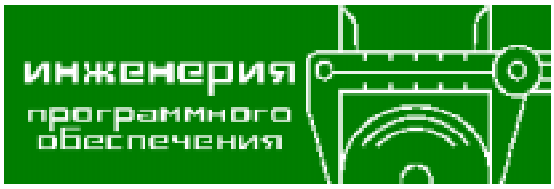


Демьянович Юрий Казимирович

О ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЯХ [1]

Имеется большое количество важнейших задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем. К таким задачам прежде всего относятся задачи точных долгосрочных прогнозов климатических изменений и геологических катаклизмов (землетрясений, извержений вулканов, столкновений тектонических плит), прогнозов цунами и разрушительных ураганов, а также экологических прогнозов и т. п. Сюда следует отнести также прогнозирование результатов экспериментов во многих разделах физики, в особенности экспериментов по выявлению основ мироздания (экспериментов на коллайдерах со встречными пучками частиц, экспериментов, направленных на получение антиматерии и так называемой темной материи, и т. д.). Важными задачами являются расшифровка генома человека, определение роли каждого гена в организме, влияние генов на здоровье человека и на продолжительность жизни. Не решена задача безопасного хранения вооружений, в особенности ядерного оружия (из-за запрета на ядерные испытания состояние накопленных ядерных зарядов можно определить лишь путем моделирования на компьютере большой мощности).

Постоянно появляются новые задачи подобного рода, и возрастают требования к точности и к скорости решения прежних задач, поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем. К сожалению, технологические возможности увеличения быстродействия процессоров ограничены: увеличение быстродействия связано с уменьшением размеров процессоров, а при малых размерах появляются трудности из-за квантово-механических эффектов, вносящих элементы недетерминированности; эти трудности пока что не удается преодолеть. Из-за этого приходится идти по пути создания параллельных вычислительных систем, то есть систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с решением одной задачи.

На современном этапе развития вычислительной техники такой способ, по-видимому, является одним из основных способов ускорения вычислений.

Первоначально идея распараллеливания вычислительного процесса возникла в связи с необходимостью ускорить вычисления для решения сложных задач при использовании имеющейся элементной базы. Предполагалось, что вычислительные модули (процессоры или компьютеры) можно соединить между собой так, чтобы решение задач на полученной вычислительной системе ускорялось во столько раз, сколько использовано в ней вычислительных модулей. Однако достаточно быстро стало ясно, что для сложных задач такое ускорение, как правило, достичь невозможно по двум причинам:

[1] Работа частично поддержана грантами РФФИ 07-01-00451 и 07-01-00269.

1) любая задача распараллеливается лишь частично (всегда имеются части, которые невозможно распараллелить),

2) коммуникационная среда, связывающая отдельные части параллельной системы, работает значительно медленнее процессоров, так что передача информации существенно задерживает вычисления.

В данной статье рассмотрены: постановка задачи распараллеливания, понятия алгоритма и его параллельной формы, концепция неограниченного параллелизма и схема сдваивания; приведены примеры, иллюстрирующие излагаемый материал. Конечно, здесь не удастся изложить основы программирования на параллельных вычислительных системах – это предполагается сделать впоследствии.

Читателям, желающим глубоко изучить проблемы параллельных вычислений, рекомендуются книги [1–4], а также курсы лекций [5–7].

1. ПОСТАНОВКА ЗАДАЧИ РАСПАРАЛЛЕЛИВАНИЯ

Многие явления природы характеризуются параллелизмом (одновременным исполнением процессов с применением различных путей и способов). В частности, в живой природе параллелизм распространен очень широко в дублирующих системах для получения надежного результата.

Параллельные процессы пронизывают общественные отношения, ими характеризуются развитие науки, культуры и экономики в человеческом обществе. Среди этих процессов особую роль играют параллельные информационные потоки. Среди них можно упомянуть потоки информации со спутников, от различных источников излучения во Вселенной, циркуляцию информации в человеческом обществе и т. п. При проведении вычислений обычными стали многозадачность и мультипрограммность, мультимедийные средства, компьютерные локальные сети, а также глобальные сети, такие как Интернет, сеть WEB и т. п. Это показывает, что серьезное изучение вопросов распараллеливания и высокопроизводительных вычислений чрезвычайно важно.

Последние годы характеризуются скачкообразным прогрессом в развитии микроэлектроники, что ведет к постоянному совершенствованию вычислительной техники. Появилось большое количество вычислительных систем с разнообразной архитектурой, исследованы многие варианты их использования при решении возникающих задач.

Среди параллельных систем различают конвейерные, векторные, матричные, систолические, спецпроцессоры и т. п. Родоначальниками параллельных систем являются ILLIAC, CRAY и CONVEX. В настоящее время все суперкомпьютеры являются параллельными системами.

Суперкомпьютеры каждого типа создаются в небольшом количестве экземпляров. Обычно каждый тип суперкомпьютеров имеет определенные неповторимые архитектурные, технологические и вычислительные характеристики, поэтому сравнение суперкомпьютеров – весьма сложная задача, не имеющая однозначного решения. Тем не менее, разработаны определенные принципы условного сравнения компьютеров (это важно для их дальнейшего совершенствования и для продвижения на рынке). В соответствии с этими принципами, суперкомпьютеры классифицируются в регулярно обновляемом списке TOP500, который размещен в Интернете по адресу www.top500.org. Этот список содержит 500 типов компьютеров, расположенных в порядке убывания мощности; в списке ука-

...параллельные информационные потоки...

зывается порядковый номер суперкомпьютера, организация, где он установлен, его название и производитель, количество процессоров, максимальная реальная производительность (на пакете LINPACK), пиковая (то есть теоретическая) производительность. Так, например, в 29-й редакции списка TOP500 (появившейся 27 июня 2007 года) на первом месте находится суперкомпьютер BlueGene/L фирмы IBM с числом процессоров 131072, максимальной реальной производительностью 280.6 триллионов операций с плавающей точкой в секунду (краткая запись: 280.6 Tflops [1]) и с пиковой производительностью 367 Tflops; два компьютера в этом списке имеют производительность, превосходящую 100 Tflops: это компьютеры Cray XT4/XT3 и Cray Red Storm (они занимают 2-е и 3-е места, соответственно).

