



С. В. Борзунов, С. Д. Кургалин, А. В. Флегель

Практикум по параллельному программированию

.....

- *Теоретическая часть: методы параллельного программирования и средства распараллеливания*
- *Разработка программ с помощью технологий OpenMP и MPI*
- *Большое число разобранных задач, примеров и упражнений*
- *Контрольные вопросы, указания к решениям, ответы*
- *Дополнительная справочная информация*

bhv[®]



С. В. Борзунов
С. Д. Кургалин
А. В. Флегель

Практикум по параллельному программированию

Санкт-Петербург

«БХВ»

2017

УДК 519.6+004.42(075.8)

ББК 32.973я73

Б82

Борзунов, С. В.

Б82 Практикум по параллельному программированию:

учеб. пособие / С. В. Борзунов, С. Д. Кургалин,

А. В. Флегель. — СПб.: БХВ, 2017. — 236 с.: ил. —

(Учебная литература для вузов)

ISBN 978-5-9909805-0-1

В учебное пособие включены основные теоретические сведения о методах программирования для многопроцессорных вычислительных систем, указания по разработке параллельных программ с помощью технологий OpenMP и MPI, а также контрольные вопросы и задачи широкого спектра сложности как для проведения занятий в компьютерных классах и аудиториях, так и для самостоятельного решения. Многие задачи снабжены ответами или решениями, в том числе с образцами кода.

*Для студентов, аспирантов
и преподавателей профильных вузов*

УДК 519.6+004.42(075.8)

ББК 32.973я73

РЕЦЕНЗЕНТЫ:

В. П. Гергель, д-р тех. наук, проф., зав. кафедрой программной инженерии Нижегородского государственного университета, директор Института информационных технологий, математики и механики;

Кафедра высшей математики и физико-математического моделирования Воронежского государственного технического университета (зав. кафедрой д-р физ.-мат. наук, проф. *И. Л. Батаронов*)

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9909805-0-1

© Борзунов С. В., Кургалин С. Д., Флегель А. В., 2017
© Оформление. ООО "БХВ", 2017

Оглавление

Список основных обозначений	6
Предисловие	7
Глава 1. Классификация архитектур вычислительных систем	11
1.1. Классификация Флинна	12
1.2. Средства параллельного программирования	15
Контрольные вопросы к главе «Классификация архитектур вычислительных систем»	16
Глава 2. Основы параллельных вычислений	17
2.1. Модели RAM и PRAM	17
2.2. Граф «операции–операнды»	28
2.3. Условия Бернштейна	30
Контрольные вопросы к главе «Основы параллельных вычислений»	36
Задачи к главе «Основы параллельных вычислений»	36
Глава 3. Технология OpenMP	39
3.1. Модель параллельной программы	40
3.2. Основные директивы OpenMP	41
3.3. Вспомогательные функции и переменные окружения	44
3.4. Основные методы распараллеливания. Параллельные циклы и параллельные секции	46
3.5. Синхронизация	50
3.6. Механизм замков	51
3.7. Реализация базовых алгоритмов	62
3.8. Параллельная реализация методов Монте-Карло	76
3.9. Матричные операции в OpenMP	80
Контрольные вопросы к главе «Технология OpenMP»	86
Задачи к главе «Технология OpenMP»	86

Глава 4. Технология MPI	93
4.1. Компиляция и запуск параллельной программы в среде MPI. Важнейшие функции MPI	94
4.2. Стандартный способ передачи сообщений	98
4.3. Групповые имена и недействительные процессы	105
4.4. Измерение времени	105
4.5. Способы передачи сообщений	106
4.6. Коллективные взаимодействия	111
Контрольные вопросы к главе «Технология MPI»	119
Задачи к главе «Технология MPI»	120
Глава 5. Пример использования параллельных технологий для решения физической задачи	123
5.1. Математическое описание квантовой системы	123
5.2. Задача электрон-атомного рассеяния в периодическом поле	126
5.3. Методы численного решения уравнений задачи	128
5.4. Примеры FFTW-подпрограмм для ДПФ	131
5.4.1. Комплексное одномерное ДПФ	131
5.4.2. Комплексное многомерное ДПФ	132
5.4.3. Двумерное комплексное ДПФ с использованием MPI	133
5.4.4. Многомерное дискретное преобразование Фурье вещественных данных с использованием MPI	134
5.5. Параллельная реализация с комбинированным использованием OpenMP и MPI	136
5.6. Анализ эффективности методов распараллеливания	138
Контрольные вопросы к главе «Пример использования параллельных технологий для решения физической задачи»	140
Глава 6. Ответы, указания, решения к задачам	141
Приложение А. Методы оценки эффективности алгоритмов	175
A.1. «O-символика»	175
A.2. Методы анализа алгоритмов	179
Приложение В. Использование командного интерпретатора операционной системы Linux для запуска параллельных программ	182
V.1. Командный интерпретатор операционной системы Linux	182
V.2. Файловая система Linux	184
V.3. Основные каталоги файловой системы	188

