УДК 519.6

Ю. К. Демьянович

ПРОБЛЕМЫ РАСПАРАЛЛЕЛИВАНИЯ В НЕКОТОРЫХ ЛОКАЛЬНЫХ ЗАДАЧАХ

Дан краткий обзор проблем распараллеливания в локальных задачах, предложены общие принципы архитектурных решений многопроцессорных систем, предназначенных для решения упомянутых задач.

Ключевые слова: *распараллеливание, локальные задачи, архитектура вычислительных систем, кратность накрытия.*

Существует большое количество задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем. К таким задачам прежде всего относятся точные долгосрочные прогнозы климатических изменений и геологических катаклизмов (землетрясений, извержений вулканов, столкновений тектонических плит), прогнозы цунами и разрушительных ураганов, а также экологические прогнозы 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 скорости решения уже поставленных задач, поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем. К сожалению, технологические возможности увеличения быстродействия процессоров ограничены: повышение 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, т.е. 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 [1—3].

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 скоро выяснилось, что для интересующих сложных задач такого ускорения, как правило, достичь невозможно по двум причинам:

— любая задача распараллеливается лишь частично (всегда имеются фрагменты, которые невозможно распараллелить),

— коммуникационная среда, связывающая отдельные части параллельной системы, функционирует значительно медленнее процессоров, так что передача информации существенно задерживает вычисления.

Из-за этого в согласии с теоретическими выводами на практике удавалось использовать незначительную часть мощности параллельной вычислительной системы: считалось достаточно эффективным использование 20 % мощности системы. Но даже и в этом случае создание параллельных суперкомпьютеров оправдано, поскольку удается добиться значительного прогресса при решении перечисленных суперсложных задач.

В свете сказанного, на первый взгляд, кажется удивительной высокая реальная производительность ряда современных суперкомпьютеров по сравнению с их пиковой производительностью. Так, например, в последней, 32-й, редакции списка TOP500 на первом месте находится суперкомпьютер RoadRunner с пиковой производительностью 1457 Tflops (т.е. $1457 \cdot 10^{12}$ операций с плавающей точкой в секунду — от *англ*. floating point operations per second); при этом реальная производительность компьютера (на тесте LINPACK, содержащем задачи линейной алгебры) составила 1026 Tflops ($1026 \cdot 10^{12}$ операций с плавающей точкой в секунду). На втором месте в этом списке находится суперкомпьютер CrayXT5 с пиковой производительностью 1381 Tflops и реальной производительностью (на тесте LINPACK) 1059 Tflops. В первом случае реальная производительность составляет 70 % от пиковой, а во втором — еще больше — 77 %.

Аналогичная ситуация прослеживается и для других суперкомпьютеров, например, в период с 2003 по 2005 г. на первом месте списке TOP500 находился суперкомпьютер Earth Simulator японской фирмы NEC с пиковой производительностью 43 Tflops и реальной производительностью (на тесте LINPACK) почти 36 Tflops (в последней версии списка TOP500 этот компьютер опустился на 74 место); и в этом случае реальная производительность составила 84 % пиковой производительности.

Объяснение этому феномену, по-видимому, следует искать в согласованности реальных задач, на которых замерялась производительность, с архитектурными решениями, принятыми при создании компьютеров. В связи с этим возникает вопрос, в какой степени обоснованно использование теста LINPACK (и вообще задач линейной алгебры) для характеристики возможностей компьютерных систем при решении суперсложных задач.

Анализ методов решения упомянутых выше суперсложных задач показывает, что при разработке архитектуры суперкомпьютеров очень важно учитывать свойства алгоритмов численного решения систем линейных алгебраических уравнений. Более того, ввиду локального характера всех упомянутых задач матрицы соответствующих им систем уравнений имеют специальную структуру, что позволяет предложить специальную архитектуру суперкомпьютеров, названную нами локально кратной архитектурой.

О классификации архитектурных решений вычислительных систем. Остановимся на некоторых общепринятых способах классификации вычислительных систем.

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

Суперкомпьютеры каждого типа создаются в нескольких экземплярах, обычно каждый тип 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).