С появлением параллельных систем возникли новые проблемы:

– как обеспечить эффективное решение задач на той или иной параллельной системе и какими критериями эффективности следует пользоваться;

– как описать класс задач, которые естественно решать на данной параллельной системе, а также класс задач, не поддающихся эффективному распараллеливанию;

– как обеспечить преобразование данного алгоритма в подходящую для рассматриваемой параллельной системы форму (то есть как распараллелить алгоритм);

– как поддержать переносимость полученной программы на систему с другой архитектурой;

– как сохранить работоспособность программы и улучшить ее характеристики при модификации данной системы; в частности, как обеспечить работоспособность программы при увеличении количества параллельных модулей.

Естественным способом решения этих проблем стало создание стандартов как для вычислительной техники (и прежде всего – для элементной базы), так и для программного обеспечения. В настоящее время разрабатываются стандарты для математического обеспечения параллельных вычислительных систем; в частности, постоянно дорабатываются стандарты MPI и Open MP, а также некоторые другие.

...задача о компьютерном моделировании...

2. ПРИМЕР ЗАДАЧИ, РЕШАЕМОЙ НА СУПЕРКОМПЬЮТЕРАХ

Характерным (и типичным) примером сложной вычислительной задачи является задача о компьютерном моделировании климата (в частности, задача о метеорологическом прогнозе). Климатическая задача включает в себя атмосферу, океан, сушу, криосферу и биоту (биологическую составляющую). Климатом называется ансамбль состояний, который система проходит за большой промежуток времени.

Под климатической моделью подразумевается математическая модель, описывающая климатическую систему с той или иной степенью точности.

В основе климатической модели лежат уравнения сплошной среды и уравнения равновесной термодинамики. В модели описываются многие физические процессы, связанные с переносом энергии: перенос излучения в атмосфере, фазовые переходы воды, мелкомасштабная турбулентная диффузия

[1] Tflops – один триллион операций с плавающей точкой в секунду; читается «терафлопс».

тепла, диссипация кинетической энергии, образование облаков, конвекция и др.

Рассматриваемая модель представляет собой систему нелинейных уравнений в частных производных в трехмерном пространстве. Ее решение воспроизводит все главные характеристики ансамбля состояний климатической системы.

Если обозначить

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \bar{v}\nabla = \frac{\partial}{\partial t} + v_x \frac{\partial}{\partial x} + \dots,$$

то простейшая система уравнений, моделирующая погоду (решается в приземном сферическом слое Σ, то есть в тропосфере, окружающей Землю), включает следующие уравнения:

– уравнение количества движения

$$\frac{D\bar{v}}{\Delta t} = -\frac{1}{\rho}\nabla p + g - 2\bar{\Omega} \times \bar{v},$$

где p – давление, ρ – плотность, g – ускорение силы тяжести, $\bar{\Omega}$ – угловой вектор скорости вращения Земли, $\bar{v}$ – скорость ветра;

– уравнение сохранения энергии

$$c_p \frac{DT}{Dt} = \frac{1}{p}\frac{Dp}{Dt},$$

где c_p – удельная теплоемкость, T – температура;

– уравнение неразрывности (уравнения сохранения массы)

$$\frac{Dp}{Dt} = -\rho \operatorname{div} \bar{v};$$

– уравнение состояния

$$p = \rho RT,$$

где R – константа.

Фактически перед нами система шести нелинейных скалярных уравнений относительно шести неизвестных функций (зависящих от трех координат $(x, y, z) \in \Sigma$ и времени t), а именно, относительно компонент v_x, v_y, v_z вектора скорости $\bar{v}$ и функций p, ρ, T. К этим уравнениям присоединяются начальные и граничные условия; полученная система уравнений представляет собой математическую модель погоды. Заметим, что климатическая модель намного сложнее. Работая с ней, приходится принимать во внимание следующее:

– в отличие от многих других наук при исследовании климата нельзя поставить глобальный натурный эксперимент;

– проведение численных экспериментов над моделями и сравнение результатов экспериментов с результатами наблюдений единственная возможность изучения климата;

– сложность моделирования заключается в том, что климатическая модель включает в себя ряд моделей, которые разработаны неодинаково глубоко; при этом лучше всего разработана модель атмосферы, поскольку наблюдения за ее состоянием ведутся давно и, следовательно, имеется много эмпирических данных;

– общая модель климата далека от завершения, поэтому в исследования включают обычно лишь моделирование состояния атмосферы и моделирование состояния океана.

Рассмотрим вычислительную сложность обработки модели состояния атмосферы. Предположим, что нас интересует развитие атмосферных процессов на протяжении 100 лет.

При построении вычислительных алгоритмов используем принцип дискретизации: вся атмосфера разбивается на отдельные элементы (параллелепипеды) с помощью сетки с шагом 1° по широте и по долготе; по

...и сравнение результатов экспериментов с результатами наблюдений...

высоте берут 40 слоев. Таким образом получается $2{,}6 \cdot 10^6$ элементов. Каждый элемент описывается десятью компонентами. В фиксированный момент времени состояние атмосферы характеризуется ансамблем из $2{,}6 \cdot 10^7$ чисел. Условия развития процессов в атмосфере требуют каждые 10 минут находить новый ансамбль, так что за 100 лет будем иметь $5{,}3 \cdot 10^6$ ансамблей. Таким образом, в течение одного численного эксперимента получим около $2{,}6 \times 5{,}3 \cdot 10^6 \approx \approx 1{,}4 \cdot 10^{14}$ числовых результатов. Если учесть, что для получения одного числового результата требуется 10^2–10^3 арифметических операций, то приходим к выводу, что для одного варианта вычисления модели состояния атмосферы на интервале 100 лет требуется затратить 10^{16}–10^{17} арифметических действий с плавающей точкой. Следовательно, вычислительная система с производительностью 10^{12} операций в секунду при полной загрузке и эффективной программе будет работать 10^4–10^5 секунд (иначе говоря, потребуется от 3 до 30 часов вычислений). Ввиду отсутствия точной информации о начальных и о краевых условиях, требуется просчитать сотни подобных вариантов.

Заметим, что расчет полной климатической модели займет на порядок больше времени.

3. СТОИТ ЛИ ПРОВОДИТЬ ЧИСЛЕННЫЙ ЭКСПЕРИМЕНТ?

