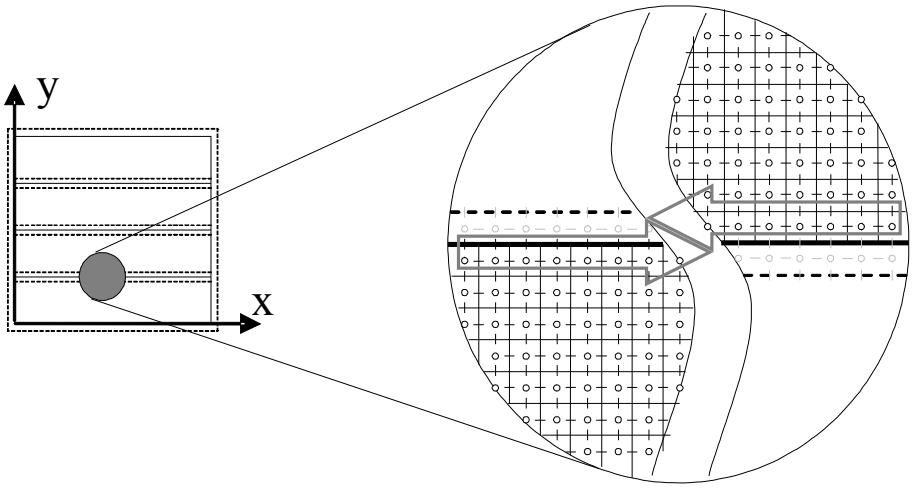


Рис. 2 Одномерная декомпозиция

По этой причине необходима пересылка значений последней строки 0-процесса 1-процессу, равно как и получение от него его первой строки. Для этой цели использовались операции блокирующей парной пересылки данных MPI (функция SendRecv). Аналогичным образом выглядит схема расчета и обмена значениями для компонент скорости u и v.

4. Результаты

Для проверки правильности работы вычислительной программы и адекватности рассматриваемой модели реальным процессам были проведены сравнения результатов расчетов с результатами расчетов, полученными другими авторами [1]. В правильности расчетов можно убедится, обратившись к рис. 3, где изображены профили *x*-компоненты скорости в среднем вертикальном сечении каверны с подвижной верхней стенкой. Точками представлены расчеты, полученные при помощи алгоритма SIMPLE [2]. На рис 4 показано ускорение вычислительного процесса, полученное при расчете на сетке 100x100 и 200x200. При расчетах на сетке 100x100 вплоть до использования 10 процессоров ускорение линейно растет, а затем начинает падать.

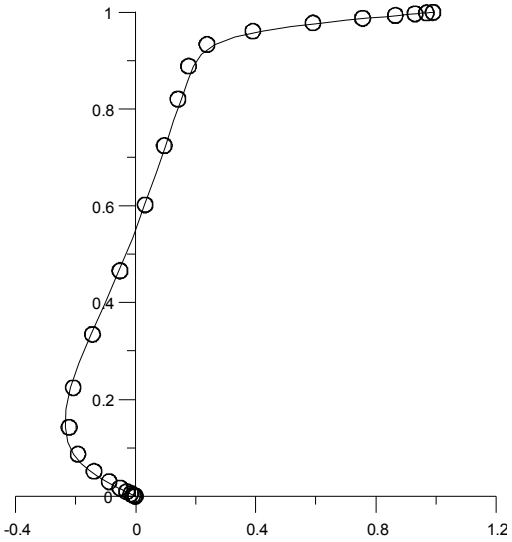


Рис. 3 Профили скорости

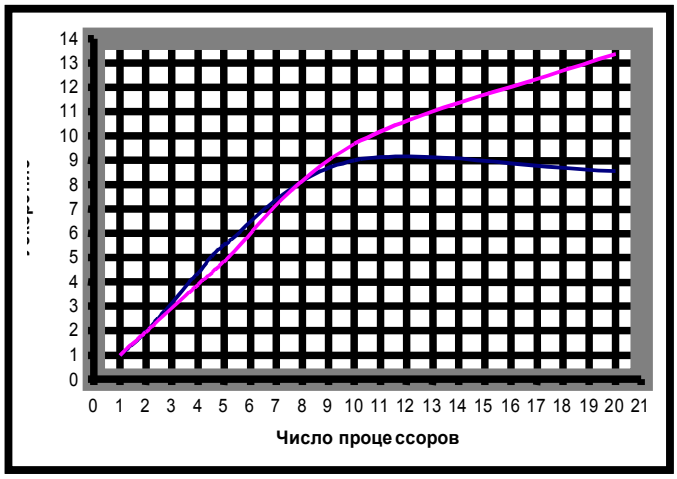


Рис.4 Ускорение параллельной программы

Это объясняется тем, что уже при десяти процессорах отношение объема пересылаемых данных к объему данных, для которых проводятся вычисления, равно 1/5. А так как операция пересылки выполняется на порядок медленнее арифметической операции, то соответственно при росте числа процессоров время расчета будет увеличиваться, а значит, и ускорение будет падать. При расчетах же на сетке 200x200 установлены большие значения ускорения. Максимальное ускорение, которого удалось достичь, равно 13.3 для 20 процессоров.

Литература

1. Лойцянский Л.Г. Механика жидкости и газа. – М.: Наука, 1987. – 840 с.
2. Патанкар С. Численные методы решения задач теплообмена и динамики жидкости. – М.: Энергоатомиздат, 1984. – 152 с.
3. Богословский Н.Н., Есаулов А.О. Параллельная реализация итерационных методов решения уравнения Пуассона // Высокопроизводительные параллельные вычисления на кластерных системах: Материалы 2-го Международного научно-практического семинара. – Нижний Новгород: Изд-во Ниж. ун-та, 2002. – С. 30–37.

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ О ВЫПОЛНИМОСТИ КНФ

С.С. Гисс, Н.Е.Тимошевская

Томский государственный университет

Задача о выполнимости КНФ формулируется следующим образом: для некоторой булевой функции $K(x_1, x_2, ..., x_m)$, представленной в виде конъюнктивной нормальной формы (КНФ), требуется установить, выполнима ли она, т.е. существует ли хотя бы один набор значений для ее переменных, на котором она обращается в единицу, и если выполнима, то указать некоторый из таких наборов. Данная задача является NP-полной, и для нее не найдено алгоритма с полиномиальной сложностью.

Поставленную задачу можно решать методом подстановки каждого из 2^m наборов значений переменных в $K(x_1, x_2, ..., x_n)$. Такой подход при большом числе переменных крайне непрактичен. В работе [1] для решения данной задачи предлагается метод сокращенного обхода в глубину бинарного дерева поиска. Корню дерева сопоставляется исходная КНФ, а остальным узлам – КНФ, получающиеся из соответствующих КНФ-родителей подстановкой вместо некоторой переменной x_j значений 0 или 1, а ребрам соответственно x_j или $\bar{x}_j$. В каждой вершине дерева выполняется проверка сопоставленной ей КНФ на принадлежность одному из полиномиальных подклассов U, Z, N, где U – класс КНФ, выполнимых на наборе из единиц; Z – класс КНФ, выполнимых на наборе, состоящем из нулей; N – класс невыполнимых КНФ, в которых содержится пара дизъюнктов вида x_i и $\bar{x}_i$. Если КНФ принадлежит классу N или тождественна константе нуль, то ветвление из соответствующей ей вершины не выполняется. Обход дерева завершается, если какая-либо из полученных КНФ попадает в класс U или Z, либо выполняется возврат в корень дерева. В первом случае исходная КНФ выполнима, во втором – нет.

Распараллеливание описанного алгоритма выполняется за счет параллельного обхода дерева [2]. На первом этапе управляющий процесс выполняет обход дерева в ширину, начиная от его корня, до тех пор, пока число вершин в очереди не достигнет числа параллельных процессов s. Вершины, попавшие в очередь, образуют множество корневых вершин поддеревьев, причем их родительские вершины уже пройдены. На втором этапе корневые вершины, точнее соответствующие им КНФ, распределяются по процессам, и каждый из них выполняет обход в глубину соответ-

ствующего поддерева с переданной ему КНФ в качестве корня. Если какой-либо из процессов объявляет КНФ выполнимой, то он строит выполняющий набор и сообщает об этом остальным процессам, после чего все процессы останавливаются. По окончании обхода своего поддерева процесс освобождается. Для его загрузки выбирается поддерево, обход которого в этот момент ведется одним из процессов, и в нем выделяется поддерево, которое передается для обхода освободившемуся процессу. Алгоритм заканчивает свою работу либо в случае, когда один из процессов объявляет КНФ выполнимой, либо при завершении обхода всех поддеревьев, что говорит о невыполнимости исходной КНФ. Выбор процесса, в дереве которого выделяется поддерево, и согласование передачи данных между процессами выполняются управляющим процессом. Описанный подход позволяет обеспечить равномерную загрузку процессоров системы и тем самым достичь высокой эффективности.

Литература

1. Паршина Н.А. Задача о выполнимости: некоторые полиномиальные классы КНФ // Кибернетика и системный анализ. – 1992. – № 3.
2. Тимошевская Н.Е. О параллельных алгоритмах обхода дерева // Вестник ТГУ. Приложение №6. – Томск: ТГУ, 2003.

ЭФФЕКТИВНОЕ ПРОГРАММИРОВАНИЕ УЗЛОВ МНОГОПРОЦЕССОРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

С.Е. Киреев

Институт вычислительной математики и математической геофизики СО РАН

Введение

Известно, что реальная производительность современных вычислительных систем, реализующих численные алгоритмы, часто составляет не более 10% от пиковой. Одной из причин такого падения производительности вычислительных систем является низкая реальная производительность отдельного узла.

Целью исследования является выработка рекомендаций по написанию эффективных программ для узлов многопроцессорных вычислительных систем. Под эффективностью программы будем понимать время ее исполнения при реализации заданного алгоритма.

В качестве среды для проведения эксперимента использовались узлы двух вычислительных кластеров: МВС-1000М (ИВМиМГ, Новосибирск): SMP-узел – 2 процессора Alpha 21264 / 833 МГц [1] и T-Forge32 (Т-системы, Москва) [2]: SUMO-узел – 2 процессора Opteron 244 / 1800 МГц.

С учетом архитектуры узла эффективность достигается за счет параллельной работы всех исполнительных устройств. Параллельную работу могут обеспечить быстрый доступ к данным и отсутствие зависимостей по данным и управлению. На одном процессоре это достигается за счет:

1) учета организации подсистемы памяти,
2) использования оптимизирующего компилятора,
3) использования специализированных библиотек.

1. Учет иерархии памяти

Классическая структура иерархии памяти показана на рис. 1. Для достижения наибольшей скорости исполнения прежде всего необходимо, чтобы активный набор данных располагался как можно ближе к процессору [3].

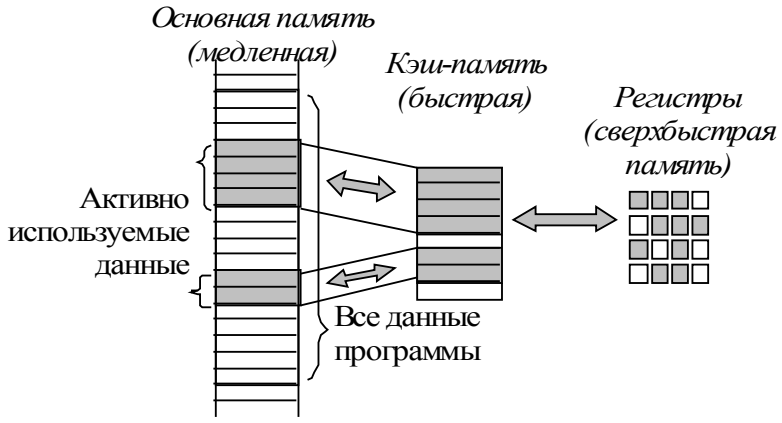


Рис. 1 Структура иерархии памяти

Рекомендация 1. Активный набор данных располагайте как можно выше по иерархии.

1.1. Использование регистров процессора

На самом верхнем уровне иерархии памяти расположены регистры процессора. Их число невелико, и они могут хранить только несколько значений, используемых процессором в данный момент.

Существуют стандарты, определяющие правила использования регистров процессоров в программах на различных операционных системах. Так, из общего числа регистров процессора часть может использоваться для хранения локальных переменных, часть – для хранения временных значений в процессе вычислений, часть – для передачи параметров в функции, часть – для возвращения результата и т.п. Зная число регистров, отводимых для хранения локальных переменных и параметров, мы определим максимальное число локальных переменных и число параметров подпрограмм, при которых программа будет работать наиболее быстро.

Рекомендация 2. Старайтесь не превышать указанные в табл. 1 количества локальных переменных и параметров подпрограмм.