Имеется несколько вариантов классификации компьютеров, обычно они носят достаточно формальный характер. Приведем некоторые примеры классификации.

Классификация Флинна — определяется единственность или множественность потоков данных и команд (классы SIMD и MIMD).

Классификация Фенга — выявляется тип параллелизма: пословный или поразрядный.

Классификация Джонсона основана на использовании двух вариантов признаков: компьютеры с общей или распределенной памятью комбинируются с двумя вариантами коммуникаций: с помощью передачи сообщений или с помощью общих переменных.

Классификация Скилликорна — компьютер описывается как абстрактная структура, состоящая из компонентов четырех типов: процессора команд, процессора данных, иерархии памяти, коммутатора.

Однако приведенные способы классификации дают мало информации о возможностях системы и о том классе задач, для которого использование системы наиболее эффективно. Отметим, что имеется еще около полутора десятков других вариантов классификации, но они также содержат мало информации.

Краткая характеристика основных классов параллельных систем. Одной из важнейших характеристик параллельных вычислительных систем является тип памяти: общая или распределенная.

Общая память представляет собой физически единую структуру, к которой имеют доступ (равный) все процессоры параллельной системы. В этом случае говорят о SMP-системе (симметричной мультипроцессорной). Основные проблемы при такой организации памяти состоят в обеспечении быстрого доступа к памяти большого объема и в быстром разрешении коллизий, возникающих при практически одновременном обращении процессоров к общей памяти. SMP-система состоит из однородных процессоров и массива общей памяти, к любому фрагменту которой все процессоры имеют одинаковый (по скорости) доступ. Когерентность кэш-памяти поддерживается аппаратно. Примерами таких систем являются компьютеры фирм IBM, HP, Fujitsu и др. Из-за того что память общая, число процессоров невелико (не более 32). Система работает под управлением одной операционной системы (ОС), программирование ведется с использованием модели общей памяти (Open MP, POSIX threads).

Распределенная память (MPP — массивно-параллельная архитектура) характеризуется тем, что состоит из отдельных фрагментов — функциональных структур, физически распределенных между вычислительными модулями (процессорами). Основные проблемы, связанные с такой архитектурой, состоят в организации быстрого доступа к памяти как к единой (виртуально) структуре и в поддержании когерентности между ее фрагментами (т.е. в поддержании тождественности „копий“, появляющихся в процессе вычислений). Массивно-параллельная система обычно состоит из однородных вычислительных узлов, которые содержат несколько процессоров, локальную память, сетевой адаптер и, возможно, другие узлы.

Количество узлов может достигать нескольких тысяч, они связываются через коммуникационную среду (коммутатор и т.п.). Примерами таких систем являются IBM RS/6000, CRAY T3E, системы на базе транспьютеров (Parsytec). В программировании используется модель обмена сообщениями; распараллеливание производится с помощью библиотек в соответствии со стандартами MPI, PVM и др.

NUMA-архитектура (Non-Uniform Memory Acess) характеризуется тем, что память физически распределена, но логически общедоступна. Система состоит из однородных базовых модулей, каждый из которых содержит некоторое (небольшое) число процессоров и блока (локальной) памяти. Аппаратно поддерживается доступ к удаленной памяти, причем доступ к локальной памяти значительно быстрее, чем к удаленной. Иногда когерентность кэш-памяти отдельных узлов поддерживается аппаратно. Примерами таких компьютеров являются Sun HPC 10000, NUMA-Q 2000 и др. Число процессоров в NUMA-системах доходит до 256. Система работает под управлением одной ОС, программирование ведется с использованием модели общей памяти (Open MP, потоков POSIX).

Основным свойством параллельных векторных систем (PVP-систем) является наличие векторно-конвейерных процессоров с командами однотипной обработки данных. Несколько векторно-конвейерных процессоров имеют доступ к общей памяти (аналогично симметричной мультипроцессорной системе), составляя единый узел; несколько узлов могут быть объединены с помощью коммутатора (как в массово-параллельных системах). Примерами служат компьютеры фирм CRAY, Fujitsu VPP. При программировании используются векторизация циклов (для одного процессора) и их распараллеливание (для нескольких процессоров).