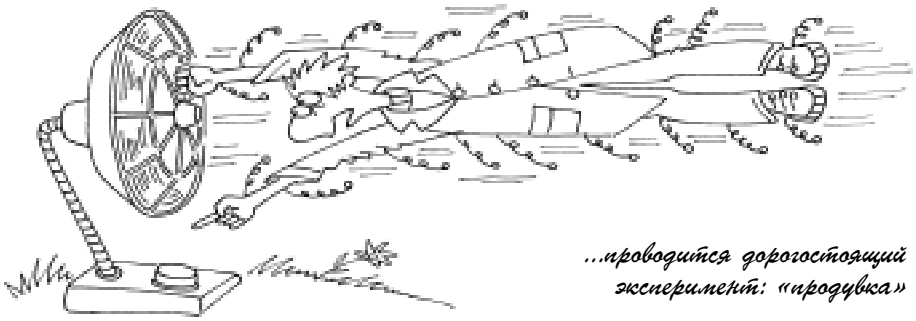
Предыдущий пример показывает, что высокопроизводительные вычислительные системы необходимы для прогноза погоды, а также для прогнозирования климатических изменений. Нетрудно понять, что задачи прогноза землетрясений, цунами, извержений и других природных катаклизмов требуют решения не менее сложных математических задач. Еще сложнее задачи высоконадежных вычислений, связанных с исследованиями космоса, с экспериментами на субатомном уровне (постройка ускорителей элементарных частиц, ядерных реакторов), с испытанием и хранением ядерного оружия и др.

Высокопроизводительная вычислительная система имеет большую стоимость; ее создание и эксплуатация требуют обучения большого числа специалистов. Создание математического обеспечения и программ для такой системы – весьма трудоемкая задача. С другой стороны, многие задачи допускают постановку натурного эксперимента, что в ряде случаев быстрее приводит к цели, чем проведение численного эксперимента, хотя стоимость натурного эксперимента может оказаться очень большой.

Какое же количество вычислительных систем на самом деле целесообразно иметь? Ответ зависит от конкретной ситуации.

Например, до запрещения испытаний ядерного оружия был возможен натурный эксперимент. После запрещения испытаний он стал невозможен, так что способы надежного хранения и совершенствования ядерного оружия определяются исключительно численным экспериментом, для проведения которого нужны мощнейшие компьютеры, соответствующие программные разработки, штат специалистов и т. д.

Существует множество областей, в которых невозможно или трудно проводить натурный эксперимент: экономика, экология, астрофизика, медицина; однако во всех этих областях часто возникают большие вычислительные задачи.



...проводится дорогостоящий натурный эксперимент: «продувка» объектов...

В некоторых областях, таких как аэродинамика, часто проводится дорогостоящий натурный эксперимент: «продувка» объектов (самолетов, ракет и т. п.) в аэродинамической трубе. В начале прошлого века «продувка» самолета братьев Райт стоила более 10 тысяч долларов, «продувка» многоразового корабля «Шаттл» стоит 100 миллионов долларов.

Однако, как оказалось, «продувка» не дает полной картины обтекания объекта, так как не удается установить датчики во всех интересующих точках: в некоторых случаях нельзя установить датчик из-за его размера, в других случаях установка датчика может исказить картину обтекания.

Для преодоления этих трудностей приходится создавать математическую модель и проводить численный эксперимент, который обходится не дешево, но все же значительно дешевле, чем натурный эксперимент.

Типичная ситуация состоит в следующем:

– исследуемые объекты являются трехмерными;

– для приемлемой точности приходится использовать сетку с одним миллионом узлов;

– в каждом узле необходимо найти числовые значения от 5 до 20 функций;

– при изучении нестационарного поведения объекта нужно определить его состояние в 10^2–10^4 моментах времени;

– на вычисление каждого значимого результата в среднем приходится 10^2–10^3 арифметических действий;

– вычисления могут циклически повторяться для уточнения результата.

Этапы численного эксперимента изображены на рис. 1.

Замечание 1. Если хотя бы один из этапов выполняется неэффективно, то неэффективным будет и весь численный эксперимент (и проводить его, по-видимому, нецелесообразно).

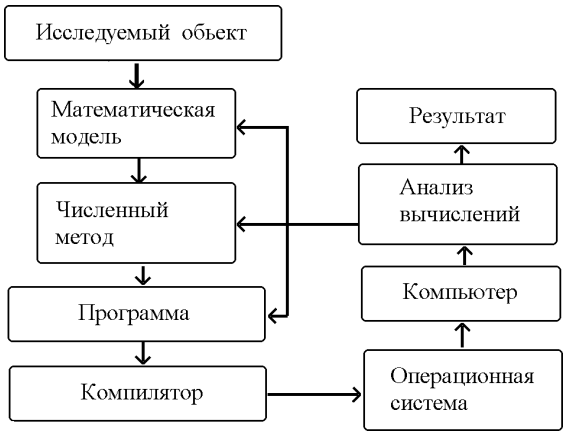


Рис. 1. Этапы численного эксперимента

4. ЧТО ТАКОЕ АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ?

4.1. ОДНОПРОЦЕССОРНЫЕ СИСТЕМЫ

В архитектуре однопроцессорных вычислительных систем (ВС) принято различать следующие устройства:

– устройства управления (УУ),

– центральный процессор (ЦП),

– память,

– устройство ввода-вывода (В/В),

– каналы обмена информацией.

Взаимодействие этих устройств показано на рис. 2.

Принцип работы однопроцессорной ВС состоит в последовательном выполнении операций, так что главной задачей при программировании для однопроцессорной ВС является представление алгоритма в виде последовательности операций, а основная проблема оптимизации сводится к минимизации числа операций и к уменьшению размера требуемой памяти.