Т а б л и ц а 1

Число регистров под локальные переменные и параметры подпрограмм

Процессор	Число регистров под локальные переменные		Число регистров под параметры подпрограмм	
	Целочисленные	Вещественные	Целочисленные	Вещественные
Alpha 21264	12	-	6	-
Opteron 244	15	16	6	8

1.2. Последовательный обход

Наиболее быстрым обходом данных является последовательный обход. Рассмотрим это на примере задачи перемножения двух квадратных матриц N×N. Если напрямую запрограммировать известную формулу $C_{ik} = \sum_{j=0}^{N-1} A_{ij} B_{jk}$, например, на языке Си, получим фрагмент программы, показанный в варианте 1 (рис. 2).

Вариант 1

```
for (i=0;i<N;i++)
 for (k=0;k<N;k++)
  for (j=0;j<N;j++)
   C[i][k]+=A[i][j]*B[j][k];
```

Вариант 2

```
for (i=0;i<N;i++)
 for (j=0;j<N;j++)
  for (k=0;k<N;k++)
   C[i][k]+=A[i][j]*B[j][k];
```

Рис.2 Варианты перемножения матриц

Зная, что массивы в языке Си хранятся по строкам, приходим к выводу, что элементы массива A перебираются последовательно, а элементы массива B – нет. Для ускорения работы программы нужно, чтобы элементы массивов перебирались последовательно. Для этого необходимо либо заранее транспонировать массив B, либо переставить циклы, как показано в варианте 2 (рис. 2).

Рекомендация 3. Старайтесь использовать последовательный обход данных (табл.2).

Таблица 2

Время перемножения матриц 1000×1000

Процессор	Вариант 1	Вариант 2	Ускорение
Alpha 21264	120,67 с	6,24 с	19,34
Opteron 244	16, 67 с	6,3 с	2,65

1.3. Учет ассоциативности кэш-памяти

Для вычислительных систем с ассоциативной кэш-памятью существует некоторый «плохой» шаг такой, что перебор данных в памяти с этим шагом выполняется намного дольше, чем с другими, отличающимися незначительно. Этот эффект называется «буксование» кэша.

Возникает он вследствие того, что для размещения элементов данных, расположенных в памяти с «плохим» шагом, кэш может предоставить только одну или несколько строк. Размер «плохого» шага вычисляется следующим образом: это числа, кратные размеру кэша (для ассоциативного), либо размеру кэша, деленного на число строк в множестве (для множественно-ассоциативного).

Рекомендация 4. При обходе данных избегайте идущих подряд обращений к элементам, расположенным с «плохим» шагом.

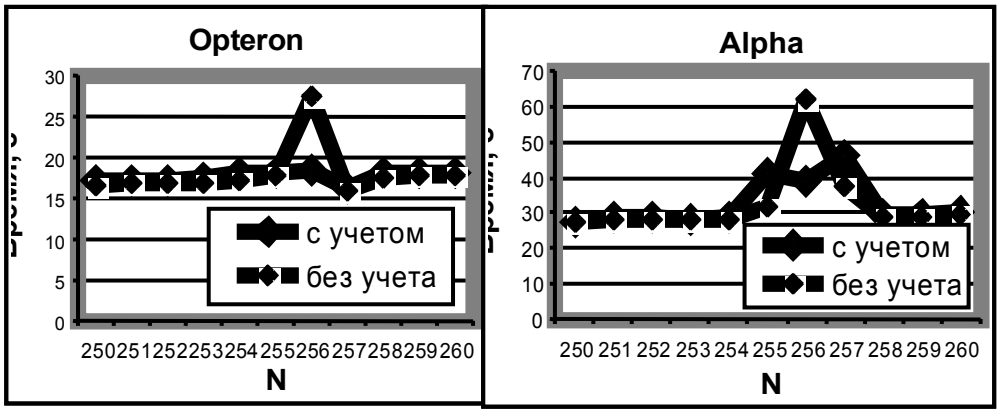


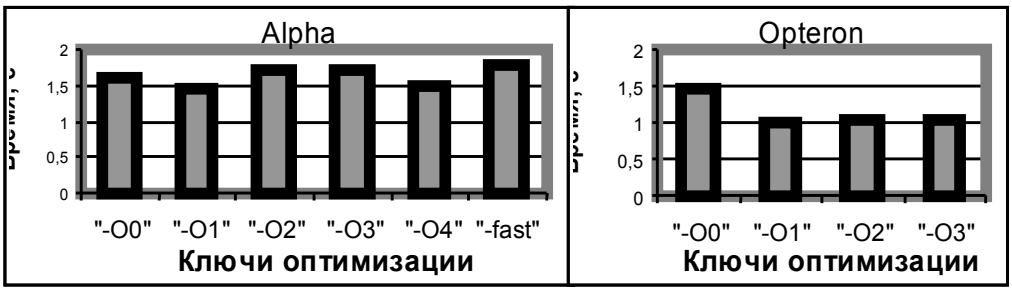
Рис.3 Результаты решения уравнения Пуассона методом Зейделя на сетке 256×256×N с учетом и без учета «плохого» шага

Эффект «буксования» кэша может возникнуть, например, при расчете элементов трехмерной сетки по семиточечной схеме (элемент и все соседи), если размер двумерного слоя равен «плохому» шагу (рис.3). Чтобы в таком случае избежать замедления, нужно изменить размер слоя, добавив после каждого слоя несколько неиспользуемых элементов с общей длиной не менее размера строки кэша.

2. Использование компилятора

Современные оптимизирующие компиляторы владеют большим числом техник оптимизации исполняемого кода, поэтому нельзя пренебрегать их возможностями при создании эффективных программ.

а) Вычисление числа Пи



б) Решение уравнения Пуассона методом Зейделя

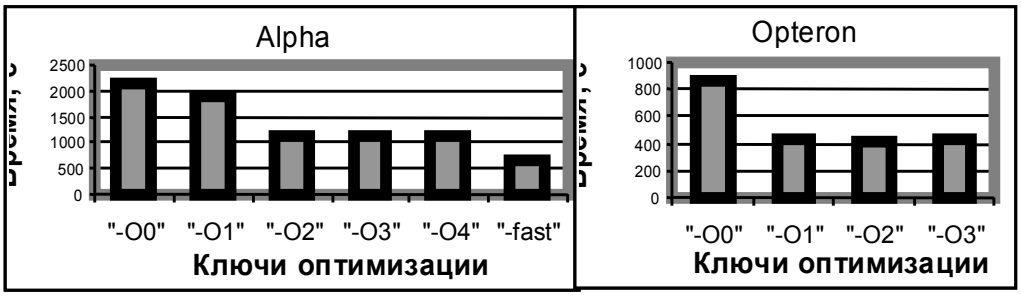


Рис. 4 Влияние оптимизационных ключей компилятора на время работы программ

С помощью опций компилятора можно включить использование той или иной техники или набора техник. Влияние техник оптимизации на эффективность получаемого кода во многом зависит от свойств конкретной программы (рис. 4).

Рекомендация 5. Для каждой конкретной программы выберите наилучший для нее набор ключей оптимизации.

3. Использование специализированных библиотек

Для многих предметных областей существуют библиотеки операций, оптимизированных под различные вычислительные системы. Они, как правило, реализованы очень эффективно, и их использование может сильно ускорить исполнение программы. Наиболее ярким примером таких библиотек являются BLAS и Lapack, которые содержат процедуры, реализующие многие операции линейной алгебры.

Рекомендация 6. По возможности используйте операции из эффективно реализованных специализированных библиотек (табл. 3).

Таблица 3

Время перемножения матриц 1000×1000

Процессор	Наилучший вариант, написанный вручную	Процедура из библиотеки BLAS	Ускорение
Alpha 21264	6,24 с	1,41 с	4,4
Opteron 244	6,3 с	0,64 с	9,8

Опыт пользователей показал, что использование данных рекомендаций позволяет в среднем на порядок увеличить производительность узла на задачах математической физики.

Литература

1. http://www2.sscc.ru/MVS-1000/MVS-1000.htm
2. http://www.t-systems.ru/
3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 609 с.

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПРОВЕРКИ ЧИСЛА НА ПРОСТОТУ

А.В. Кузнецова

Томский государственный университет

Во многих алгоритмах шифрования современных криптосистем используются простые числа большого размера, генерация которых является нетривиальной задачей. Один из методов решения данной задачи заключается в следующем: генерируется в качестве кандидата случайное число n подходящего размера, с помощью специальных алгоритмов выполняется его проверка на простоту, и, если число оказывается составным, то генерируется следующее число, и т.д. Для проверки числа на простоту известны две основные группы тестов: детерминированные тесты, применяемые только для чисел специального вида; вероятностные тесты, которые с некоторой вероятностью утверждают, что число простое, и точно определяют, что число составное, или наоборот.

В данной работе рассматривается вероятностный тест Миллера–Рабина. Если в результате выполнения этого теста утверждается, что число n – составное, то это действительно так, а если n – простое, то это верно с некоторой вероятностью.

Тест Миллера–Рабина основан на теореме Ферма, которая утверждает, что если n – простое число, то $a^{n-1} = 1 \bmod n$, где a – любое целое число ($1 < a < n - 1$). Обратное утверждение теоремы неверно. В тесте Миллера–Рабина для набора чисел $a_1, a_2, ..., a_m$, называемых основаниями, проверяется выполнение тождества $a_i^{n-1} = 1 \bmod n$ ($i = 1, ..., m$) и еще ряд дополнительных условий. Если для всех оснований тождество и условия выполняются, то число n объявляется простым с вероятностью ошибки меньшей, чем $(1/4)^m$, в противном случае число объявляется составным. Отметим, что чем больше число проверяемых оснований, тем меньше вероятность ошибки. Проверка условий теста может выполняться одновременно и независимо для нескольких оснований, что позволяет эффективно использовать параллельные вычисления. Следующий параллельный алгоритм ориентирован на использование многопроцессорной системы с распределенной памятью.

Пусть s – число процессов, соответствующее числу процессоров, задействованных в вычислениях; m – требуемое число оснований, причем оно при необходимости увеличивается до числа, кратного s. Тогда каждый из процессов должен протестировать заданное число n по $r = m/s$ основа-

ниям. Тестируемое число n подается на вход управляющему процессу, который рассылает его и число r остальным. Затем каждый из процессов случайным образом выбирает r оснований и выполняет для них проверку условий теста Миллера–Рабина. Если хотя бы один из процессов сообщает, что число n – составное, то n объявляется составным и все процессы завершаются. Если же результаты всех процессов таковы, что n – простое, то n объявляется простым с вероятностью ошибки $(1/4)^m$.

Для достижения наибольшего ускорения число процессоров рекомендуется брать кратным числу оснований. Однако даже при равном числе оснований проверка условий теста для различных оснований может выполняться не за одинаковое время, что приводит к простаиванию процессоров и снижает эффективность алгоритма. Время работы теста для одного основания зависит как от выбранного основания, так и от длины числа в его двоичном представлении.

Эксперименты, проведенные на кластерной системе, показали, что для небольших чисел (порядка 100 бит) ускорение не велико и эффективность использования системы падает с ростом числа процессоров. Причина в том, что время, затрачиваемое на передачу числа и сбор результатов, составляет большую долю от общего времени работы, т.е. относительно велико по сравнению со временем, затрачиваемым непосредственно на вычисления. Для чисел большого размера (порядка 256, 512, 1024 бит) время, требующееся для вычислений, возрастает, процент параллельных вычислений увеличивается и, как следствие, возрастает эффективность использования системы.

Таким образом, предложенный метод позволяет ускорить проверку простоты очередного сгенерированного случайного числа и при отрицательном ответе перейти к следующему.

Далее нами предполагается разрабатывать параллельные алгоритмы, которые специальным образом строят заведомо простые большие числа.

АЛГОРИТМ СБОРКИ МИНИМАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ФРАГМЕНТОВ В СБОРОЧНОЙ ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

С.П. Нечаев

Новосибирский государственный университет

1. Введение

Сборочная технология представляет собой новый подход к созданию параллельных программ. Она заключается в том, что параллельная программа собирается из заранее заготовленных атомарных фрагментов вычислений. Преимущества такого подхода в следующем: достаточно один раз создать атомарный фрагмент, а потом можно в зависимости от конфигурации мультикомпьютера и размера задачи собирать из таких фрагментов параллельную программу, которая оптимальным образом использует доступные ресурсы [1].

Основные принципы сборочной технологии заключаются в следующем:

• Программа конструируется из небольших атомарных фрагментов вычислений (код + обрабатываемые им данные), далее фрагментов.

• Фрагментированность программы сохраняется в ходе вычислений.

• Фрагменты программы в ходе вычислений мигрируют между процессорными элементами (ПЭ) мультикомпьютера для обеспечения равной загрузки всех его ресурсов.

В программировании численных моделей основным атомарным фрагментом является ячейка (узел) сетки (все переменные ячейки и код их обработки). Небольшой размер фрагмента имеет следствием простоту конструирования программы (ее сборки). Однако качество исполнения собираемой программы (ее производительность, потребляемые ресурсы и т.д.), собранной из атомарных фрагментов-ячеек, оказывается низким из-за больших затрат на реализацию управления и структур данных. Для повышения качества программ необходимо собирать их из фрагментов подходящих размеров и структуры [2].