Наконец, кластерные системы представляют собой совокупность рабочих станций, или персональных компьютеров, которые используются в качестве простого варианта массивно-параллельного компьютера; в качестве коммуникационной среды применяется одна из сетевых технологий (Fast/Gigabit Ethernet, Myrinet и т.п.) с использованием шинной структуры или коммутатора. Кластерная система может иметь гетерогенный характер (т.е. представлять собой объединение компьютеров разной архитектуры). Программирование ведется в рамках модели передачи сообщений (например, с использованием библиотеки, поддерживающей стандарт MPI).

Пример сложной задачи. Характерным (и типичным) примером сложной вычислительной задачи является компьютерное моделирование климата. Климатическая система 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 ее основе лежат уравнения динамики сплошной среды и равновесной термодинамики. В модели описываются многие физические процессы, связанные с переносом энергии: перенос излучения в атмосфере, фазовые переходы воды, мелкомасштабная турбулентная диффузия 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

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

то простейшая система уравнений, моделирующая динамику атмосферы, решается в приземном сферическом слое (в тропосфере, окружающей Землю); она включает следующие уравнения:

— уравнение количества движения

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

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

— уравнение сохранения энергии

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

где c_p — удельная теплоемкость атмосферы, T — температура;

— уравнение неразрывности (сохранения массы)

$$\frac{Dp}{Dt} = -\rho\,\mathrm{div}\mathbf{v},$$

— уравнение состояния

$$p = \rho R T,$$

где R — константа.

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

— 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 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 высоте берут 40 слоев. Таким образом, получается $2{,}6\cdot10^6$ элементов. Каждый элемент описывается десятью компонентами. В фиксированный момент времени состояние атмосферы характеризуется ансамблем из $2{,}6\cdot10^7$ чисел. Условия развития процессов в атмосфере требуют каждые 10 минут находить новый ансамбль, так что за 100 лет будем иметь $5{,}3\cdot10^6$ ансамблей. Таким образом, в течение одного численного эксперимента будет получено около $\left(2{,}6\cdot10^7\right)\cdot\left(5{,}3\cdot10^6\right)\approx1{,}4\cdot10^{14}$ 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 краевых условиях требуется просчитать сотни подобных вариантов.

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

Структура и роль памяти в вычислительной системе. Идеальная память должна обеспечивать процессор командами и данными непрерывно, чтобы не допускать простоев процессора. Кроме того, память должна иметь большую емкость с тем, чтобы обеспечивать обработку необходимых заданий. Это достигается использованием многоуровневой памяти с соответствующей иерархией; иногда говорят об иерархии памятей (во множественном числе).

Время доступа к данным зависит от объема и типа используемой памяти. Малое время доступа характерно для памяти малого объема; поэтому, введя малую быструю память (кэш-память), пересылают данные из основной памяти в буферную, обрабатывают их с использованием этой памяти и результат отправляют обратно в основную память.

Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за то время, пока другие блоки обрабатываются процессором, позволяет существенно снизить простои процессора в ожидании данных.

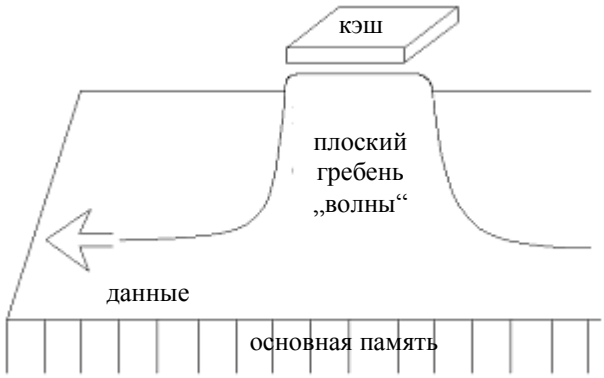

Рис. 1

Отметим, что чем более плоский „гребень“ у „волны“ (т.е. чем „шире“ волна), тем эффективнее обработка информации (рис. 1).