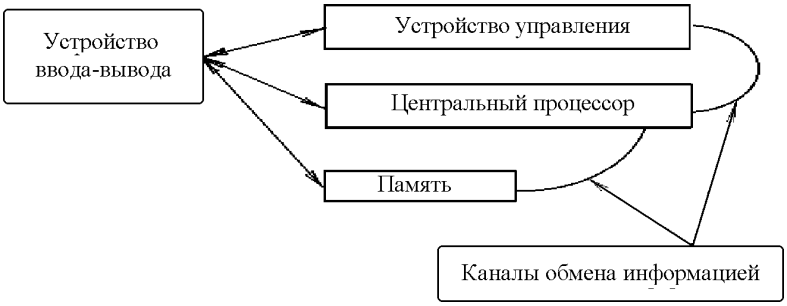


Рис. 2. Схема однопроцессорной ВС

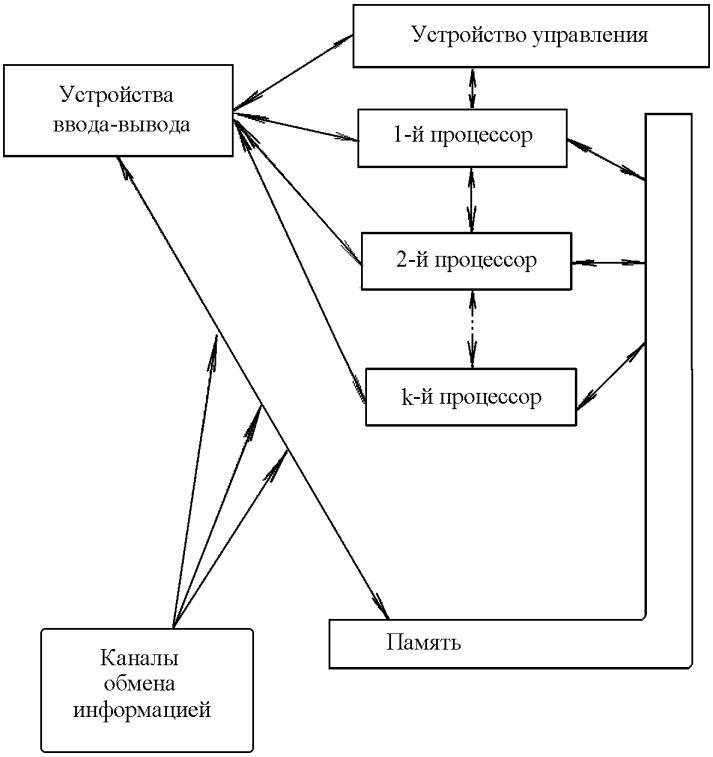


Рис. 3. Схема многопроцессорной ВС

4.2. МНОГОПРОЦЕССОРНЫЕ СИСТЕМЫ

Многопроцессорные системы формально имеют сходную структуру (рис. 3):

– устройство управления;
– первый процессор;
– второй процессор;
...
– *k*-й процессор;
– память (общую или разделенную);
– устройство ввода-вывода;
– каналы обмена информацией.

Узкое место такой системы – коммуникационная сеть (каналы обмена информацией). Сложность сети обычно растет пропорционально квадрату числа имеющихся устройств. В настоящее время невозможно создать сеть с эффективной связью между любыми двумя устройствами многопроцессорной ВС. Возможности коммуникационной сети существенно ограничивают класс рассматриваемых задач.

4.3. ИДЕЯ КОНВЕЙЕРНЫХ ВЫЧИСЛЕНИЙ

Параллельное исполнение команд широко используется и в однопроцессорных системах для ускорения процесса вычислений; многие устройства (принтер, видеоадаптер и т. п.) работают автономно в то время, пока процессор проводит дальнейшие вычисления. Одним из методов распараллеливания является конвейеризация вычислений. Остановимся на нем подробнее.

При поступлении потока задач каждая из них может расщепляться на последовательность подзадач с тем, чтобы любая такая подзадача реализовывалась на части вычислительной системы. Это позволяет эффективнее использовать имеющееся оборудование, уменьшая его простои и частично совмещая решение упомянутых подзадач.

Вычислительная система, предназначенная для такого использования, называется *конвейерной*, а процесс подобных вычислений – *конвейером*.

В конвейере различают r последовательных этапов, так что, когда i-я операция проходит s-й этап, то $(i + k)$-я операция проходит $(s – k)$-й этап (рис. 4).

Таким образом, имеется две тенденции:

– использование многих устройств одновременно для однотипных операций (это называется *распараллеливанием*);

– повышение загруженности каждого устройства использованием различных его

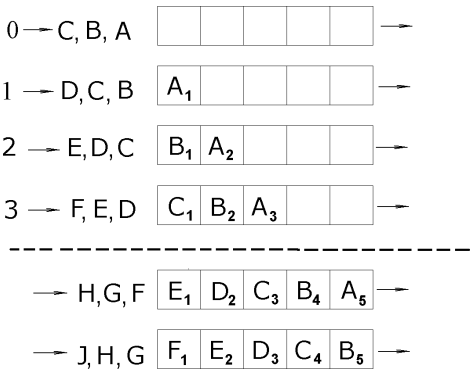


Рис. 4. Схема конвейера

частей для одновременного проведения разнотипных операций (это называется *конвейеризацией*).

Конечно, конвейеризация может сыграть существенную роль в уменьшении числа необходимых арифметических устройств и тем самым значительно повлиять на увеличение эффективности.

4.4. ПРИМЕРЫ ПАРАЛЛЕЛЬНЫХ ВС

Вот некоторые примеры параллельных систем.

1. Система ILLIAC-IV (1974 г.) имела 64 процессора, выполняла 100–200 млн оп./сек с 64-разрядными словами. Память этой системы распределенная (у каждого процессора 2048 слов).

Конструктивно система содержала матрицу процессоров 8×8, команды выполнялись синхронно, и коммуникациями служили быстрые каналы, связывающие каждый процессор с четырьмя соседними процессорами.

2. В системе CRAY-1 (1976 г.) был использован конвейерный принцип. Система выполняла 80140 миллионов операций в секунду с 64-разрядными словами и имела 12 функциональных конвейерных устройств. Режим работы – синхронный. Здесь имелось 8 векторных регистров по 64 слова, а также быстрые регистры, позволяющие быстро перестраивать систему конвейеров в разнообразные цепочки с передачей данных через эти быстрые регистры.

3. Система Earth-Simulator (фирма NEC, 2002 г.) – одна из последних разработок среди суперкомпьютеров (она имеет 5120 параллельных процессоров). Эта система достигла пиковой производительности 40,9 триллиона операций (с плавающей точкой) в секунду (кратко 40,9 Терафлопс), а также производительности 35,8 Терафлопс на реальных задачах линейной алгебры (из пакета LINPACK) и была мощнейшей в мире системой до 2005 года.

4. С 2005 года наиболее мощной является система BlueGene/L фирмы IBM; в дополнение к приведенным в первом пункте сведениям отметим, что ее общая память составляет 32 триллиона байтов, потребляемая

Параллельное исполнение команд широко используется...

мощность 1,5 мегаватта, занимаемая площадь 2500 квадратных футов (около 250 м^2).