2. Динамическая балансировка загрузки

Известно, что равная загрузка ПЭ мультикомпьютера способствует повышению эффективности параллельной программы. Некоторые вычисления, например метод частиц в ячейках или адаптивные разностные схе-

мы, устроены так, что загруженность ПЭ во время работы программы меняется непредсказуемым образом. Поэтому для эффективной работы программы требуется динамическая балансировка загрузки. При использовании сборочной технологии для обеспечения динамической балансировки загрузки нужно поддерживать фрагментированную структуру программ во время их выполнения. Но если поддерживать фрагментацию на уровне атомарных фрагментов, то затраты на это превысят получаемый от такого подхода выигрыш. Поэтому предлагается двухуровневая фрагментация программ.

Сначала пользователем создается атомарный фрагмент – фрагмент, который отражает смысл поставленной задачи. Размер атомарного фрагмента должен быть как можно меньше, чтобы упростить формулировку задачи в терминах программы. Для уменьшения потерь собираемой программы на реализацию управления из атомарных фрагментов собирается минимальный фрагмент. При этом его вид определяется особенностями задачи и архитектуры вычислительной системы. В численных алгоритмах это может быть, например, слой или колонка ячеек. Сборка минимального фрагмента выполняется до начала работы программы, а во время вычислений он неделим. Таким образом, собираемая программа описывается в терминах атомарных фрагментов, а на исполнение идет программа, описанная в терминах минимальных фрагментов, в которых уже частично учтены особенности и задачи, и вычислителя.

Во время работы программы полученные минимальные фрагменты могут перемещаться от перегруженного ПЭ к недогруженному для выравнивания загрузки [1].

3. Сборка

При сборке всем тем переменным атомарных фрагментов, которые соответствуют значению одной и той же функции в одной и той же точке пространства, ставится в соответствие одна переменная минимального фрагмента и указывается правило, по которому значение новой переменной вычисляется как функция от значений исходных (рис. 1).

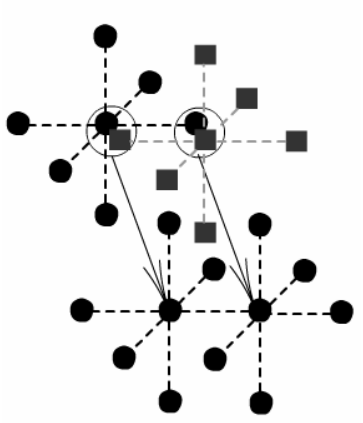


Рис. 1. Сборка

Рис 1. Сборка.

4. Ограничения

Созданный алгоритм работает при следующих достаточно серьезных ограничениях:

1. Вычисления производятся в прямоугольной системе координат.

2. Код атомарного фрагмента изменяет значение не более одной переменной.

Последнее ограничение связано с тем, что один и тот же элемент

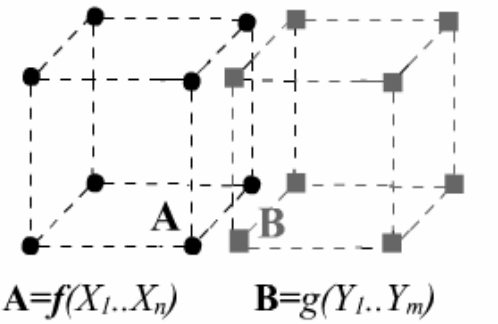


Рис. 2 ...менных

массива, полученный в результате сборки, должен обрабатываться двояким образом: как переменная A клетки одного фрагмента и как переменная B соседнего. В общем случае значения этих переменных вычисляются как разные функции от разных аргументов, поэтому нужно адекватным образом отразить вклад каждой из собираемых переменных в значение получающегося элемента массива.

5. Описание алгоритма

Основная задача заключалась в разработке алгоритма, который по заданному алгоритму сборки параллельной программы и коду атомарного фрагмента генерирует минимальный фрагмент, т.е. конструирует код и переменные.

Следующий пример иллюстрирует работу алгоритма. Пусть задан атомарный фрагмент – шаблон сетки, код, его обрабатывающий, размер и структура минимального фрагмента, который необходимо сконструировать. Предполагается, что минимальный фрагмент всегда имеет прямоугольную структуру (полоса или слой атомарных структур). Размер выражается числом атомарных фрагментов, собираемых по соответствующему направлению. Например, размер 4×2×3 означает, что минимальный фрагмент собирается из четырех атомарных фрагментов по оси X, двух по Y и трех по Z.

На рис. 3 изображены атомарный фрагмент и минимальный фрагмент, который требуется собрать.

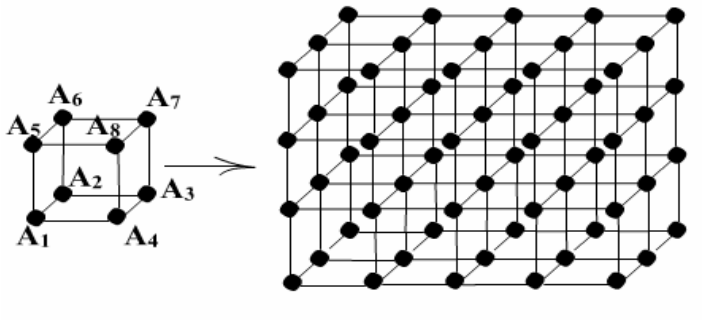


Рис. 3 Атомарный и минимальный фрагменты

Алгоритм состоит из следующих этапов:

На первом этапе для каждого из трех направлений (X, Y, Z) строятся списки переменных, которые должны стать элементами одного и того же массива. Вычисляются размеры каждого из массивов. Для иллюстрации этого этапа присоединим к исходному атомарному фрагменту по такому же фрагменту с каждого из трех направлений.

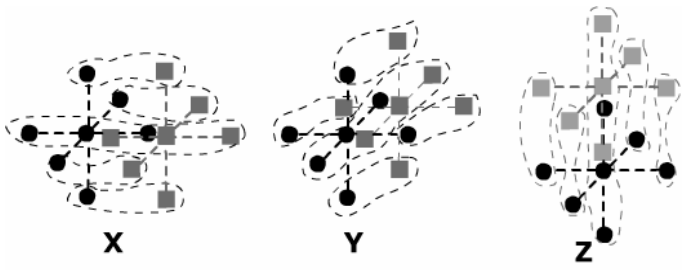


Рис. 4 Переменные – элементы одного массива

На рис. 4 обведены те переменные, которые должны быть элементами одного массива по соответствующему направлению.

На втором этапе выбирается переменная исходного атомарного фрагмента, которая «связывает» все три направления. Эта переменная должна входить во все три массива, причем в каждый из массивов еще как минимум с одной переменной из того же самого фрагмента. На рис. 5 эта пере-

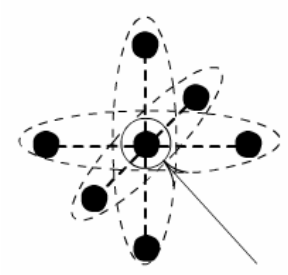


Рис. 5 Переменная, связывающая все три направления

менная обведена сплошной линией. Если такой переменной не найдется, значит, по данной схеме сборку невозможно произвести.

На третьем этапе объявляется массив (размеры его были вычислены ранее), добавляется заголовок цикла, который будет обрабатывать собранную область, и затем по всему коду исходного атомарного фрагмента имена переменных заменяются на соответствующие имена элементов массива. Предполагается, что наиболее часто в качестве минимальных фрагментов будут использоваться «полоса» и «слой» (рис. 6). Удобно будет их рассматривать как трехмерные массивы размеров $a_1 \times a_2 \times a_3$, где $a_i \in N$, $\exists!\ a_i \neq 1$ для полосы и вида $a_1 \times a_2 \times a_3$, где $a_i \in N$, $\exists!\ a_i \neq 1$ для слоя.

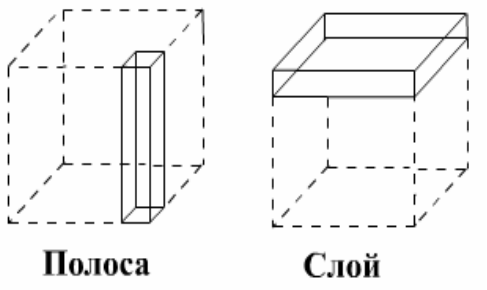


Рис. 6 Основные виды минимальных фрагментов

Заключение

Сборочная технология – перспективный подход к созданию высокоэффективных параллельных программ.

Преимущества сборочной технологии следующие:

1. Эффективное использование ресурсов. Фрагментированная структура программы обеспечивает возможность перемещения вычислений от перегруженных ПЭ к недогруженным для балансировки загрузки.

2. Автоматическая сборка программ. Пользователю необязательно обладать узкоспециальными знаниями по параллельным вычислениям. Он пишет лишь последовательный код своей программы или ее части, а параллельный код генерируется автоматически.

3. Простота отладки. Поскольку пользователь имеет дело только с кодом атомарного фрагмента, а его объем много меньше объема кода всей программы, то отладить и оптимизировать его значительно проще. Разработанный алгоритм реализует один из ключевых моментов сборочной технологии – сборку минимального фрагмента вычислений. Разработана программа, реализующая данный алгоритм. В дальнейшем она будет встроена в систему сборочного асинхронного параллельного программирования.

Литература

1. Kraeva M., Malyshkin V. Implementation of PIC Method on MIMD Multicomputers with Assembly Technology //Proc. of HPCN Europe 1997 (High Performance Computing and Networking) Int. Conference, LNCS. – Springer Verlag, 1997. – Vol.1255. – P. 541–549.
2. Kraeva M.A., Malyshkin V.E. Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers //In the special issue of the International Journal on Future Generation Computer Systems, devoted to Parallel Computing Technologies. – 2001. – Vol. 17, № 6. – P. 755–765.

ПОСЛЕДОВАТЕЛЬНЫЙ И ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМЫ КЛАСТЕРИЗАЦИИ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

О.И. Нечаева

Новосибирский государственный университет

Введение

Во многих областях науки возникает задача, в которой требуется разбить множество некоторых объектов на подмножества, элементы которых близки в некотором смысле. Такого рода задачи называются задачами *кластеризации* множества объектов, а получившиеся подмножества – *кластерами*. При этом следует заметить, что понятие «кластер» в данном случае определяет подмножество близких между собой элементов, а не вычислительную систему, состоящую из нескольких компьютеров.

В данной работе в качестве объектов выступают символьные последовательности, т.е. последовательности символов некоторого конечного алфавита.

Кластеризация применяется в самых разнообразных областях. Например, в медицине – для систематики заболеваний и их лечения, в генетике – для изучения взаимосвязи между структурой генетической цепочки и информацией, зашифрованной в различных её частях, в археологии – для установления таксономии различных находок, в музыке – для выявления структуры мелодии или установления плагиата и т.д.

Одним из важнейших методов кластеризации является «итерационный метод динамических ядер» [1], или метод «k-средних».

В статье рассматривается последовательная и параллельная реализация алгоритма кластеризации символьных последовательностей методом k-средних с использованием расстояния Левенштейна.

Задача кластеризации и последовательный алгоритм её решения методом k-средних заключаются в следующем. Пусть Y – это некоторый *алфавит* символов, Y* – множество всевозможных последовательностей символов из Y. Рассмотрим некоторое множество символьных последовательностей $X \subseteq Y^*$, $X = \{ x^1, \ldots, x^m \}$, где $x^i = [y^i_1, y^i_2, \ldots, y^i_{Ni}]$, $y^i_j \in Y$, N_i – длина i-й последовательности.

Для того чтобы проводить кластеризацию множества символьных последовательностей, нужно определить меру близости между его элемен-

тами. Для символьных последовательностей одной из наиболее распространённых мер близости является расстояние Левенштейна.

Расстоянием Левенштейна $d(x^1, x^2)$ между двумя последовательностями $x^1 = [\, y^1_1, y^1_2, \ldots, y^1_{N1}\,]$ и $x^2 = [\, y^2_1, y^2_2, \ldots, y^2_{N2}\,]$ называется неотрицательное целое число, равное минимальному количеству операций «вставок», «удалений» и «замен» символов в x^1, необходимых, чтобы из x^1 получить x^2. Вычислить расстояние Левенштейна можно, например, с помощью алгоритма Вагнера–Фишера [2], который основан на методе динамического программирования.

В процессе вычисления расстояния между последовательностями $[\, y^1_1, \ldots, y^1_i, \ldots, y^1_{N1}\,]$ и $[\, y^2_1, \ldots, y^2_j, \ldots, y^2_{N2}\,]$ заполняется массив $\{d_{i,j}\}$ размером $(N_1 + 1)\times(N_2 + 1)$ по рекуррентной формуле

$$d_{i,j} = \begin{cases} i, & j = 0 \\ j, & i = 0 \\ \min\left\{d_{i-1\,j} + 1, d_{ij-1} + 1, d_{i-1\,j-1} + \delta\left(y^1_i, y^2_j\right)\right\}, & i > 0, j > 0 \end{cases} \qquad (1)$$

где $d_{i,j} = d([\, y^1_1, y^1_2, \ldots, y^1_i\,], [\, y^2_1, y^2_2, \ldots, y^2_j\,])$, $\delta(y^1_i, y^2_j)=1$, если $y^1_i \neq y^2_j$, и $\delta(y^1_i, y^2_j)=0$ в противном случае. Искомым расстоянием Левенштейна между x^1 и x^2 является число, равное последнему элементу массива $\{d_{i,j}\}$, т.е. $d(x^1, x^2)=d_{N1,N2}$.