Что такое локальность данных? Признаки локальных данных:

1) процессор многократно использует выбранные из основной памяти данные для получения результата до того момента, когда результат будет отправлен в основную память;

2) положение выбираемых данных локализовано в основной памяти (они выбираются последовательно, „подряд“).

Эти признаки характерны для научных и инженерных расчетов (при решении уравнений математической физики), например, когда рассматривается физический процесс с локальными законами. В этом случае при переходе от значения к значению приходится обрабатывать небольшие „порции“ данных с большим количеством вложенных циклов, так что все данные из обрабатываемой „порции“ используются многократно.

В связи с тем что локально обрабатываемые данные могут возникать в динамике вычислений и не быть сконцентрированными в одной области при статическом размещении в основной памяти, буферную память организуют как ассоциативную (в которой данные содержатся в совокупности с их адресом в основной памяти, рис. 2). В результате получаем гибкое согласование структуры данных, требуемых в процессе вычислений, со статическими структурами основной памяти.

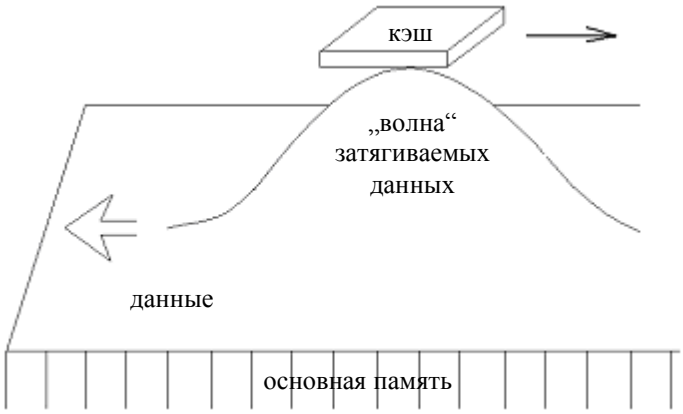

Рис. 2

Локально кратная архитектура параллельной системы. Наиболее эффективные методы решения упомянутых выше задач сводятся к выбору координатных функций в подходящем линейном пространстве и к построению приближения в его конечномерном подпространстве. Типичным примером является широко распространенный метод конечных элементов (МКЭ), харак-

терная черта которого — применение координатных функций с компактным носителем и с малой кратностью накрытия. Функции с аналогичными свойствами применяются для обработки числовых информационных потоков, высокая плотность которых требует больших вычислительных мощностей (сюда относятся обработка двумерных и трехмерных информационных потоков, связанных с натурными экспериментами, томографией, с обработкой космических потоков, передачей двумерных и трехмерных изображений, распознаванием образов и т.п.).

Для эффективного решения подобных задач необходимы вычислительные системы, архитектура которых согласована с применяемыми методами. Рассмотрим один из вариантов подобной архитектуры.

Пусть $\mathbb{N}$ — множество натуральных чисел, а $\mathbb{N}_0 \overset{\text{def}}{=} \mathbb{N} \cup \{0\}$. На некотором n-мерном дифференцируемом многообразии T (для удобства в качестве T можно рассматривать, например, тор или сферу) введем клеточное подразделение $U \overset{\text{def}}{=} \{K_j\}_{j \in J}$, т.е. рассмотрим непересекающиеся множества (клетки) K_j, гомеоморфные n-мерному открытому шару, объединение замыканий которых совпадает с T,

$$T = \bigcup_{j \in J} \bar{K}_j,$$

J — некоторое конечное множество индексов.

Пусть $p \in \mathbb{N}_0$ и $s \in \mathbb{N}$.

Определение 1. Клетки называются инцидентными друг другу, если их замыкания пересекаются. Две клетки называются *p-инцидентными*, если пересечение их замыканий имеет размерность j, где $j \ge p$, $j \in \mathbb{N}_0$.

Выделим в множестве U некоторое подмножество U^*, среди клеток которого нет инцидентных, т.е. $U^* \subset U$, причем для $\forall K', K'' \in U^*$ имеем $\bar{K} \cap \bar{K} = \varnothing$.