5. О ПОНЯТИИ АЛГОРИТМА

Понятие алгоритма, используемое в теории алгоритмов, зачастую не адекватно понятию «алгоритм», которое используется в теории и практике программирования.

Математическая энциклопедия (МЭ) так определяет это понятие. «Алгоритм – точное предписание, которое задает вычислительный процесс (называемый в этом случае алгоритимическим), начинающийся с произвольного исходного данного (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этим исходным данным результата».

На этом статья, посвященная определению алгоритма, не кончается: каждый может ознакомиться с ее полным текстом (см. МЭ, т. 1, 1977 г., с. 202). Здесь слова «вычислительный процесс» не означают обязательно операции с числами это может быть работа с любыми четко определенными объектами по заданным правилам.

На с. 206 этой же энциклопедии дается понятие «Алгоритм в алфавите А»: это – «точное общепонятное предписание, определяющее потенциально осуществимый процесс последовательного преобразования слов в алфавите А, процесс, допускающий любое слово в алфавите А в качестве исходного». Очевидно, понятие «алгоритм в ал-

фавите А» – более узкое понятие, чем понятие алгоритма, отмеченное выше (то есть это частный случай более общего понятия алгоритма).

В конечном счете вычислительная система дискретного действия обрабатывает двоичные коды, и потому можно считать, что алфавит А состоит из двух символов 0 и 1; с этой точки зрения любое моделирование на упомянутой системе можно рассматривать как численное решение (с той или иной точностью) поставленной задачи. На первый взгляд представляется правильным сначала «разработать алгоритм решения» поставленной задачи, затем его запрограммировать и реализовать на имеющейся вычислительной системе. Однако, давно замечено, что понятие алгоритма для этих целей не достаточно: даже в простейших случаях алгоритм определяется лишь в момент реализации. Приведем пример, иллюстрирующий возникшую ситуацию.

Рассмотрим задачу об отыскании квадратного корня из числа $a > 0$.

Будем использовать соотношение

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n} \right), \; x_0 > 0, \qquad (5.1)$$

до тех пор пока $|x_{n+1} - x_n| < \varepsilon$, где a, ε, x_0 – заданные положительные числа.

Является ли это соотношение описанием алгоритма? Поскольку его можно реализовать в виде программы вычислений на компьютере, то можно предположить, что это действительно описание алгоритма.

Однако начальное значение $x_0 > 0$ нельзя брать произвольным (так, например, при программировании на языке Си или на Паскале нельзя брать x_0 иррациональным числом); при вычислениях, возможно, придется использовать «машинную арифметику» с присущими ей правилами округления и учитывать другие ограничения (на первый взгляд кажется, что использование обыкновенных дробей позволило бы отказаться от округления, но это привело бы к быстрому исчерпанию ресурсов по памяти и быстродействию).

Таким образом, результат будет зависеть от свойств используемой вычислительной системы (разрядности, правил округления и т. п.) и программного окружения; однако, упомянутые свойства (во всем многообразии деталей), как правило, пользователю не известны, так что «разработать алгоритм решения» поставленной задачи, строго говоря, не удастся.

В примере (5.1) указанные действия не имеют четкого конструктивного определения. При изучении вопросов распараллеливания подобная неопределенность значительно возрастает, и с этим приходится мириться, но при гигантской сложности современных задач эта неопределенность требует повышенной устойчивости параллельных версий алгоритмов.

Назовем *псевдоалгоритмом* (с указанной областью определения) однозначно понимаемую совокупность указаний о выполняемых действиях, считая, что сами действия не обязательно имеют четкое конструктивное определение. При уточнении определения упомянутых действий исходный псевдоалгоритм можно заменить новым, называемым уточнением предыдущего. В результате получается цепочка уточняющих псевдоалгоритмов, последним звеном которой является алгоритм реализации вычислений. Учет возможных изменений свойств используемой вычислительной системы (или ее замены) и программного окружения приводит к дереву псевдоалгоритмов, листьями которого служат различные алгоритмы реализации, связанные с исходным псевдоалгоритмом – корнем этого дерева.

Заметим, что в рассматриваемой области исследований обычно термин *алгоритм* используют как эквивалент термина *псевдоалгоритм.*

6. О «МАШИННОЙ АРИФМЕТИКЕ». ЭФФЕКТ «ПРОПАДАНИЯ ЗНАЧАЩИХ ЦИФР»

При решении сложных вычислительных задач используются вещественные числа. Однако, множество вещественных чисел бесконечно, а разрядная сетка любой ЭВМ конечна; поэтому как представления вещественных чисел, так и арифметические дей-

ствия над ними в ЭВМ, как правило, содержат погрешность, называемую ошибкой округления. Совокупность правил, по которым обрабатываются числа в ЭВМ, называется «машинной арифметикой». Упомянутые правила различны для разнотипных ЭВМ; чаще всего используется так называемая «арифметика с плавающей точкой», при этом число a в ЭВМ представляется в виде числа в двоичной системе счисления (в нижеследующем представлении надчеркивание служит для указания того, что стоящие рядом символы a_1, a_2, a_3, ..., a_k разряды числа, а не сомножители):

$$\pm \overline{0.a_1a_2a_3...a_k} \cdot 2^m, \ |m| \le L; \qquad (6.1)$$

здесь k и L – фиксированные натуральные числа. Разряды a_i принимают значения 0 или 1, $i = 1, 2, ..., k$, а их совокупность, отмеченная фигурной скобкой, называется *мантиссой* числа a. Сомножитель 2^m расширяет множество чисел, представимых в виде (6.1); показатель m называется *порядком* числа a. Не останавливаясь подробнее на «машинной арифметике», покажем, что в некоторых случаях она может привести к катастрофическим результатам.

Предположим, что в «арифметике с плавающей точкой» (см. (6.1)) требуется вычислить выражение

$$x = \frac{a+b+c}{d}, \qquad (6.2)$$

где $a = 1$, $b = 2^{-k}$, $c = -1$, $d = 2^{-k}$.

Очевидно, что $x = 1$ – правильный результат вычислений.