Для решения задачи кластеризации применим метод kсредних [1], описание которого приводится ниже.

Пусть $P_1, \ldots, P_k$, где k – зафиксировано, $k \le m$, разбиение множества X на кластеры, при этом $X = P_1 \cup P_2 \cup \ldots \cup P_k$, $P_i \cap P_i = \varnothing$ для всех пар (i, j). С каждым кластером P_i связан некоторый элемент a^i из Y* – «типичный представитель», или *ядро* кластера, и отнесение очередной последовательности к кластеру проводится путём сравнения её с ядрами всех кластеров и выбора ближайшего. Ядро a^i является аналогом центра тяжести кластера P_i и должно удовлетворять соотношению

$$a^i = \arg\min_{a \in A} \sum_{x \in P_i} d(a,x),\ i = 1,\ldots,k, \qquad (2)$$

где $A \subseteq Y^*$ – множество ядер, которое в общем случае не совпадает с множеством X.

Математическая постановка метода k-средних заключается в том, чтобы найти набор ядер $a^1, \ldots, a^k$ и разбиение множества X на кластеры $P_1, P_2, \ldots, P_k$, минимизирующие следующую величину:

$$D = \sum_{i=1}^{k} \sum_{x \in P_i} d\left(a^i, x\right). \qquad (3)$$

Ниже приведены основные шаги последовательного варианта итерационного алгоритма кластеризации методом k-средних. Будем обозначать через n номер итерации.

Последовательный алгоритм:

0. Фиксируется начальное разбиение $P_1(0), ..., P_k(0)$ произвольно либо по какому-нибудь эвристическому правилу, и в каждом множестве $P_i(0)$ вычисляется ядро $a^i(0)$, $i = 1...k$.

1. На итерации с номером n для каждой последовательности x^j, $j = 1...m$, вычисляются расстояния до всех k ядер и выбирается среди них минимальное:

$$d(x^j, a^i(n)) = min \{ d(x^j, a^1(n)), ..., d(x^j, a^k(n)) \}. \quad (4)$$

Последовательность x^j добавляется к множеству $P_i(n+1)$. В случае, когда минимум достигается при нескольких значениях i, выбор между ними может быть сделан произвольно.

2. Если $P_i(n+1)=P_i(n)$ для каждого $i = 1...k$, то алгоритм останавливается. Иначе в каждом множестве $P_i(n+1)$ вычисляется новое ядро $\hat{a}^i$, и делается переход к шагу 1 с найденными ядрами: $a^i(n+1) := \hat{a}^i$, $i = 1...k$.

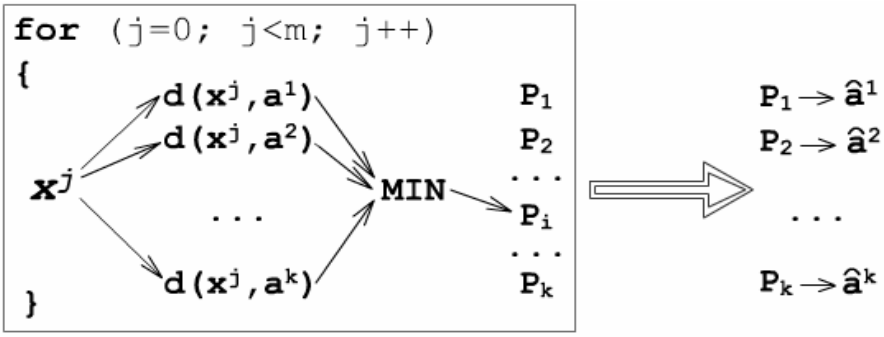


Рис. 1 Схема одной итерации алгоритма

На каждой итерации алгоритма уменьшается критерий качества D, отсюда следует сходимость алгоритма – после конечного числа шагов разбиение $P_1, P_2, ..., P_k$ уже не меняется.

Главная вычислительная сложность в методе k-средних с расстоянием Левенштейна – это *поиск ядра в кластере*. Задача нахождения «центра тяжести» с использованием этого расстояния, вообще говоря, является NP-полной.

Если искать ядро полным перебором, то сложность будет порядка $O(tN^2|\Sigma|^N)$, где t – количество последовательностей в кластере P; N – это максимум среди длин N_i, $i = 1...t$; Σ – множество символов, встречающихся в последовательностях кластера, $\Sigma \subseteq Y$, $|\Sigma|$ – количество этих символов. Для уменьшения сложности можно использовать эвристический алгоритм, который основан на том, что ядро выбирается среди последовательностей кластера.

Пусть $x^1, ..., x^t$ – последовательности кластера P. Тогда ядром кластера будем называть $x^p \in P$, сумма расстояний от которого до всех элементов кластера минимальна:

$$x^p = arg \min_{x \in P} \sum_{i=1}^{t} d(x, x^i). \quad (5)$$

Сложность нахождения ядра в кластере P равна $O(N^2 t^2)$, так как вычисление расстояния Левенштейна для двух последовательностей x^i и x^j имеет сложность $O(N_i N_j)$ и вычисляется t^2 раз.

Однако если проводится кластеризация большого числа длинных последовательностей, этот эвристический алгоритм также практически неприменим. Существуют методы, позволяющие уменьшить время вычислений. Например, можно построить более эффективный по качеству или по времени алгоритм нахождения ядра [3], но это возможно сделать, лишь обладая информацией о некоторых специфических свойствах рассматриваемых последовательностей. Ещё один метод основан на переходе от последовательностей к их частотным словарям, а значит, в евклидово пространство, в котором все вычисления на каждой итерации линейны [4]. Однако этот метод эффективен только для последовательностей с небольшим алфавитом, таких как последовательности ДНК с алфавитом из четырёх символов.

В основном такого рода методы требуют ограничений на последовательности. Более универсальным способом ускорения вычислений является распараллеливание алгоритма метода k-средних.

Параллельная реализация метода k-средних

Если проанализировать последовательный алгоритм метода k-средних, то можно заметить в нем внутренний параллелизм, который заключается, во-первых, в том, что при распределении входных последовательностей по кластерам вычислять расстояния от ядер до элементов X можно одновременно. Во-вторых, вычисление ядер в кластерах может происходить независимо друг от друга. На основании этого можно построить параллельный алгоритм кластеризации методом k-средних.

Пусть $X = \{ x^1, ..., x^m \}$ – множество последовательностей, которое нужно разбить на k кластеров. Тогда для параллельной реализации потребуется $k+1$ процессоров, среди которых имеется один управляющий и k рабочих процессоров. Все процессоры соединены по принципу «каждый с каждым», и кластеру P_j соответствует j-й рабочий процессор.

Управляющий процессор фиксирует начальное разбиение $P_1(0), ..., P_k(0)$ произвольно либо по какому-нибудь эвристическому правилу и каждому j-му рабочему процессору посылает множество $P_j(0)$ соответственно. После остановки параллельного итерационного алгоритма результат кластеризации возвращается в управляющий процессор.

Параллельный итерационный алгоритм для рабочих процессоров

0. Исходными данными в j-м процессоре является множество $P_j(0)$, $j = 1...k$.

На n-м шаге выполняются следующие операции:

1. Каждый j-й процессор вычисляет ядро $a^j(n)$ множества $P_j(n)$, $j = 1...k$.

2. Все процессоры обмениваются найденными ядрами так, что в результате обмена в каждом из них оказываются все k ядер.

3. В каждом j-м процессоре элементы множества $P_j(n)$ распределяются по подмножествам $P_1'(n), ..., P_k'(n)$ следующим образом: для каждой последовательности $x \in P_j(n)$ вычисляются расстояния до всех k ядер и x помещается в $P_i'(n)$, у которого ядро имеет минимальное расстояние до x по (4). В случае, когда минимум достигается при нескольких значениях i, выбор между ними может быть сделан произвольно.

4. Каждый j-й процессор рассылает множества $P_i'(n)$, $i = 1...k$, $i \neq j$, соответствующим рабочим процессорам и от каждого из них получает подмножество $P_j'(n)$.

5. В каждом j-м процессоре множество $P_j(n+1) = \cup P_i'$, $i = 1...k$, является исходным для следующего итерационного шага.

6. Если во всех процессорах $P_j(n+1) = P_j(n)$, $j = 1...k$, то $P_1(n), ..., P_k(n)$ является результатом кластеризации.

Пункты 1 и 3 алгоритма являются наиболее затратными по времени и выполняются одновременно во всех рабочих процессорах. Поэтому использование этого параллельного алгоритма метода k-средних позволяет уменьшить сложность вычислений примерно в k раз.

Заключение

Исследована последовательная реализация метода k-средних с расстоянием Левенштейна для кластеризации символьных последовательностей.

Сделан вывод о том, что для большого числа символьных последовательностей этот метод практически неприменим, и сократить время вычислений зачастую возможно только для узкого класса последовательностей.

Параллельная реализация позволяет обойти эти ограничения и ускорить вычисления в среднем во столько раз, сколько кластеров нужно получить.

Литература

1. Albayrak S. Unsupervised Clustering Methods for Medical Data: An Application to Thyroid Gland Data // Lecture Notes in Compute Science. – Berlin: Springer-Verlag, 2003. – Vol. 2714. – P. 695.

2. Lee C.S. George. Neural Fuzzy Systems. – New Jersey: Prentice Hall P T R, 1996.
3. Gorban A.N., Porova T.G., Sadovsky M.G. Classification of Symbol Sequences over their Frequency Dictionaries: Towards the Connection Between Structure and Natural Taxonomy // Open Sys. Inform. Dyn. – 2000. – № 7. – P.1.
4. Кузнецов П.Г. Обучение при распознавании символьных последовательностей // Вычислительные системы. – 1994. – № 150. – С. 164.

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА ФАКТОРИЗАЦИИ «КВАДРАТИЧНОЕ РЕШЕТО»

И.И. Худяшов

Томский государственный университет

На экспоненциальной сложности задачи факторизации числа основана стойкость многих современных криптосистем. В данной работе рассматривается алгоритм факторизации «Квадратичное решето» [1]. Распараллеливание этого алгоритма не снизит его экспоненциальную сложность до полиномиальной, но в некоторых случаях может заметно понизить время его работы.

Алгоритм факторизации «Квадратичное решето» решает задачу 2-факторизации, в которой требуется найти нетривиальный делитель d заданного составного числа n.

Рассматриваемый алгоритм относится к методам, основанным на следующей идее. Пусть n – число, которое необходимо факторизовать, и пусть найдены такие a и b, что $a^2 = b^2 \bmod n$, но $a \neq \pm b \bmod n$; тогда нетривиальный делитель d определяется как наибольший общий делитель чисел $a–b$ и n или $a+b$ и n. Методы отличаются друг от друга способом нахождения чисел a и b, удовлетворяющих указанным условиям.

В алгоритме фиксируется t первых простых чисел, образующих множество S, называемое фактор-базой. Находятся пары чисел (x_i, y_i) такие, что $x_i^2 = y_i \bmod n$ ($i = 1,\ldots t, t+1$), и все простые делители числа y_i входят в множество S. После этого составляется и решается однородная система из t линейных уравнений с булевыми векторами размера $(t+1)$ в качестве решений. Из множества последних берется ненулевой булев вектор, и по нему выписываются пары чисел (x_i, y_i), соответствующие единичным компонентам вектора. Затем вычисляется число a как произведение выписанных x_i и число b как произведение квадратных корней из выписанных y_i. По найденным a и b, если они удовлетворяют описанным выше условиям, находится d – нетривиальный делитель n.

Вся сложность этого алгоритма заключена в поиске $(t+1)$ пар чисел (x_i, y_i), который заключается в переборе $O(2^k)$ чисел, где k – число бит в двоичном представлении числа n. Распараллеливание именно этого шага позволит уменьшить время работы алгоритма. Взаимодействие параллельных процессов осуществляется следующим образом.

Пусть имеется s+1 процессов, один из них назовем управляющим, а остальные – рабочими. На вход управляющему процессу поступают число n и размер t фактор-базы S. Управляющий процесс, после построения им S, рассылает n и S рабочим процессам. Рабочий процесс с номером r строит свои пары чисел (x_i, y_i) по следующим правилам: $x_1 = [\sqrt{n}] + r$, $y_1 = x_1^2 \bmod n$; $x_i = x_{i-1} + k$, $y_i = x_i^2 \bmod n$ для $i > 1$. Для каждой из таких пар проверяется возможность разложения y_i в фактор-базе. Как только находится подходящая пара, она передается управляющему процессу, который после набора необходимого числа пар останавливает работу всех рабочих процессов и продолжает вычисления согласно алгоритму. В случае, если необходимое число пар не найдено в течение заданного интервала времени, управляющий процесс принудительно завершает работу рабочих процессов и сообщает о невозможности факторизации числа за указанное время.