В.4. Пользователи и группы	189
В.5. Основные операции с файлами	192
В.6. Процессы	194
В.7. Установка сред параллельного программирования на рабочую станцию	198
В.7.1. Среда OpenMP	199
В.7.2. Среда MPI	199

Приложение С. Параллельный алгоритм численного интегрирования нестационарного уравнения Шредингера

С.1. Многомерное нестационарное уравнение Шредингера	201
С.2. Пошаговый оператор расщепления	203
С.3. Спектральные базисы	204
С.4. Параллельный алгоритм многомерного быстрого преобразования Фурье	206
С.5. Численный пример: ионизация молекулярного иона водорода H_2^+	211

Приложение D. Преобразование Фурье

D.1. Дискретное преобразование Фурье	216
--	-----

Библиографический список	222
---	------------

Указатель имен	230
---------------------------------	------------

Предметный указатель	231
---------------------------------------	------------

Список основных обозначений

\mathbb{N}	множество натуральных чисел
\mathbb{R}	множество вещественных чисел
A и B	конъюнкция логических выражений A и B
A или B	дизъюнкция логических выражений A и B
$T_p(N)$	время работы вычислительной системы с p процессорами над задачей с параметром размера входных данных N
$T_\infty(N)$	время работы паракомпьютера над задачей с параметром размера входных данных N
$O(g(n))$	класс функций, растущих не быстрее $g(n)$
$\Omega(g(n))$	класс функций, растущих по крайней мере так же быстро, как $g(n)$
$\Theta(g(n))$	класс функций, растущих с той же скоростью, что и $g(n)$
$S_p(N)$	ускорение, получаемое при использовании параллельного алгоритма на машине с p процессорами
$E_p(N)$	эффективность использования процессоров параллельным алгоритмом
$C_p(N)$	стоимость вычислений на машине с p процессорами
f	доля последовательных вычислений в алгоритме
$R(u)$	множество входных объектов оператора u
$W(u)$	множество выходных объектов оператора u
$u \perp v$	операторы u и v находятся в отношении зависимости
$u \delta^0 v$	зависимость u и v по выходу
$u \delta v$	истинная или потоковая зависимость u и v
$u \bar{\delta} v$	антизависимость u и v

Предисловие

В настоящее время решение актуальных научно-технических и прикладных задач, в их числе моделирование физических, химических, биологических и иных сложных процессов и объектов, прогнозирование в экономике и финансовом секторе, управление большими промышленными и транспортными комплексами и др., не обходится без привлечения высокопроизводительных компьютерных систем, в которых применяются методы параллельного программирования.

Данное учебное пособие содержит базовые теоретические представления о методах параллельного программирования и сведения о средствах распараллеливания. Цель пособия состоит в том, чтобы помочь приобрести умения и навыки в решении практических задач с использованием возможностей многопроцессорных вычислительных систем.

В учебном пособии суммирован многолетний опыт преподавания на факультете компьютерных наук Воронежского государственного университета (ВГУ) курсов «Параллельное программирование», «Технологии параллельных вычислений», «Параллельные и Grid-технологии» и «Параллельные алгоритмы обработки данных» для направлений подготовки бакалавров «Математика и компьютерные науки», «Информационные системы и технологии», «Программная инженерия» и для направления подготовки магистров «Математика и компьютерные науки», а также курса «Информационно-коммуникационные технологии» для повышения квалификации преподавателей вузов.

Начиная с 2002 года, когда на кафедре цифровых технологий ВГУ был установлен первый в университете параллельный компьютерный кластер, здесь ведется подготовка в области параллельных вычислений. На базе кластера была создана лаборатория высокопроизводительных параллельных вычислений, которая в 2011 году преобразована в Суперкомпьютерный центр ВГУ.

Для освоения материала учебного пособия требуется знание основ математического анализа, дискретной математики, линейной алгебры, численных методов и навыки программирования на языке C.

Учебное пособие состоит из шести глав.

В первой главе приводится обзор основных средств параллельного программирования и рассматривается классификация архитектур вычислительных систем.