В ЭВМ числа a, b, c, d представлены абсолютно точно:

$$a = +0.\underbrace{100...0}_{k} \cdot 2^{+1}, \quad b = +0.\underbrace{100...0}_{k} \cdot 2^{-k+1},$$

$$c = -0.\underbrace{100...0}_{k} \cdot 2^{+1}, \quad d = +0.\underbrace{100...0}_{k} \cdot 2^{-k+1}.$$

Для сложения «в столбик» складываемые числа следует привести к одному порядку. ЭВМ приводит числа к наибольшему порядку:

$$a + b = +0.\underbrace{100...0}_{k} \cdot 2^{+1} + 0.\underbrace{100...0}_{k} \cdot 2^{-k+1} =$$

$$= +0.\underbrace{100...0}_{k} \cdot 2^{+1} + 0.\underbrace{000...01}_{k} \cdot 2^{+1} \overset{\text{ЭВМ}}{=\!=\!=\!=\!=} >$$

$$\overset{\text{ЭВМ}}{=\!=\!=\!=\!=} > +0.\underbrace{100...0}_{k} \cdot 2^{+1}$$

Мантисса второго слагаемого не умещается в предписанной разрядной сетке (см. (6.1)) и единица последнего разряда упомянутой мантиссы выпадает. В результате сложение чисел a и b дает ***неверный*** результат.

Легко видеть, что остальные действия в (6.2) ЭВМ производит точно.

Итак, при трех арифметических действиях ЭВМ получает 0 вместо 1:

$$x = \frac{a+b+c}{d} \overset{\text{ЭВМ}}{=\!=\!=\!=\!=} > 0 \quad \textbf{\textit{Ужасно!}}$$

Очевидно, что ни увеличение k, ни увеличение L не влияют на полученный неверный результат.

С другой стороны, изменение порядка вычислений, при котором к числу a сначала добавляется число c, а к полученной сумме добавляется число b, приводит к верному результату вычислений на ЭВМ. Действительно, добавление к a числа c дает 0, ибо дело сводится к вычислению

$$a + c = +0.\underbrace{100...0}_{k} \cdot 2^{+1} - 0.\underbrace{100...01}_{k} \cdot 2^{+1},$$

которое не требует преобразований, связанных с приведением к одному порядку, поэтому лишь вычитаются мантиссы, и в этом действии ЭВМ получает 0. Дальнейшее очевидно: после добавления нуля к числу b остается разделить число $0.\underbrace{100...01}_{k} \cdot 2^{-k}$ на точно такое же число d, что, конечно, даст единицу. Итак, имеем

$$x = \frac{a+b+c}{d} \overset{\text{ЭВМ}}{=\!=\!=\!=\!=} > 1. \quad \textbf{\textit{Прекрасно!}}$$

Таким образом, при наличии ошибок округления изменение порядка арифметических действий может коренным образом изменить результат вычислений на ЭВМ. К сожалению, программирование ведется с использованием формул, подобных формуле (6.2); для такой формулы, как правило, не известны значения входящих в нее переменных a, b, c, d, и поэтому нет возможности заранее выбрать тот или иной порядок

арифметических действий. Требуется глубокий анализ алгоритмов для того, чтобы уменьшить влияние подобных ситуаций на результаты вычислений.

7. ПАРАЛЛЕЛЬНАЯ ФОРМА АЛГОРИТМА

Для реализации алгоритма на параллельной системе его следует представить в виде *последовательности групп операций*, причем операции в каждой группе можно выполнять одновременно на имеющихся в системе функциональных устройствах.

Пусть операции алгоритма разбиты на группы, а множество групп полностью упорядочено, так что каждая операция любой группы зависит либо от начальных данных, либо от результатов выполнения операций, находящихся в предыдущих группах.

Представление алгоритма в таком виде называется *параллельной формой* алгоритма. Каждая группа операций называется *ярусом*, а число таких ярусов – *высотой* параллельной формы. Максимальное число операций в ярусах – *шириной* параллельной формы.

Один и тот же алгоритм может иметь много параллельных форм. Формы минимальной высоты называются *максимальными.*

Рассмотрим следующие примеры.

Пример 1. Пусть требуется вычислить выражение

$$(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8).$$

...4 процессора загружены только на первом ярусе, а на последнем ярусе загружен лишь один процессор...

Фактически здесь можно проводить вычисления, используя различные параллельные формы, приводящие к одинаковым результатам.

1.1. Одна из параллельных форм такова:
Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.
Ярус 1. a_1a_2, a_3a_4, a_5a_6, a_7a_8.
Ярус 2. $a_1a_2 + a_3a_4$, $a_5a_6 + a_7a_8$.
Ярус 3. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Высота этой параллельной формы равна 3, а *ширина* 4.

Особенность этой параллельной формы в том, что 4 процессора загружены только на первом ярусе, а на последнем ярусе загружен лишь один процессор.

1.2. Вторая параллельная форма такова:
Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.
Ярус 1. a_1a_2, a_3a_4.
Ярус 2. a_5a_6, a_7a_8.
Ярус 3. $a_1a_2 + a_3a_4$, $a_5a_6 + a_7a_8$.
Ярус 4. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

1.3. Можно рассмотреть еще одну (третью) параллельную форму:
Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.
Ярус 1. a_1a_2, a_3a_4.
Ярус 2. $a_1a_2 + a_3a_4$, a_5a_6.
Ярус 3. a_7a_8.
Ярус 4. $a_5a_6 + a_7a_8$.
Ярус 5. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Ясно, что высота второй формы 4, третьей формы – 5, а ширина обеих форм одинакова и равна 2.

Для эффективного распараллеливания процесса стремятся

1) к увеличению загруженности системы процессоров;

2) к отысканию параллельной формы с заданными свойствами.

Пример 2. Пусть требуется перемножить числа a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.

2.1. *Обычная схема* последовательного умножения такова.
Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.
Ярус 1. a_1a_2
Ярус 2. $(a_1a_2)a_3$
...
Ярус 7. $(a_1a_2...a_7)a_8$

Высота этой параллельной формы равна 7, ширина равна 1.

2.2. Уменьшения числа ярусов можно добиться с помощью *алгоритма сдваивания*, который состоит в следующем:

Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.

Ярус 1. a_1a_2, a_3a_4, a_5a_6, a_7a_8.

Ярус 3. $(a_1a_2)(a_3a_4)$, $(a_5a_6)(a_7a_8)$.

Ярус 4. $(a_1a_2a_3a_4)(a_5a_6a_7a_8)$.

Высота такой параллельной формы равна 3, ширина – 4. Процесс подобного вида называется *схемой сдваивания*. Для его реализации нужно на каждом ярусе осуществлять максимально возможное число попарно непересекающихся перемножений чисел, полученных на предыдущем ярусе.