Литература

1. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. – CRC Press, 1997.

ИСПОЛЬЗОВАНИЕ ВИРТУАЛЬНЫХ ТОПОЛОГИЙ НА МУЛЬТИКОМПЬЮТЕРАХ

А.В. Черкасов

Новосибирский государственный технический университет

Введение

На сегодняшний день наиболее распространённым средством параллельной реализации алгоритмов численного моделирования является MPI (Message Passing Interface) [3]. MPI представляет собой набор спецификаций для написания параллельных программ (ПП), взаимодействующих по средством обмена сообщениями. Известно, что работа таких программ состоит из двух чередующихся фаз: фазы вычислений и фазы обменов. Следовательно, эффективность работы ПП зависит не только от качественного программирования вычислений в узлах многопроцессорной вычислительной системы (МВС), но и от продуманной организации взаимодействия разных частей программы (подробнее см. [1] и [2]).

В работе рассматривается зависимость времени выполнения межпроцессорных обменов в MPI-программе от использования в ней различных типов виртуальных топологий и различных способов рассылки данных. Исследование проведено на разных мультикомпьютерах. Для проведения эксперимента реализована MPI-программа, которая конструирует заданную виртуальную топологию и измеряет пропускную способность всех связей в ней. Пропускная способность определяется как время передачи данных установленного размера между узлами мультикомпьютера.

1. Среда эксперимента

Исследование проводилось на следующих мультикомпьютерах:

- кластер МВС-1000/М, Москва (768CPUs: 2×Alpha 21264 (667 MHz), Myrinet2000, шесть 128-входовых коммутаторов) [4];
- кластер МВС-1000/М, Новосибирск (32CPUs: 2×Alpha 21264 (833 MHz), Myrinet1000, 16-входовой коммутатор) [5];
- кластер T-Forge32 (32CPUs: 2×Opteron244 (1,8 GHz), Gigabit Ethernet, 16-входовой коммутатор) [6].

2. Использование различных типов ВТ

Под виртуальной топологией (ВТ) будем понимать заданное соединение некоторого множества виртуальных процессоров. Механизм ВТ даёт

разработчику мощный и гибкий инструмент для отображения ПП на физические ресурсы многопроцессорной системы. При этом программная среда автоматически осуществляет отображение виртуальных связей, используемых в программе, на физические коммуникации МВС.

Можно априорно предположить, что в случае, когда все процессоры МВС взаимодействуют через один коммутатор, т.е. по полному графу, то время взаимодействия любой пары процессоров такой системы одинаково с точностью до несущественных отклонений. Следовательно, одна и та же MPI-программа, использующая разные ВТ, размер которых не превосходит числа физических процессоров системы, будет одинаково быстро работать. Это объясняется тем, что любой граф, которым может быть представлена ВТ, является подграфом полного графа, представляющего физические связи кластера. Пример отображения ВТ на некоторую МВС с одним коммутатором приведен на рис.1.

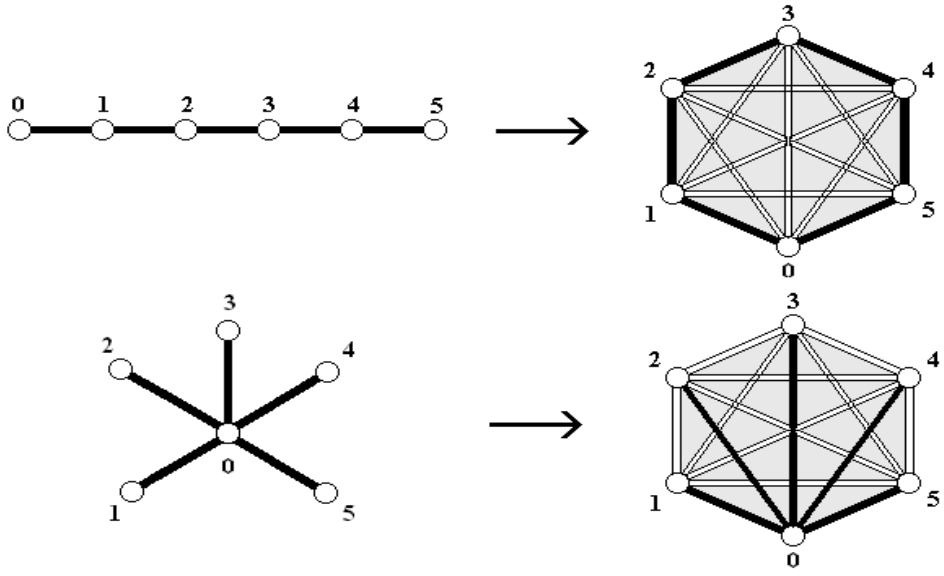


Рис. 1 Отображение виртуальных топологий “линейка” и “звезда” на коммуникации 6-процессорной МВС

В эксперименте исследованы классические типы ВТ, такие как линейка, 2D и 3D решётки, тор, звезда, дерево и полный граф. Размер передаваемого сообщения – 5 Мб. Для определения пропускной способности связи производилось 100 пересылок тестового сообщения. Исследование проводилось только для взаимодействий типа “точка–точка”.

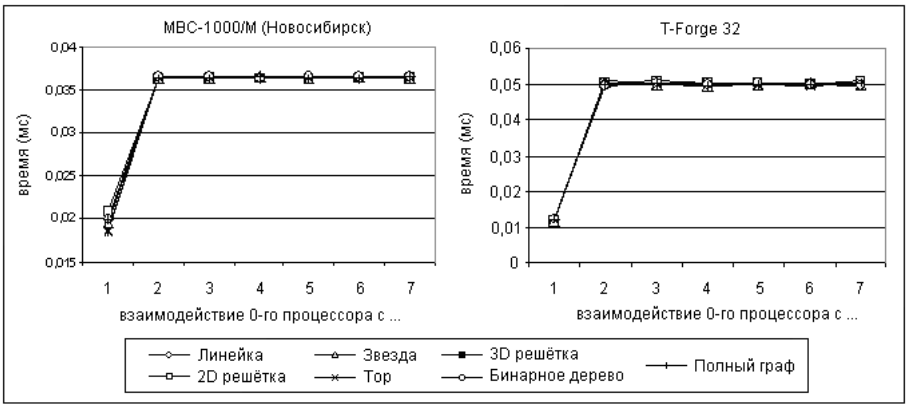


Рис. 2 Зависимость времени передачи сообщений от типа ВТ

Зависимость времени межпроцессорного обмена от типа ВТ для кластеров с одним коммутатором представлена на рис.2. Размер тестовой топологии фиксирован – 8 процессоров. Как видно из рисунка, из общей картины выделяется время взаимодействия 0-го процессора с 1-м процессором. Это объясняется тем, что тестируемые кластеры состоят из двухпроцессорных SMP-узлов и взаимодействие процессоров одного узла происходит быстрее, чем взаимодействие процессоров разных узлов (в приведенном примере 0-й и 1-й процессоры находятся в одном вычислительном узле).

Выводы. Из результатов следует, что наше априорное предположение получило практическое подтверждение. А именно: на кластерах с одним физическим коммутатором нет разницы, каким типом ВТ пользоваться.

Однако для “больших” кластеров, в которых количество коммутаторов больше одного, ситуация меняется: физическая топология системы может оказывать влияние на время работы программы, хотя обычно это ослабляется использованием более быстрых коммуникационных устройств.

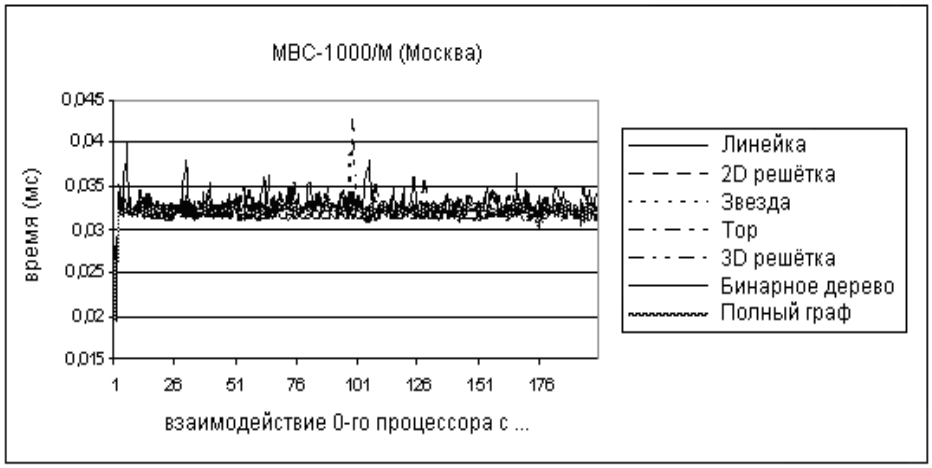


Рис. 3 Зависимость времени передачи сообщений от типа ВТ

Рассмотрим в качестве “большого” кластера московский мультикомпьютер МВС-1000/М. Результаты тестирования этой системы на ВТ размерностью 200 процессоров приведены на рис. 3. В данном случае разброс значений времени для различных ВТ больше, чем, например, для новосибирского кластера. Причина этого заключается в том, что запускаемая программа размещается на процессорах, осуществляющих передачу сообщений более чем через один коммутатор. Поэтому время взаимодействия разных пар процессоров различается.

Выводы. Исходя из результатов, полученных для московского кластера МВС-1000/М, нельзя выделить ВТ какого-либо типа как наиболее предпочтительную для организации межпроцессорных взаимодействий на данном кластере. Все рассмотренные типы ВТ являются одинаково эффективными.

В общем случае для “больших” кластеров рекомендуется провести сначала тестирование кластера на различных ВТ, а затем, основываясь на полученных результатах, выбрать ту или иную ВТ.

3. Способы рассылки данных в MPI-программах

Теперь несколько слов о способах рассылки данных в ПП. Допустим, есть некоторый процессор P, которому нужно разослать свои блоки данных D1, D2,…, DN процессорам P1, P2,…, PK. Это можно сделать двумя основными способами. Первый способ: послать один за другим блоки данных D1,…, DN процессору P1, а затем – процессору P2,…, процессору PK. Второй (ортогональный) способ: сначала послать блок данных D1 процессорам P1, P2,…, PK; затем отправить этим процессорам блок данных D2,…, DN.

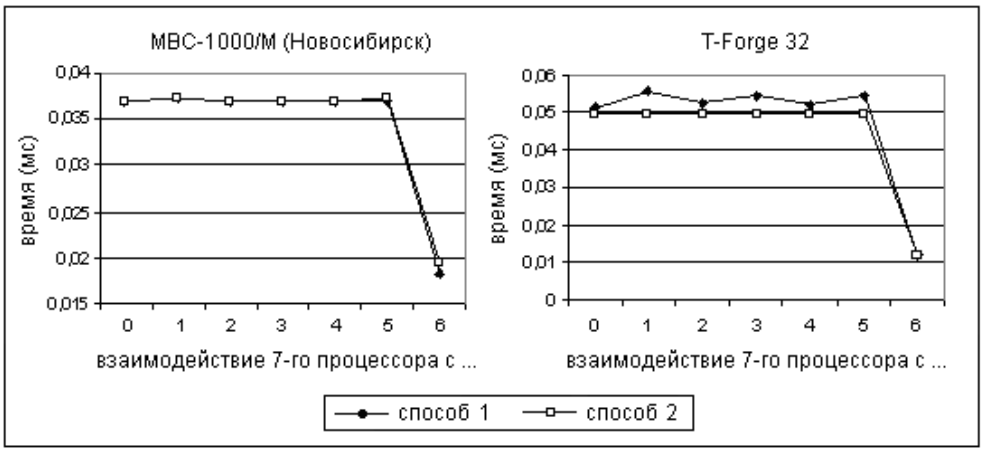


Рис. 4 Зависимость времени передачи данных в ВТ от способа их рассылки

Как видно из рис. 4 и 5, разные способы рассылки сообщений на разных мультикомпьютерах работают не одинаково быстро. Причиной тому могут быть разные реализации MPI, разные размеры системного буфера, а также разное устройство и организация буферов обмена у разных коммуникационных устройств (что имеет место на тестируемых кластерах МВС-1000/М и T-Forge32).