Вторая глава посвящена основным законам параллельных вычислений. В рамках модели PRAM рассматриваются количественные характеристики эффективности параллельных алгоритмов. Здесь также вводится понятие графа «операции–операнды» и формулируются условия Бернштейна.

Следующие две главы содержат описание широко используемых технологий параллельного программирования.

В третьей главе рассматривается среда программирования вычислительных систем с общей памятью OpenMP и применяемая при работе с данной средой модель вычислений. На примере технологии OpenMP приводятся реализации базовых параллельных алгоритмов, в том числе алгоритма суммирования элементов массива, сортировки данных, основных матричных операций, метода статистических испытаний Монте-Карло.

Четвертая глава посвящена описанию технологии программирования систем с распределенной памятью MPI. Рассматриваются важнейшие функции MPI, операции попарного обмена сообщениями в блокирующем и неблокирующем вариантах и коллективные операции передачи данных.

В пятой главе на основе представленных в данном пособии методов демонстрируется пример использования параллельных технологий для решения сложной физической задачи.

Шестая глава содержит ответы, указания и решения задач.

В приложениях находится справочная информация, включающая методы анализа алгоритмов, описание работы с командным интерпретатором операционной системы Linux, теоретические основы Фурье-анализа.

Материал учебного пособия организован следующим образом. Каждая глава начинается с теоретической части, где задается используемая терминология и приведены основные положения и методы параллельного программирования. В тексте, наряду с базовыми определениями, теоретическими положениями и моделями вычислений, содержатся указания по их использованию при разработке параллельных программ. Далее проводится подробный разбор нескольких типовых задач. Отметим, что если в тексте приводится пример, то его окончание обозначается знаком □. Завершается глава списком контрольных вопросов, служащих для проверки и закрепления теоретической части материала, затем даются задачи для решения в аудитории или в компьютерном классе. Их можно

использовать и для самостоятельной работы. Многие задачи снабжены ответами, указаниями или решениями, в том числе с образцами кода программ.

Перечислим основные отличия этой книги от других учебных пособий по параллельному программированию и высокопроизводительным вычислениям.

Во-первых, книга рассчитана на широкий уровень подготовки читателей — от студентов младших курсов, только начинающих знакомиться с предметом, до магистрантов, аспирантов, научных сотрудников и специалистов. Наличие в пособии как теоретических материалов, так и подробно разобранных примеров решения вычислительных задач, а также присутствие заданий для самостоятельной работы делает его полезным как для преподавания в вузах курсов, связанных с параллельным программированием, так и для использования исследователями и программистами для проведения расчетов на параллельных компьютерных системах.

Во-вторых, изложение материала ведется на достаточно строгом математическом уровне, все утверждения (леммы, теоремы) приводятся с доказательством, либо — в небольшом числе случаев — даются ссылки на специализированную литературу по изучаемой теме. Наличие учебного материала, охватывающего построение и анализ параллельных алгоритмов, отличает данную книгу от многих руководств по высокопроизводительным вычислениям, имеющих на русском языке.

В-третьих, в предлагаемых примерах и задачах большое внимание уделяется проблеме обеспечения необходимой точности вычислений. Во всех случаях строго оговариваются требуемые от компьютерной программы границы погрешностей вычислений. Тексты программ, представляющих решение как типовых, так и наиболее сложных задач, приводятся, обычно, целиком, без сокращений, что особенно важно при первоначальном знакомстве с материалом книги. Наличие в книге справочной информации позволяет уменьшить потребность в обращении к специализированной справочной литературе.

Ко всем параллельным программам, используемым в качестве примеров, приводятся графики ускорения вычислительной части программного кода в зависимости от числа нитей в системе с общей памятью. Расчет величин ускорения проводился на кластере Суперкомпьютерного центра ВГУ. Наглядное представление зависимости ускорения от параметра — числа нитей — позволяет обучающимся глубже понять принципы работы параллельных алгоритмов.

В книге имеется достаточное число иллюстраций, помогающих наглядно представить изучаемые объекты и связи между ними.

После большинства вновь появляющихся в тексте терминов в круглых скобках даны соответствующие им термины на английском языке.

При первом упоминании в тексте фамилий ученых и исследователей в ссылках приводятся их краткие биографические данные, взятые из [5, 98].

Список основных обозначений содержит определения наиболее часто используемых величин.

Дополняет текст библиографический список, в который включены современные источники информации (в том числе и электронные) по программированию многопроцессорных вычислительных систем.

Завершают учебное пособие именной и предметный указатели.

Глава 1

Классификация архитектур вычислительных систем

Разработка программ для многопроцессорных вычислительных систем или, как говорят, параллельное программирование, в настоящее время привлекает все большее внимание как исследователей, так и прикладных программистов. К этому приводят следующие причины.