8. О КОНЦЕПЦИИ НЕОГРАНИЧЕННОГО ПАРАЛЛЕЛИЗМА

Концепция неограниченного параллелизма предполагает использование идеализированной модели неограниченной параллельной вычислительной системы.

Неограниченной параллельной вычислительной системой будем называть систему, работа которой производится в течение некоторого количества единиц дискретного времени, называемых тактами, и которая обладает следующими свойствами:

1) система имеет *любое нужное число идентичных процессоров*;

2) система имеет *произвольно большую память, одновременно доступную* всем процессорам;

3) каждый процессор за упомянутую единицу времени может выполнить *любую унарную* или *бинарную* операцию из некоторого априори заданного множества операций; операции из этого множества будем называть *основными*;

4) время выполнения всех вспомогательных операций (то есть операций, не являющихся основными), время взаимодействия с памятью и время, затрачиваемое на управление процессором, считаются *пренебрежимо малыми*;

5) никакие *конфликты* при общении с памятью *не возникают*;

6) все входные данные перед началом работы системы *записаны в память*;

7) после окончания вычислительного процесса все результаты *остаются в памяти*.

Рассмотрим схему сдваивания в условиях неограниченного параллелизма.

В дальнейшем символом $\lceil a \rceil$ обозначаем ближайшее к a целое число, не меньшее числа a.

Теорема 1. *Высота h параллельной формы умножения n чисел, соответствующей схеме сдваивания, не превосходит $\lceil \log_2 n \rceil$, а ее ширина w не превосходит $\lceil n/2 \rceil$.*

Доказательство почти очевидно. В случае, когда n не является степенью двойки, найдем k так, чтобы $2^{k-1} < n < 2^k$, и дополним произведение $2^k - n$ сомножителями, равными единице. Далее строим параллельную форму, соответствующую схеме сдваивания, и подсчитываем высоту и ширину этой формы. ▪

Аналогичным образом применяется схема сдваивания к сумме n чисел; в результате получается следующее утверждение.

Теорема 2. *Высота параллельной формы сложения n чисел, соответствующей схеме сдваивания, не превосходит $\lceil \log_2 n \rceil$, а ее ширина не превосходит $\lceil n/2 \rceil$.*

Доказательство аналогично доказательству теоремы 1 (в случае, когда n не является степенью двойки, добавляем в сумму соответствующее число нулевых слагаемых). ▪

Пусть теперь требуется найти не только произведение n чисел a_1, a_2, a_3, ..., a_n, но и все последовательные произведения

$$a_1,\ a_1a_2,\ a_1a_2a_3,\ \ldots,\ a_1a_2\ldots a_{n-1}.$$

Эта задача по схеме сдваивания реализуется очень эффективно. Продемонстрируем это на примере восьми сомножителей ($n = 8$); на следующей ниже схеме искомые произведения подчеркнуты.

Данные: a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8.

Ярус 1. $\underline{a_1a_2}$, a_3a_4, a_5a_6, a_7a_8.

Ярус 2. $\underline{(a_1a_2)}a_3$, $\underline{(a_1a_2)(a_3a_4)}$, $(a_5a_6)a_7$, $a_5a_6a_7a_8$.

Ярус 3. $\underline{(a_1a_2a_3a_4)a_5}$, $\underline{(a_1a_2a_3a_4)(a_5a_6)}$, $\underline{(a_1a_2a_3a_4)(a_5a_6a_7)}$, $\underline{(a_1a_2a_3a_4)(a_5a_6a_7a_8)}$.

Здесь высота $h = 3$, ширина $w = 4$, и имеется полная загрузка процессоров, однако некоторые результаты «лишние» в том смысле, что в окончательном ответе они не нужны (последнее – достаточно типичное свойство параллельных алгоритмов).

Нетрудно реализовать эту схему для произвольного числа n сомножителей, дополняя их необходимым числом единиц в случае, когда n не является степенью двойки. Эта схема тоже называется *схемой сдваивания* (для последовательных произведений). Высота и ширина этой схемы такая, как указано в теореме 1.

Отметим три особенности параллельных алгоритмов.

1. Ряд алгоритмов имеет *очень много* «лишних» операций.

2. Разные схемы параллельных алгоритмов реализуют разные алгоритмы, хотя формально (ввиду ассоциативности и коммутативности сложения и умножения в исходных формулах) результат точных действий одинаков; однако при реализации (из-за использования «машинной арифметики» и связанных с этим ошибок округления) результат может оказаться существенно другим.

3. Устойчивость параллельных алгоритмов при большом числе процессоров оказывается хуже устойчивости последовательных алгоритмов.

Свойство алгоритма, состоящее в том, что малые изменения начальных данных вызывают малые изменения результатов вычисления, называют *устойчивостью алгоритма*. Важным свойством алгоритма является степень зависимости результата от ошибок округления, а ошибки округления зависят от порядка арифметических действий. В шестом разделе было показано, что неудачный порядок арифметических действий может привести к катастрофическим последствиям из-за ошибок округления.

В случае распараллеливания ситуация усугубляется. Обычно существует много параллельных форм алгоритма; если нет ошибок округления, то все эти формы дают один и тот же результат. Однако, выбор параллельной формы предопределяет порядок арифметических действий (см. примеры из седьмого раздела). Поскольку реализация вычислений на ЭВМ для сверхсложных задач невозможна без ошибок округления, то некоторые параллельные формы могут приводить к неверным результатам вычислений. Заметим, что число процессоров современных вычислительных систем может быть очень велико (десятки и сотни тысяч – см. первый и четвертый разделы), поэтому процесс распараллеливания не может быть произведен вручную. Приходится проводить распараллеливание автоматизированно, а это ведет к полной потере контроля за порядком арифметических действий. Таким образом, актуальной становится задача об исследовании устойчивости алгоритма относительно некоторого класса возможных для него параллельных форм; такая устойчивость называется *сверхустойчивостью* алгоритма на упомянутом классе. Исследования в этой области только начинаются.

9. О ВЫЧИСЛЕНИИ СТЕПЕНИ НА ПАРАЛЛЕЛЬНОЙ СИСТЕМЕ

В заключение приведем пример, показывающий, что исследование сложности параллельных алгоритмов иногда приводит к весьма неожиданным результатам.

Нам потребуется следующее утверждение.