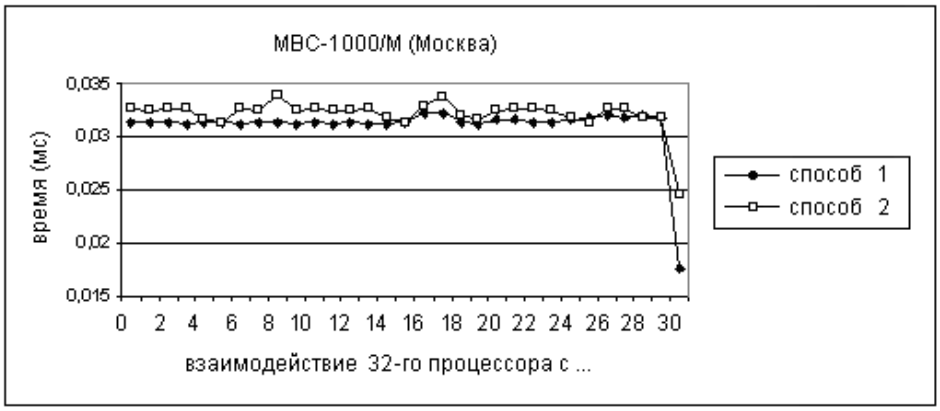


Рис. 5 Зависимость времени передачи данных в ВТ от способа их рассылки

Выводы. При организации межпроцессорного взаимодействия способом 1 на кластере МВС-1000/М (Москва), время выполнения ПП, по крайней мере не меньше, чем способом 2. Для кластера же T-Forge 32, наоборот, второй способ даёт лучшие показатели времени, чем первый. И, наконец, на системе МВС-1000/М (Новосибирск) оба способа работают одинаково быстро. Эти данные можно использовать при написании эффективных ПП для рассмотренного набора мультикомпьютеров.

Литература

1. Малышкин В.Э. Параллельное программирование мультикомпьютеров: Учебное пособие. – Ярославль: Изд-во Ярослав. ун-та, 1999. – 134 с.
2. Малышкин В.Э. Основы параллельных вычислений: Учебное пособие. – Новосибирск: Изд-во НГТУ, 1998. – 60 с.
3. Корнеев В.Д. Параллельное программирование в MPI. – Новосибирск: Изд-во СО РАН, 2000. – 213 с.
4. www.jscc.ru – Межведомственный суперкомпьютерный центр.
5. www2.sscc.ru – Сибирский суперкомпьютерный центр.
6. www.t-platforms.ru/opteron.html – ЦКТ, кластер T-Forge 32.

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МЕТОДА ПОЛНОЙ РЕДУКЦИИ

В.И. Лаева, М.А. Чижухина

Томский государственный университет

Метод полной редукции является одним из эффективных прямых методов решения разностного уравнения Пуассона в прямоугольнике [1–3]. В данной работе рассмотрен параллельный алгоритм этого метода.

Пусть требуется найти решение задачи Дирихле для уравнения Пуассона

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = -\varphi(x,y), \ (x,y) \in D, \\ u(x,y) = g(x,y), \ (x,y) \in \partial D, \end{cases} \qquad (1)$$

$$\overline{D} = \{0 \le x \le l_x, 0 \le y \le l_y\}.$$

На прямоугольной равномерной сетке

$$\varpi = \left\{ (x,y) = (i \cdot h_x, j \cdot h_y) \in \overline{D}, i = 0,...,M, j = 0,...N, h_x = \frac{l_x}{M}, h_y = \frac{l_y}{N} \right\}$$

разностный аналог задачи (1) имеет вид [3]:

$$\begin{aligned} &-\overline{Y}_{j-1} + C \cdot \overline{Y}_j - \overline{Y}_{j+1} = \overline{F}_j, 1 \le j \le N-1, \\ &\overline{Y}_0 = \overline{F}_0, \overline{Y}_N = \overline{F}_N. \end{aligned} \qquad (2)$$

где $\overline{Y_j} = (u_{1,j}, u_{2,j}, \ldots, u_{M-1,j})$ – вектор-столбец неизвестных; $\overline{F_j}$ – заданная правая часть; C – трёхдиагональная, симметричная матрица.

Идея метода редукции состоит в последовательном исключении из уравнений (2) неизвестных $\overline{Y}_j$ сначала с нечётными индексами j, затем j, кратными 2, 4 и т. д. Каждый шаг процесса исключения уменьшает число неизвестных, и если $N = 2^q$, $q > 0$, то в результате останется только одно уравнение, из которого можно найти $\overline{Y}_{N/2}$. Обратный ход метода заключается в последовательном нахождении неизвестных $\overline{Y}_j$ сначала с номерами j, кратными $N/4$, затем $N/8$, $N/16$ и т.д. Метод полной редукции можно описать следующими формулами [3]:

$$
\begin{aligned}
&C^{(0)} = C, \quad \overline{F_j^{(0)}} = \overline{F_j}, \\
&C^{(k)} = \left(C^{(k-1)}\right)^2 - 2 \cdot E, \\
&\overline{F}_j{}^{(k)} = \overline{F}_{j-2^{k-1}}{}^{(k-1)} + C^{(k-1)} \overline{F}_j{}^{(k-1)} + \overline{F}_{j+2^{k-1}}{}^{(k-1)}, \\
&j = 2^k, 2 \cdot 2^k, 3 \cdot 2^k, \ldots, N - 2^k, \\
&k = 1, 2, \ldots, q - 1;
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
&C^{(k-1)} \cdot \overline{Y}_j = \overline{F}_j{}^{(k-1)} + \overline{Y}_{j-2^{k-1}} + \overline{Y}_{j+2^{k-1}}, \\
&\overline{Y}_0 = \overline{F}_0, \ \overline{Y}_N = \overline{F}_N, \\
&j = 2^{k-1}, 3 \cdot 2^{k-1}, 5 \cdot 2^{k-1}, \ldots, N - 2^{k-1}, \\
&k = q, q-1, \ldots, 1.
\end{aligned}
\tag{4}
$$

По формулам (3) преобразуются правые части, а по формулам (4) находится решение.

$C^{(k)}$ – матричный полином степени 2^k относительно матрицы C, представимый в факторизованном виде:

$$
C^{(k)} = \prod_{l=1}^{2^K} \left(C - 2 \cdot \cos\left(\frac{(2 \cdot l - 1) \cdot \pi}{2^{k+1}} \right) \cdot E \right), k = 0, 1, \ldots q - 1,
$$

поэтому для обращения $C^{(k)}$ в (4) достаточно выполнить 2^q прогонок для трёхдиагональных матриц.

Рассматриваемый метод хорошо распараллеливается в силу того, что на каждом уровне редукции производятся независимые вычисления, которые могут выполняться одновременно. Параллельная реализация алгоритма проводилась методом декомпозиции исходной расчётной области по координате y. Таким образом, на каждом процессоре рассчитывается $z = 1 + (N + 1)/p$ уравнений, где $p = 2^r$ – количество процессоров $(0 < r < q)$. При прямом ходе максимальная степень параллелизма наблюдается до (q–r)-го шага, затем она начинает падать. Это связано с остановкой процессоров, на которых новые значения не пересчитываются, и на q–1 шаге работает только один. Пересылки проводятся на всех уровнях редукции, кроме первого и последнего, каждый процессор выполняет 4 двусторонних обмена по M чисел. При обратном ходе сначала работает только один процессор, затем за r шагов подключаются остальные. С $(r + 1)$-го шага все процессоры начинают работать автономно, и пересылки заканчиваются. Для приёма и передачи данных использовалась блокирующая процедура двустороннего обмена из библиотеки MPI (MPI_SendRecv).

Ускорение предложенного алгоритма метода полной редукции было вычислено по формуле

$$
S_p = \frac{Q}{P + \chi \cdot O},
\tag{5}
$$

где

$Q = (7 \cdot N \cdot \log_2 N + 6 \cdot \log_2 N + 1) \cdot M$ – количество арифметических операций последовательного алгоритма;

$P = \left(\frac{21}{2} \cdot N + 7 \cdot \frac{N}{p} \cdot (\log_2 N - \log_2 p - 1) + 3 \cdot (3 \cdot \log_2 p - 5) \right) \cdot M$ – количество арифметических операций параллельного алгоритма;

$O = (4 \cdot (\log_2 N - 2) + 2 \cdot \log_2 p) \cdot M$ – количество обменов, требуемых для параллельной реализации метода;

χ – количество арифметических операций с плавающей точкой, которые можно выполнить за время, необходимое для одной пересылки; M, N – размерности сетки по переменным x и y соответственно. Из формулы (5) видно, что ускорение параллельного алгоритма метода полной редукции не зависит от размерности сетки по x. Идеальное ускорение при больших N, $\chi = 0$ и $p \to N$ равно $\frac{2 \cdot \log_2 N}{3}$.

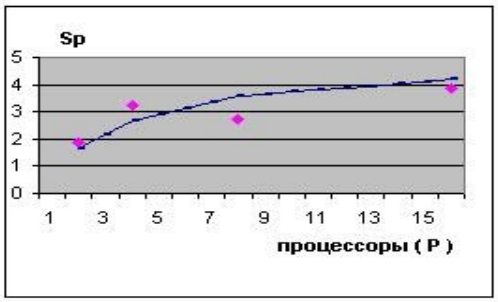


Рис. 1 Ускорение алгоритма (N=1024)

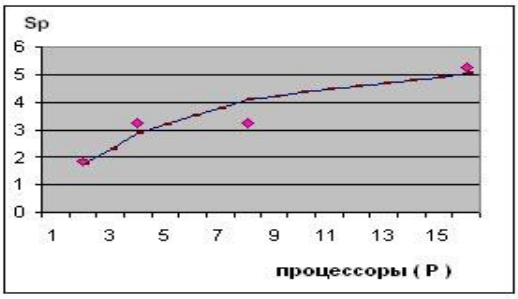


Рис. 2 Ускорение алгоритма (N=2048)

Рассматривалось численное решение задачи Дирихле для двумерного уравнения Лапласа в квадратной области с граничными условиями, равными 1. На рис. 1, 2 показано ускорение параллельного алгоритма для расчётной сетки 1024×1024 и 2048×2048 узлов. Сплошная линия соответствует аналитической зависимости, рассчитанной по формуле (5), с параметром $\chi = 100$, а ромбики – экспериментальным данным.

В таблице приведено время решения задачи на сетках с числом узлов 512×512, 1024×1024 и 2048×2048 в зависимости от используемых процессоров.

Время счета программы для различной размерности и числа используемых процессоров

N	p				
	1	2	4	8	16
512	0,7677	0,427	0,2505	0,328	0,412
1024	3,409	1,843	1,049	1,238	0,886
2048	17,83	9,58	5,49	6,39	3,31

На сетке 512×512 наименьшее время достигается на 4 процессорах и максимальное ускорение равно 3. При N=1024 время расчёта на одном процессоре больше времени расчёта на 16 процессорах в 4 раза, а при N=2048 – почти в 5,5 раз. Полученные результаты позволяют сделать вывод, что метод полной редукции хорошо распараллеливается и ускорение параллельного алгоритма возрастает с увеличением размерности расчётной сетки.

Тестирование программы проводилось на кластере ТГУ (9 двухпроцессорных узлов с локальной памятью на базе процессоров Intel Pentium-3).

Литература

1. Хокни Р., Джессхоуп К. Параллельные ЭВМ. – М.: Радио и связь, 1986. – 392 с.
2. Марчук Г.И. Методы вычислительной математики. – М.: Наука, 1980. – 536 с.
3. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. – М.: Наука, 1978. – 592 с.

ОБУЧЕНИЕ ПАРАЛЛЕЛЬНЫМ КОМПЬЮТЕРНЫМ ТЕХНОЛОГИЯМ СТУДЕНТОВ ММФ ТГУ

А.В. Старченко, В.Н. Берцун

Томский государственный университет

Характер задач, которые необходимо решать в XXI веке, существенно изменился. Возникает необходимость детального исследования явления или поведения некоторого объекта в целом. Именно суперкомпьютеры позволили приступить к расчету сложных задач науки и техники. Они дают возможность получать качественно новые результаты в фундаментальных исследованиях. Особенно эффективны такие вычислительные системы для задач с показательной сложностью вычислений. В настоящее время суперкомпьютеры начинают активно применять в таких областях, как прогноз погоды и изучение глобальных климатических изменений, проектирование интегральных схем, генная инженерия, томография, восстановление изображений, анализ загрязнения окружающей среды, кардиология, решение задач глобальной оптимизации, разведка и освоение газонефтяных месторождений и т.п. Наличие достаточного количества квалифицированных специалистов в области высокопроизводительных технологий позволит существенно сократить отставание России в проведении прогностических исследований во многих стратегически важных направлениях науки, в том числе и фундаментальной. Потребность в таких специалистах имеет тенденцию устойчивого роста, как и количество суперЭВМ в России.

Сравнительно недорогим способом наращивания мощностей вычислительных систем является создание многопроцессорных кластеров, объединяющих серийно выпускаемые персональные компьютеры. В конце 2000 года в Томском государственном университете был осуществлен запуск в эксплуатацию вычислительного кластера на базе 9 двухпроцессорных компьютеров Pentium III-650 с организацией сетевого доступа пользователей.

Однако одним из факторов, сдерживающих применение суперкомпьютеров, является неподготовленность пользователей. В связи с этим актуальным является существенное расширение компьютерного образования в области параллельных вычислений. Приток талантливой молодежи в сферу суперкомпьютерных технологий в значительной мере будет определять успехи в этой области. Поэтому важным является открытие в высших учебных заведениях специализаций и специальностей по подготовке высо-

коквалифицированных кадров.