Определение 2. Клетки множества U^* будем называть *помеченными*. Если известно, что клетка помеченная, то будем обозначать ее символом K^*.

Определение 3. Две клетки K' и K'' называются (p,s)-*связанными*, если существует последовательность клеток

$$C^{(p,s)}(K', K'') \overset{\text{def}}{=} \{K' \overset{\text{def}}{=} K^{(0)}, K^{(1)}, K^{(2)}, \ldots, K^{(s)} \overset{\text{def}}{=} K''\}, \qquad (1)$$

каждые две соседних клетки в которой p-инцидентны. Последовательность (1) называется (p,s)-*связующей цепочкой* (или просто (p,s)-*цепочкой*) для клеток K' и K'', число s — *длиной цепочки*, а число p называется *гарантированной мощностью цепочки*.

Для заданных клеток K' и K'' и фиксированной пары чисел (p,s) может быть несколько цепочек вида (1).

Множество всех цепочек обозначим $C^{(p,s)}(K', K'')$, оно не пусто лишь для конечного количества целочисленных значений s и p. Очевидно, что множество

$$C^{(p)}(K', K'') \overset{\text{def}}{=} \bigcup_{s \in \mathbb{N}} C^{(p,s)}(K', K'')$$

представляет собой множество цепочек одинаковой гарантированной мощности, связывающих клетки K' и K'', а множество

$$C^{(s)}(K', K'') \overset{\text{def}}{=} \{C^{(p,s)}(K', K'') \mid p \in \mathbb{N}_0\}$$

является множеством цепочек одинаковой длины (возможно, различных гарантированных мощностей), связывающих клетки K' и K''.

Введем числа $s^{(p)}_{\min}(K',K'')$ и $s^{(p)}_{\max}(K',K'')$, определяемые формулами

$$s^{(p)}_{\min}(K',K'') \overset{\text{def}}{=} \min_{s\in\mathbb{N}}\{s \mid C^{(p,s)}(K',K'') \neq \varnothing\},$$

$$s^{(p)}_{\max}(K',K'') \overset{\text{def}}{=} \max_{s\in\mathbb{N}}\{s \mid C^{(p,s)}(K',K'') \neq \varnothing\}.$$

Число $s^{(p)}_{\min}(K',K'')$ называется минимальной, а число $s^{(p)}_{\max}(K',K'')$ — максимальной *p-удаленностью* клеток K' и K''; полезными оказываются также числа

$$s_{\min}(K',K'') \overset{\text{def}}{=} \min_{p\in\mathbb{N}_0} s^{(p)}_{\min}(K',K''),$$

$$s_{\max}(K',K'') \overset{\text{def}}{=} \max_{p\in\mathbb{N}_0} s^{(p)}_{\max}(K',K''),$$

называемые минимальной и максимальной удаленностью клеток K' и K'' соответственно. Отметим некоторые свойства p-инцидентности и (p,s)-связанности. Пусть $p_1 \le p$, тогда

1) если две клетки p-инцидентны, то они и p_1-инцидентны,

2) если цепочка (1) — (p,s)-связующая для клеток K' и K'', то она является и (p_1,s)-связующей для этих клеток,

3) если две клетки (p,s)-связаны, то они и (p_1,s)-связаны,

4) если две клетки $(p,1)$-связаны, то они p-инцидентны,

5) справедливы включения

$$C^{(p,s)}(K',K'') \subset C^{(p_1,s)}(K',K''),\ C^{(p)}(K',K'') \subset C^{(p_1)}(K',K'').$$

Архитектуру предлагаемой вычислительной системы согласуем с клеточным подразделением следующим образом.

Будем считать, что каждая клетка представляет собой фрагмент памяти (далее термины „фрагмент памяти“ и „клетка“ отождествляем), а коммуникации между (p,s)-связанными фрагментами памяти K' и K'' происходят по каналам, ассоциированным с (p,s)-связующими цепочками длиной s и мощностью p.