Лемма 1. *При всех* $x \neq e^{ik2\pi/n}$ *справедлива формула*

$$x^n = 1 + n\left(\sum_{k=1}^{n} e^{ik2\pi/n}\left(x - e^{2k\pi i/n}\right)^{-1}\right)^{-1} \qquad (9.1)$$

Доказательство. Положим $q = e^{2\pi i/n}$; тогда доказываемое соотношение можно написать в виде

$$\left(x^n - 1\right)\left(\sum_{k=1}^{n} q^k\left(x - q^k\right)^{-1}\right) = n\,.$$

Обозначая выражение в скобках через S:

$$S \overset{\text{def}}{=} \sum_{k=1}^{n} q^k\left(x - q^k\right)^{-1} = \sum_{k=1}^{n} \frac{1}{xq^{-k} - 1} \qquad (9.2)$$

и предполагая, что

$$\left|xq^{-k}\right| < 1,\ k = 1, 2, ..., n, \qquad (9.3)$$

используем формулу суммы для бесконечной геометрической прогрессии со знаменателем xq^{-k}:

$$S = -\sum_{k=1}^{n}\sum_{j=0}^{\infty}(xq^{-k})^j = -\sum_{j=0}^{\infty}x^j\sum_{k=1}^{n}q^{-kj}. \qquad (9.4)$$

1. При $q^{-j} \neq 1$ внутренняя сумма в выражении (9.4) представляет собой сумму n членов геометрической прогрессии со знаменателем q^{-j}; используя формулу суммы и обозначение $q = e^{2\pi i/n}$, находим

$$\sum_{k=1}^{n}q^{-kj} = q^{-j}\sum_{k=1}^{n}q^{-j(k-1)} = q^{-j}\sum_{k'=0}^{n-1}q^{-jk'} =$$

$$= q^{-j}\,\frac{q^{-nj}-1}{q^{-j}-1} = e^{-j2\pi i/n}\,\frac{e^{-j2\pi i}-1}{e^{-j2\pi i/n}-1} = 0.$$

2. Случай $q^{-j} = 1$ получается лишь для тех j, для которых $e^{-j2\pi i/n} = 1$, что эквивалентно равенству $j = nl$ при некотором целом l; в этом случае $q^{-kj} = 1$, и потому

$$\sum_{k=1}^{n}q^{-kj} = n.$$

Итак, из (9.4) имеем

$$S = -\sum_{\substack{0\le j<+\infty\\ j=nl\\ l-\text{целое число}}}x^j\sum_{k=1}^{n}q^{-kj} =$$

$$= -n\sum_{l=0}^{+\infty}x^{nl} = -\frac{n}{1-x^n}. \qquad (9.5)$$

Последнее означает, что $(x^n - 1)S = n$. Вспоминая определение S (см. (9.2)) и подставляя (9.5) в (9.2), убеждаемся в справедливости тождества (9.1). Благодаря аналитичности выражений в (9.2), видим, что утверждение (9.1) справедливо для всех $x \neq e^{ik2\pi/n}$, $k = 1, ..., n$. Лемма доказана. ■

Рассмотрим теперь следующий пример.

Пример 3. В условиях неограниченного параллелизма займемся задачей о вычислении x^n, где x – вещественное число, а n – натуральное.

Для отыскания x^n по схеме сдваивания требуется $\lceil\log_2 n\rceil$ параллельных умножений.

Если воспользоваться леммой 1, то можно поступить следующим образом:

1) сделать одно параллельное сложение для вычисления разностей

$$x - e^{ik\pi/n},\ k = 1, ..., n,$$

2) провести одно параллельное деление для вычисления частных

$$\frac{e^{ik\pi}}{x - e^{ik\pi}},\ k = 1, ..., n,$$

3) найти сумму n слагаемых по схеме сдваивания за $\lceil\log_2 n\rceil$ параллельных сложений,

4) сделать одно деление: делим n на полученную только что сумму,

5) добавить единицу.

В результате оказывается, что выражение x^n можно вычислить, используя две параллельные мультипликативные операции и $\lceil\log_2 n\rceil + 2$ параллельных аддитивных операций.

Замечание 2. В приведенном примере предполагается, что числа $e^{ik\pi}$, $k = 1, 2, ..., n$ подсчитаны заранее и запасены в вычислительной системе (это естественное предположение при возведении большого количества чисел x в одну и ту же степень n). Если предположить, что сложение выполняется быстрее умножения, и не принимать в расчет другие аспекты реализации, то можно считать, что предлагаемый в пунктах 1)–5) алгоритм вычисления x^n эффективнее схемы сдваивания при умножении.

10. ЗАКЛЮЧЕНИЕ

Подводя итоги, отметим следующее:

– параллелизм обработки информации является естественным отражением параллельных процессов в природе и в общественной жизни;

– для решения сверхсложных вычислительных задач необходимы суперкомпьютеры с огромными быстродействием и памятью;

– в настоящее время все суперкомпьютеры представляют собой параллельные вычислительные системы;

– при постановке задачи на суперкомпьютер решается задача распараллеливания алгоритма и оценивается его сверхустойчивость (см. восьмой пункт);

– для упрощения параллельного программирования и для поддержания переносимости программ разрабатываются специальные средства (стандарты MPI, Open MP, DVM и др.)

О средствах и способах параллельного программирования поговорим в следующий раз.

Литература

1. *Воеводин В.В.* Математические модели и методы в параллельных процессах. М., 1986. 296 с.

2. *Корнеев В.В.* Параллельные вычислительные системы. М., 1999. 320 с.

3. *Yukiya Aoyama, Jun Nakano.* RS/6000 SP: Practical MPI Programming. IBM. Technical Support Organization., 2000. 221 p. www.redbook.ibm.com

4. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб., 2002. 608 с.

5. *Демьянович Ю.К., Иванцова О.Н.* Технология программирования для распределенных параллельных систем: Курс лекций. СПб., 2005. 93 с.

6. *Демьянович Ю.К., Евдокимова Т.О.* Теория распараллеливания и синхронизация: Учебное пособие. СПб., 2005. 108 с.

7. *Демьянович Ю.К., Лебединский Д.М.* Операционная система Unix (Linux) и распараллеливание: Курс лекций. СПб., 2005. 109 с.

Демьянович Юрий Казимирович,
доктор физико-математических наук, профессор, заведующий кафедрой параллельных алгоритмов математико-механического факультета СПбГУ.