На кафедре вычислительной математики и компьютерного моделирования механико-математического факультета Томского государственного университета был разработан перечень спецкурсов специализации 01.01.28–«Параллельные компьютерные технологии», подготовка по которой была начата с февраля 2002 года. В настоящее время по этой специализации проводится обучение студентов механико-математического факультета. Основу этой специализации составляют курсы [1]: «Теория разностных схем», «Математическая моделирование на графах», «Технологии параллельного программирования»; «Методы сплайн-функций», «Матричные вычисления на суперкомпьютерах»; «Вычислительные методы в задачах экологии», «Современные методы анализа и визуализации данных», «Методы параллельных вычислений»; «Компьютерные сети и системы», «Методы решения некорректных задач», «Современные методы решения больших задач на многопроцессорных вычислительных системах». В таблице представлено распределение указанных спецкурсов по семестрам.

Перечень специализированных курсов для обучения студентов ММФ по специализации 01.01.28 – «Параллельные компьютерные технологии»

№	Название спецкурса	Се-местр	Кол-во часов	Отчет-ность
1	Теория разностных схем, часть 1	6	32	Зачет
2	Математическое моделирование на графах	6	32	Зачет
3	Технологии параллельного программирования	6	32	Зачет
4	Теория разностных схем, часть 2	7	50	Экзамен
5	Методы сплайн-функций	7	34	Зачет
6	Матричные вычисления на суперкомпьютерах	7	52	Зачет
7	Вычислительные методы в задачах экологии	8	64	Зачет
8	Методы параллельных вычислений	8	64	Экзамен
9	Современные методы анализа и визуализации данных	8	32	Зачет
10	Методы решения некорректных задач	9	32	Зачет
11	Компьютерные сети и системы	9	48	Экзамен
12	Современные методы решения больших задач на многопроцессорных вычислительных системах	9	32	Зачет

Студенты, специализирующиеся по параллельным компьютерным технологиям, выполняют курсовые и дипломные работы, тематика которых связана с разработкой параллельных алгоритмов решения задач вычислительной математики. Обучение студентов теории и практике

организации параллельных вычислений проводится с использованием учебных пособий, подготовленных в ТГУ [2–4].

Для привлечения талантливой молодежи, желающей более глубоко освоить технологии и методы параллельного программирования, в сентябре 2001 года при ММФ ТГУ была создана студенческая научно-исследовательская лаборатория (СНИЛ) по параллельным вычислениям. Сотрудники СНИЛ обсуждают новости в мире параллельных вычислений и технологий, участвуют в создании нового математического обеспечения для многопроцессорных вычислительных систем с распределенной памятью, выступают с докладами на конференциях, поддерживают свой информационный сайт, показывающий результаты их научной деятельности.

Для обсуждения вопросов подготовки специалистов и современного состояния проблем высокопроизводительных вычислений на многопроцессорных системах в Томском государственном университете при поддержке ФЦП «Интеграция» с периодичностью в два года организуется Сибирская школа-семинар по параллельным вычислениям, первая из которых была проведена в декабре 2001 года [5].

Литература

1. http://www.math.tsu.ru
2. Миньков Л.Л., Селиховкин А.М., Тимченко С.В. Введение в методы параллельных вычислений. – Томск: Изд-во Том. ун-та, 2001. – 34 с.
3. Старченко А.В., Есаулов А.О. Параллельные вычисления на многопроцессорных вычислительных системах. – Томск: Изд-во Том. ун-та, 2002. – 56 с.
4. http://cluster.tsu.ru/other/cluster/index.htm
5. Сибирская школа-семинар по параллельным вычислениям /Под ред. проф. А.В.Старченко. – Томск: Изд-во Том. ун-та, 2002. – 129 с.

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ПРИ МОДЕЛИРОВАНИИ МЕЗОМАСШТАБНЫХ АТМОСФЕРНЫХ ПРОЦЕССОВ

А.О. Есаулов

Томский государственный университет

В работе представлены результаты расчетов атмосферных процессов, полученные с использованием негидростатической мезомасштабной модели атмосферного пограничного слоя и многопроцессорной вычислительной техники. Модель включает уравнения гидродинамики, баланса эквипотенциальной температуры и атмосферной влажности, в ней рассматривается взаимодействие атмосферы с рельефной подстилающей поверхностью с различными теплофизическими свойствами. Также учитываются фазовые переходы влаги в атмосфере, образование облачности и выпадение осадков. Расчеты производились на многопроцессорной вычислительной системе с распределенной памятью.

Введение

Процессы, происходящие в планетарном пограничном слое, играют важную роль при формировании как погоды и климата Земли в целом, так и метеорологических режимов в ограниченных масштабах (например, в масштабах города [1] или региона). Однако сложность этих процессов не позволяет воспроизводить турбулентную структуру атмосферы с использованием прямого численного моделирования. Современные способы замыкания системы уравнений гидродинамики атмосферы позволяют решать множество практических задач, тем не менее достижение высокой разрешающей способности мезомасштабных моделей остается важной проблемой моделирования. Одним из способов ее решения является использование компьютеров с параллельной архитектурой. Ниже представлена двумерная нестационарная численная модель пограничного слоя атмосферы.

1. Основные уравнения модели:

уравнение неразрывности:

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0;$$

уравнения сохранения количества движения:

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial wu}{\partial z}=-\frac{1}{\rho}\frac{\partial p_{nh}}{\partial x}+f\left(v-V_{geo}\right)+\frac{\partial}{\partial x}\left(K_H\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z^m\frac{\partial u}{\partial z}\right);$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial wv}{\partial z}=-f\left(u-U_{geo}\right)+\frac{\partial}{\partial x}\left(K_H\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z^m\frac{\partial v}{\partial z}\right);$$

$$\frac{\partial w}{\partial t}+\frac{\partial uw}{\partial x}+\frac{\partial w^2}{\partial z}=-\frac{1}{\rho}\frac{\partial p_{nh}}{\partial z}+\frac{\partial}{\partial x}\left(K_H\frac{\partial w}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z^m\frac{\partial w}{\partial z}\right).$$

Здесь t – время; u,v,w – компоненты вектора скорости ветра в направлении декартовых координат x,y,z соответственно; $\left(U_{geo},V_{geo}\right)$ – вектор скорости геострофического ветра; p_{nh} – негидростатическая компонента давления; ρ – плотность; K_H,K_z^m – коэффициенты горизонтальной и вертикальной турбулентной диффузии; f – параметр Кориолиса.

Для турбулентного замыкания используется двухпараметрическая «k-L» модель [1]:

$$\frac{\partial k}{\partial t}+\frac{\partial ku}{\partial x}+\frac{\partial kw}{\partial z}=\frac{\partial}{\partial x}\left(\sigma_e\sqrt{k}l\frac{\partial k}{\partial x}\right)+\frac{\partial}{\partial z}\left(\sigma_e\sqrt{k}l\frac{\partial k}{\partial z}\right)+P_S+P_B-c_D\frac{k^{3/2}}{l};$$

$$\frac{\partial l}{\partial t}+\frac{\partial lu}{\partial x}+\frac{\partial lw}{\partial z}=\frac{\partial}{\partial x}\left(\sigma_e\sqrt{k}l\frac{\partial l}{\partial x}\right)+\frac{\partial}{\partial x}\left(\sigma_e\sqrt{k}l\frac{\partial l}{\partial z}\right)+\frac{l}{k}\left[c_{l1}P_S+c_{l2}P_B\right]+c_{l3}\sqrt{k}\left[1-\left(\frac{l}{\kappa z}\right)^2\right];$$

$$P_S=K_z^m\left[2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial w}{\partial z}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)^2+\left(\frac{\partial v}{\partial z}\right)^2\right];$$

$$P_B=-\frac{g}{\theta_v}K_h\left(\frac{\partial\theta_v}{\partial z}-\gamma_{\theta_v}\right);$$

$$K_z^m=0{,}5\sqrt{k}l,\quad K_h=0{,}675\sqrt{k}l.$$

Здесь k – кинетическая энергия турбулентности; l – интегральный масштаб турбулентности; P_S,P_B – генерация турбулентных пульсаций за счет сдвиговых напряжений и конвективного перемешивания; K_h – коэффициент турбулентной диффузии.

В качестве уравнения баланса энергии используется уравнение переноса эквипотенциальной температуры:

$$\frac{\partial\theta_v}{\partial t}+\frac{\partial\theta_v u}{\partial x}+\frac{\partial\theta_v w}{\partial z}=\frac{\partial}{\partial x}\left(K_H\frac{\partial\theta_v}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_h\frac{\partial\theta_v}{\partial z}\right)+\frac{1}{\rho c_p}\frac{\partial R}{\partial z}-\frac{L_w\Phi_v}{c_p},$$

где последние два члена обуславливают воздействие изменения радиационного теплового баланса при распространении солнечного и отраженного излучения в атмосфере и процессов фазовых переходов влаги на температуру воздуха.

В качестве соотношений баланса влаги используются уравнения переноса водяного пара, облачной и дождевой влаги, рассматриваются процессы испарения воды с поверхности Земли, выпадения дождя, а также

моделируются взаимопереходы одних видов влаги в другие (используется микрофизика для теплого дождя, предложенная в работе [2]).

Расчет радиационного переноса тепла в атмосфере складывается из вычислений коротковолновой (солнечный свет) и длинноволновой (тепловое излучение) составляющих, кроме того, производится учет наличия облачности.

При параметризации взаимодействия атмосферы с поверхностью Земли используются основные соотношения теории подобия Монина–Обухова. На верхней и боковых границах определяются соотношения, выражающие выход метеорологических параметров на некоторые фоновые значения.

2. Метод расчета и параллельная реализация

Для учета неоднородного рельефа подстилающей поверхности при дискретизации исходной системы дифференциальных уравнений используется преобразование координат, переводящее исходную криволинейную область в прямоугольную, причем осуществляется сгущение узлов сетки к поверхности Земли.

Дискретный аналог системы уравнений выводится при помощи метода конечного объема. Разностные уравнения получаются на неразнесенных сетках (в центре конечного объема вычисляются давление, компоненты скорости и прочие характеристики). При аппроксимации конвективных потоков применяются монотонизированные противопотоковые схемы второго порядка [3].

Решение систем алгебраических уравнений, получающихся в результате дискретизации, производится методом Гаусса–Зейделя (для компонент скорости) и явным методом Булеева (для турбулентных и термодинамических характеристик), а для получения поля давления, согласованного с векторным полем скорости, используется алгоритм SIMPLE Патанкара [4].

Распараллеливание вычислительного алгоритма выполняется путем разрезания исходной области по координате x на подобласти, каждая из которых представляет собой «столб» атмосферы над частью поверхности Земли (рис. 1). Выбор такого разбиения связан с необходимостью проведения интегрирования по вертикальной координате при расчете радиационных потоков, что будет менее эффективно в вычислительном плане в случае распределенности данных одного столбца сетки по процессорам.

На рис. 1 представлено распределение данных на примере 4 процессоров. Шаблон конечно-разностной схемы имеет 12 точек, так как при преобразовании координат в уравнениях переноса появляются члены со смешанными производными (4 диагональные точки), а аппроксимация

конвективных членов со вторым порядком требует 5 точек в каждом из координатных направлений.

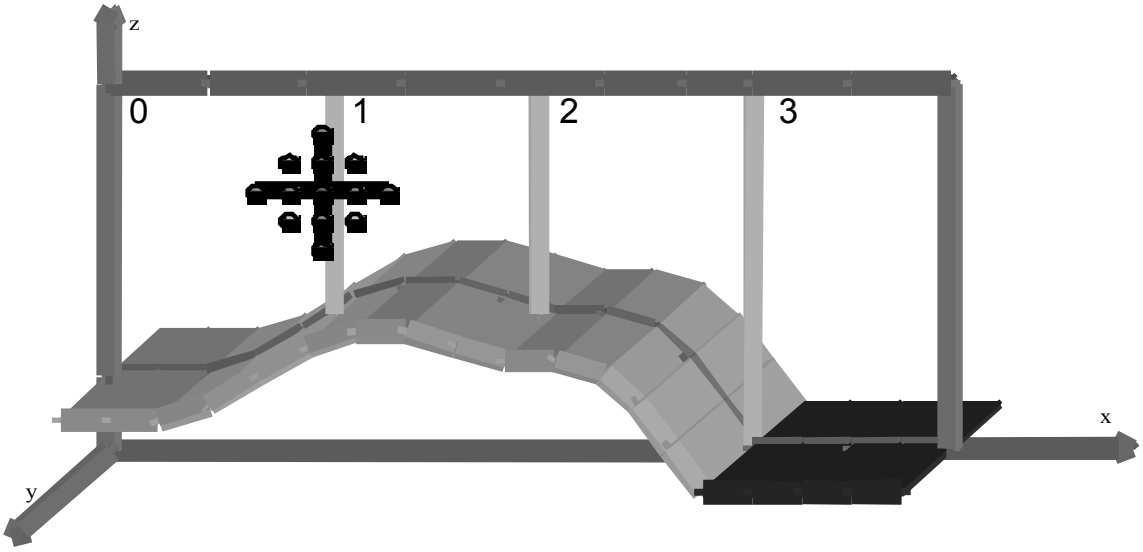


Рис. 1 Распределение данных по процес сорам

Для вычисления приграничного значения, расположенного у границы подобласти “0”, требуются значения, расположенные в подобласти “1”, причем необходимо осуществить пересылку двух столбцов данных. Таким образом, на каждом шаге итерационного алгоритма производится согласование смежных значений путем обмена данными между процессорами.