Во-первых, в нашей стране (и в мире в целом) возникают все новые суперкомпьютерные центры, оснащаемые многопроцессорными вычислительными системами, в которых используются методы параллельного программирования. Суперкомпьютерные центры привлекаются для решения наиболее сложных задач, в вузах — и для подготовки современных специалистов в области информационных технологий самого высокого уровня. Во-вторых, большинство широко распространенных компьютерных систем снабжается многоядерными процессорами, и, кроме того, в них присутствуют несколько (два, четыре, восемь и более) процессоров. В-третьих, задачи, с которыми приходится иметь дело программистам, имеют тенденцию к резкому повышению требований по отношению к ресурсам вычислительной системы как по сложности вычислений, так и по объему памяти.

В силу названных причин актуальным является освоение современных суперкомпьютерных средств с максимальным использованием в них ресурсов параллелизма.

Трудности, встречающиеся при разработке обычных (последовательных) версий программ, безусловно, остаются и при создании параллельных версий таких программ. Кроме того, к ним добавляются проблемы, отражающие особенности решения задач на многопроцессорных вычислительных системах. Конечно, желательно иметь возможность пользоваться стандартными средствами распараллеливания, однако они, как

будет показано ниже, обладают целым рядом недостатков, главным из которых является не слишком высокая эффективность таких программ. В противном случае возникают проблемы, связанные с сопровождением и переносимостью программных комплексов. К счастью, такие средства реализованы для многих параллельных архитектур. Для дальнейшего рассмотрения нам потребуется классификация архитектур параллельных вычислительных систем.

1.1. Классификация Флинна

Большое разнообразие существующих архитектур вычислительных систем приводит к необходимости их классификации по различным параметрам. Исторически один из первых способов разделения архитектур по критерию множественности потоков команд и потоков данных был предложен Флинном¹.

Поток определяется как последовательность команд или данных, выполняемых или обрабатываемых процессором [47, 50]. С этой точки зрения программа предоставляет процессору поток инструкций для исполнения; данные для обработки поступают также в виде потока. В соответствии с классификацией Флинна потоки команд и потоки данных предполагаются независимыми. В связи с этим вычислительные системы разделяют на следующие классы.

1. Один поток команд, один поток данных (Single Instruction stream, Single Data stream; SISD). Такими характеристиками обладают стандартные компьютеры с одноядерным процессором, которые в каждый момент времени могут выполнять только одно действие.
2. Один поток команд, несколько потоков данных (Single Instruction stream, Multiple Data stream; SIMD). В таких системах одна и та же операция выполняется одновременно над различными данными. К данному классу относят, например, векторные вычислительные системы, в которых одна команда может выполняться над множеством элементов данных.
3. Несколько потоков команд, один поток данных (Multiple Instruction stream, Single Data stream; MISD). Несмотря на недостаточную практическую значимость данного подхода, машины MISD могут быть полезны в некоторых узкоспециальных задачах.

¹ Флинн (Michael J. Flynn) (род. 1934) — американский исследователь, специалист в области архитектур вычислительных систем.

4. Несколько потоков команд, несколько потоков данных (Multiple Instruction stream, Multiple Data stream; MIMD). Системы MIMD составляют наиболее обширную и разнообразную группу в классификации Флинна. Большинство современных многопроцессорных вычислительных систем относится именно к этому классу.

Свойства рассмотренных классов вычислительных систем схематично представлены в табл. 1.1. В ячейках записаны примеры арифметических операций, доступных для одновременного выполнения соответствующими системами.

Таблица 1.1

Классификация Флинна

Вычислительные системы

		Поток данных	
		один	несколько
Поток команд	один	SISD $\begin{Bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{Bmatrix}$	SIMD $\begin{Bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{Bmatrix}$
	несколько	MISD $\begin{Bmatrix} a_1 + b_1 \\ a_1 - b_1 \\ a_1 * b_1 \end{Bmatrix}$	MIMD $\begin{Bmatrix} a_1 + b_1 \\ a_2 - b_2 \\ a_3 * b_3 \end{Bmatrix}$

Широко используется уточнение классификации Флинна, согласно которому происходит разделение категории MIMD по способу организации памяти вычислительной системы. Среди MIMD-систем выделяют **мультипроцессоры** — машины с общей памятью (Uniform Memory Access, UMA), и **мультикомпьютеры** — машины, не обладающие удаленным доступом к памяти (NO Remote Memory Access, NORMA). Взаимодействие между процессорами в мультикомпьютерах осуществляется с помощью механизма передачи сообщений [3, 90].