Как известно, процессор (вместе с соответствующим оборудованием) можно рассматривать как фрагмент памяти с большим быстродействием. Будем считать, что каждая помеченная клетка представляет собой процессор (дальше термины „процессор“ и „помеченная клетка“ отождествляются). Если помеченная клетка фиксирована, то по отношению к ней любая другая клетка может рассматриваться как кэш-память. Более традиционно каждой помеченной клетке K^* соотнести некоторую группу $G(K^*)$ „близлежащих“ к ней (в смысле упомянутой топологии) непомеченных клеток, например, клеток, связанных с K^* цепочками длиной $s \le s_0(K^*)$, где $s_0(K^*)$ — некоторое фиксированное число. Совокупность клеток $G_s(K^*)$, связанных цепочками длиной s с клеткой K^*, $s \le s_0(K^*)$, называется кэшем s-го уровня для процессора K^*.

Определение 4. Объединение $M(K^*) \overset{def}{=} G(K^*) \cap \{K^*\}$ называется вычислительным модулем, а $G(K^*)$ — его кэшем. Совокупность всех вычислительных модулей называется вычислительной системой.

Существенным моментом, отличающим предлагаемую архитектуру от обычно рассматриваемых, является то, что кэши различных вычислительных модулей могут пересекаться; в частности, априори фиксированная клетка K может принадлежать кэшам нескольких вычислительных модулей.

Определение 5. Число вычислительных модулей, которым принадлежит данный фрагмент памяти $K \in U$, называется кратностью накрытия этого фрагмента и обозначается $\varkappa(K)$.

Число $\varkappa_0 \overset{\text{def}}{=} \max_{K \in U} \varkappa(K)$ называется кратностью вычислительной системы.

Вычислительная система с рассмотренной архитектурой называется локально кратной вычислительной системой.

Если кратность накрытия можно изменять добавлением или удалением вычислительных модулей, то вычислительную систему можно назвать локально кратной вычислительной системой с изменяемой кратностью накрытия.

Закончим изложение замечанием о том, что для практических целей было бы важно иметь *однородную* локально кратную вычислительную систему, т.е. такую, у которой кратность накрытия для всех фрагментов памяти одинакова: $\varkappa(K) = \varkappa_0 \ \forall K \in U$.

Заключение. Анализ сложных задач и методов их численного решения показывает, что при разработке архитектуры суперкомпьютеров очень важно учитывать свойства алгоритмов решения систем линейных алгебраических уравнений. Ввиду локального характера упомянутых задач требования к архитектуре суперкомпьютеров могут быть в значительной степени конкретизированы; они дают архитектурные решения, которые могут привести к созданию суперкомпьютеров, занимающих промежуточное положение между компьютерами с разделяемой памятью и компьютерами с распределенной памятью: память разделена на фрагменты, к каждому фрагменту имеют прямой доступ несколько вычислительных модулей; в свою очередь, каждый вычислительный модуль имеет прямой доступ к нескольким фрагментам памяти. Такая архитектура полностью соответствует локальным аппроксимациям, методам конечных элементов, сплайнам и вейвлетам, применяемым при решении перечисленных выше сложных задач [4, 5]; этот подход приведет к существенному повышению реальной производительности на упомянутых задачах.

Работа частично поддержана грантами РФФИ № 07-01-00451 и 07-01-00269.

СПИСОК ЛИТЕРАТУРЫ

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб: БХВ-Петербург, 2002. 608 с.

2. *Эндрюс Г. Р.* Основы многопоточного, параллельного и распределенного программирования. М.: Вильямс, 2003. 512 с.

3. [Электронный ресурс]: <http://parallel.ru>.

4. *Демьянович Ю. К.* Локальная аппроксимация на многообразии и минимальные сплайны. СПб: Изд-во СПбГУ, 1994. 356 с.

5. *Демьянович Ю. К.* Вэйвлеты на многообразии // Докл. РАН. 2009. Т. 421, № 2. С. 1—5.

Сведения об авторе

Юрий Казимирович Демьянович — д-р. физ.-мат. наук, профессор; Санкт-Петербургский государственный университет, кафедра параллельных алгоритмов; зав. кафедрой; E-mail: Yuri.Demjanovich@JD16531.spb.edu

Рекомендована институтом

Поступила в редакцию
10.03.09 г.