Адаптация алгоритма к многопроцессорным компьютерам является успешной, если для параллельной версии алгоритма наблюдается существенное ускорение и результаты вычислений для последовательной и параллельной версий алгоритма совпадают в пределах определенной приемлемой точности. Конечно, идеальным случаем можно было бы считать совпадение результатов с точностью до бита, однако на практике нередко наблюдается различие, возникающее вследствие несохранения порядка выполнения арифметических операций и наличия ошибок округления. Естественно, любая параллельная реализация алгоритма не должна существенно влиять на качество получения результата.

Отдельного внимания заслуживает вопрос о согласовании полей скорости и давления в алгоритме SIMPLE [4] при параллельной реализации. Граничные условия для поправки давления, необходимой для коррекции полей скорости, ставятся следующим образом:

$\frac{\partial p'}{\partial n} = 0$ на нижней и боковых границах расчетной области (n – нормаль к границе);

$p' = 0$ на верхней границе.

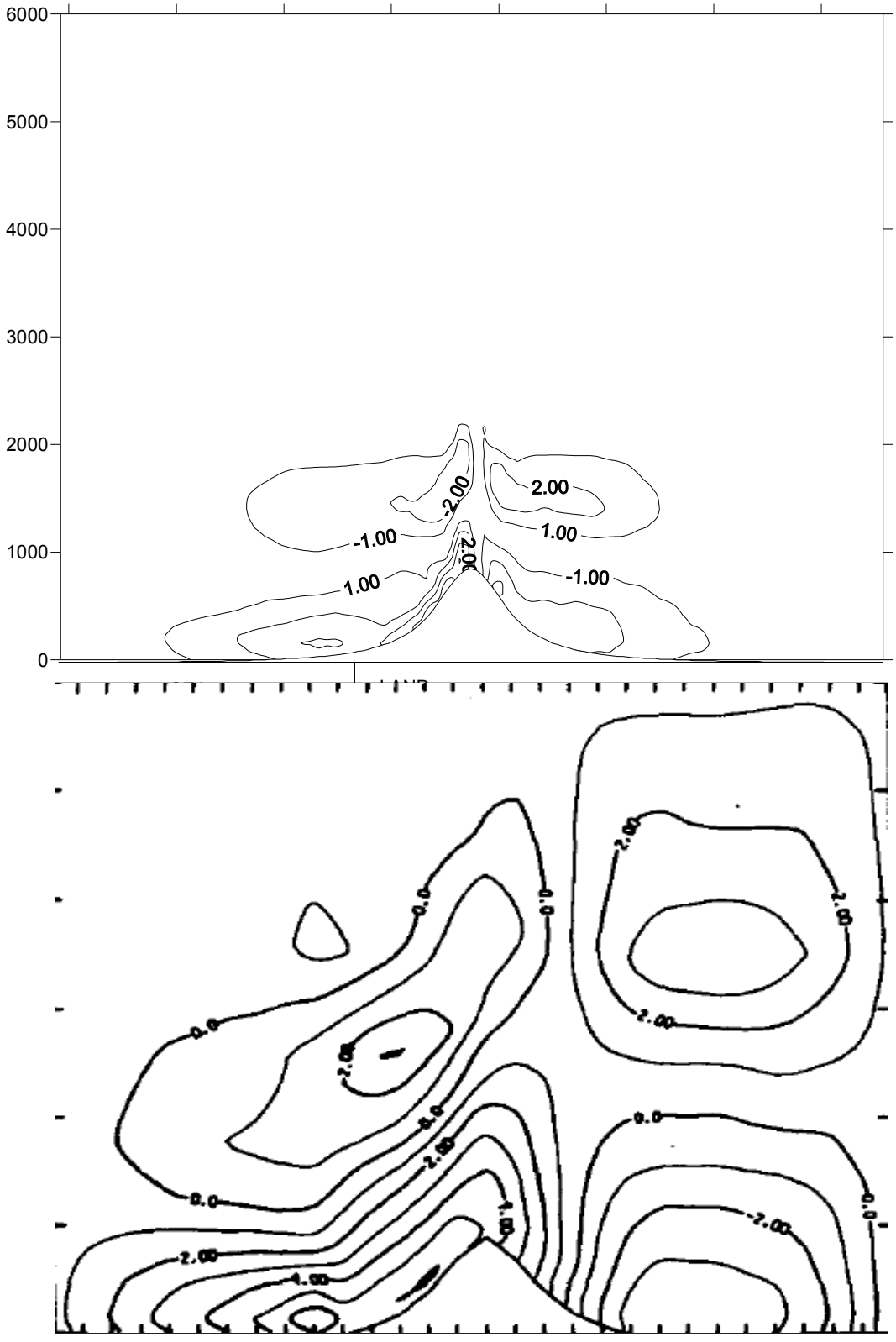


Рис. 2 Распределение горизонтальной компоненты скорости на 16:00
(нижний рисунок из работы [5])

При определении поля поправки давления используется 5-точечный шаблон, поэтому процессорам достаточно обмениваться лишь одним столбцом данных. На границах распределения данных по процессорам

(“разрезах”) ставятся условия первого рода, то есть при вычислении приграничного к “разрезу” узла используется точно такая же вычислительная схема, что и во внутренних узлах, с той лишь разницей, что одно значение должно было быть заранее получено с соседнего процессора. Так как на пересылку данных затрачивается значительное время, целесообразно предварительно произвести расчеты в приграничных узлах, а затем совместить обмен данными и расчеты во внутренних узлах. Одновременный обмен

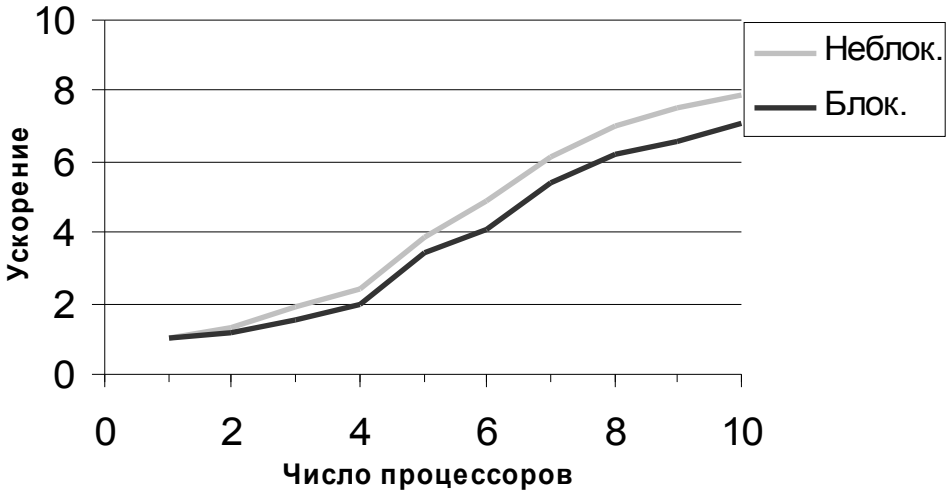


Рис. 3 Ускорение параллельного алгоритма при использовании блокирующих и неблокирующих обменов

данными и проведение расчетов возможно при использовании неблокирующих операций обмена MPI.

В качестве примера применения модели и разработанного алгоритма рассматривалась задача о горно-долинной бризовой циркуляции [5]. Расчетная область представляла собой прямоугольник высотой 6 км и длиной 350 км, в центре располагался холм высотой 900 м с полушириной подошвы 15 км. От левого края расчетной области на 150 км внутрь распространяется водная поверхность. Подробное описание условий проведения вычислительного эксперимента можно найти в работе [5].

Рис. 2 представляет распределение горизонтальной компоненты скорости для одного из моментов времени вычислительного эксперимента. Из рисунка видно, что к 16 часам времени суток для рассматриваемой конфигурации подстилающей поверхности имеет место движение приповерхностного воздуха к вершине холма. Причем наличие обширной водной поверхности усиливает силу бриза, нарушая симметрию распределения компоненты скорости u. В атмосферном пограничном слое на высоте 1500–2000 м поток движется в противоположном направлении, что говорит о существовании крупномасштабных циркуляций воздуха, вызванных

конвективными условиями и неоднородностью свойств подстилающей поверхности.

Несмотря на то, что в данной работе, в отличие от [5], использовалась негидростатическая модель, более современные параметризации турбулентного переноса и расчеты велись на более подробных сетках (39x50), можно отметить, что в вычислительном эксперименте структура течения воспроизводится хорошо на качественном и количественном уровне.

Вычисления производились на кластере ИОА СО РАН. Рис. 3 представляет ускорение параллельного алгоритма в зависимости от числа процессоров. Ускорение почти в 8 раз на 10 процессорах является весьма хорошим результатом.

Литература

1. Starchenko A., Yesaulov A. Some results of numerical simulation of atmospheric aerosol and gaseous component dispersion over an industrial center // Proc. SPIE. – 2004. – Vol.5397. – P. 109–120.

2. Hurley P. The Air Pollution Model (TAPM) Version 2 / CSIRO Atmospheric Research Technical Paper. – 2002. – № 55.

3. Есаулов А.О., Старченко А.В. К выбору схемы для численного решения уравнений переноса // Вычислительная гидродинамика. – Томск: Изд-во Том. ун-та, 1999. – С. 27–32.

4. Патанкар С. Численные методы решения задач теплообмена и динамики жидкости. – М.: Энергоатомиздат, 1984.

5. Mahrer Y., Pielke R. The Effects of Topography on See and Land Breezes in a Two-Dimensional Numerical Model // Mountly Weather Review. – 1977. – Vol.105. – P. 1151–1162.

СОДЕРЖАНИЕ

ПЛЕНАРНЫЕ ДОКЛАДЫ

Воеводин Вл.В. Распределенная обработка данных……3
Сидоренко Ю.Н., Тимченко С.В. Параллельные методы решения задач глобальной оптимизации: генетические алгоритмы и эволюционные стратегии……10
Старченко А.В. Параллельные вычисления в задачах охраны окружающей среды……...17

ДОКЛАДЫ ПО МЕТОДАМ И ТЕХНОЛОГИЯМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Берцун В.Н. Параллельные алгоритмы сплайнов…..……23
Вакилов А.Н., Прудников В.В., Филиканов Е.Л. Библиотека параллельных программ для моделирования критического поведения макроскопических неупорядоченных систем……..26
Казаков В.Ю., Тимченко С.В. Параллельный алгоритм для решения задач многокритериальной оптимизации……..32
Миньков Л.Л., Пикущак Е.В. Оценка эффективности распараллеливания явных разностных схем для решения уравнений газовой динамики……...38
Подольский В.Е., Толстых С.С. Оптимизация кластерных вычислений с использованием критериев структурной сложности…….45
Снытников А.В. Динамическая балансировка загрузки в задаче моделирования динамики протопланетного диска…….51
Тимошевская Н.Е. Параллельная генерация сочетаний и перестановок….55
Богословский Н.Н., Старченко А.В. Параллельная реализация одного метода решения уравнений Навье-Стокса……..60
Гисс С. С., Тимошевская Н.Е. Распараллеливание алгоритма решения задачи о выполнимости КНФ……..……65
Киреев С.Е. Эффективное программирование узлов многопроцессорной вычислительной системы…….……67
Кузнецова А.В. Параллельный алгоритм проверки числа на простоту…...72
Нечаев С. П. Алгоритм сборки минимальных вычислительных фрагментов в сборочной технологии параллельного программирования…..……..74
Нечаева О.И. Последовательный и параллельный алгоритмы кластеризации символьных последовательностей……..……80
Худяшов И. И. Распараллеливание алгоритма факторизации «Квадратичное решето»……86

Черкасов А.В. Использование виртуальных топологий на мультикомпьютерах……88
Лаева В.И., Чижухина М.А. Параллельная реализация метода полной редукции……. 93
Старченко А.В., Берцун В.Н. Обучение параллельным компьютерным технологиям студентов ММФ ТГУ……..……..97
Есаулов А.О. Использование параллельных вычислений при моделировании мезомасштабных атмосферных процессов……...100

Научное издание

Вторая Сибирская школа-семинар
по параллельным вычислениям

Томск, 16-19 декабря 2003 года

Редактор **Е.В.Лукина**
Технический редактор **Р.М.Подгорбунская**
Компьютерная верстка **А.О.Есаулов**

Лицензия ИД 04617 от 24.04.01 г. Подписано в печать 11.05.04 г.
Формат 60x84 $^{1}/_{16}$. Бумага офсетная №1.
Печать офсетная. Гарнитура « ».
Печ. л. 6,75; уч.-изд. л. 6,00; усл. печ. л. 6,3.

ФГУП «Издательство ТГУ», 634029, г. Томск, ул. Никитина, 4
Типография «Иван Федоров», 634003, г. Томск, Октябрьский взвоз, 3