Для целей параллельного программирования принципиальным моментом является принадлежность архитектуры используемой вычислительной системы к одному из трех вариантов: системы с общей памятью; системы с распределенной памятью; гибридные системы, совмещающие элементные базы двух вышеперечисленных.

Классификация многопроцессорных вычислительных систем класса MIMD по принципу организации памяти приведена на рис. 1.1.

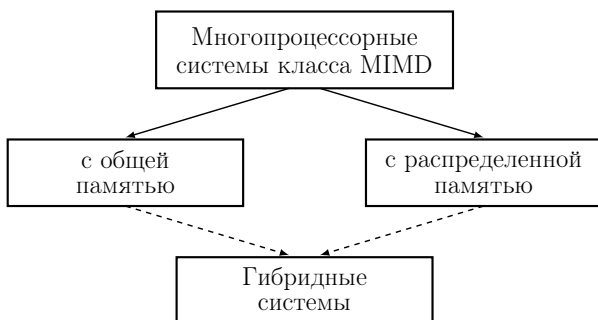


Рис. 1.1. Классификация многопроцессорных вычислительных систем класса MIMD по принципу организации памяти

Примечание. В последнее время все большее внимание привлекают вычислительные системы, использующие графические процессоры (GPU). Программирование таких систем имеет свои особенности, поскольку современные GPU, в отличие от центральных процессоров, представляют из себя массивно-параллельные вычислительные устройства с большим количеством вычислительных ядер и иерархически организованной собственной памятью [41]. В силу этого работа с графическими процессорами изучается обычно после получения базовых знаний в области программирования мультипроцессоров и/или мультикомпьютеров.

В системах с общей памятью несколько физических процессоров или процессорных ядер (как говорят, **вычислительных узлов**) имеют возможность совместно использовать единую область памяти. Данной ситуации отвечает **модель с общей памятью**. Вопросы одновременного доступа нескольких вычислительных узлов к участкам памяти занимают центральное место в модели и решаются с помощью механизмов **синхронизации**, к которым относят барьеры, замки, семафоры и др. (см. далее и в [4, 14, 51]).

В системах с распределенной памятью каждый вычислительный узел имеет доступ только к принадлежащему ему участку памяти — локальной памяти. В этом случае для межпроцессорного обмена данными преду-

считается возможность отправки и приема сообщений по коммуникационной сети, объединяющей вычислительную систему. Соответствующая модель программирования носит название **модели передачи сообщений**.

Необходимо отметить, что программирование гибридных систем основано на использовании вышеперечисленных моделей или их комбинаций.

1.2. Средства параллельного программирования

Сформулированы несколько подходов к использованию ресурсов параллелизма в разрабатываемой программе [15]:

- 1) автоматическое распараллеливание последовательной версии программы средствами компилятора;
- 2) использование специализированных языков для параллельного программирования;
- 3) использование библиотек, предоставляющих возможности параллельного исполнения кода;
- 4) программирование с использованием особых расширений языка — средств распараллеливания.

Каждый из этих подходов обладает как преимуществами, так и недостатками.

Например, автоматическое распараллеливание программ (первый пункт) очень удобно для прикладных программистов, но часто его результаты, особенно для программ, основанных на сложных алгоритмах, оказываются неудовлетворительными.

Второй и третий подходы обладают положительной стороной, связанной с неизменностью кода по сравнению с последовательной версией программы.

Среди недостатков перечисленных трех подходов следует выделить отсутствие некоторых возможностей для оптимизации компилятором исполняемого кода, достаточно высокие накладные расходы при обращении к библиотечным процедурам и функциям, а также снижение переносимости исходного кода программы по сравнению с ее последовательной версией.

Наибольшие возможности для использования ресурсов параллелизма предоставляет четвертый подход — использование особых расширений языка (средств распараллеливания). Этот подход и будет подробно рассмотрен в данном учебном пособии.

Контрольные вопросы к главе «Классификация архитектур вычислительных систем»

1. Опишите классификацию Флинна архитектур вычислительных систем.
2. На основании какого критерия производится деление МММД-систем на мультипроцессоры и мультикомпьютеры?
3. Приведите классификацию многопроцессорных вычислительных систем по принципу организации памяти.
4. Расскажите о преимуществах и недостатках различных подходов к использованию ресурсов параллелизма.

Глава 2

ОСНОВЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

2.1. Модели RAM и PRAM

Для анализа производительности программ широко используется модель вычислительной системы, называемая **машиной с произвольным доступом к памяти** (Random Access Machine, RAM) [58, 82].

Перечислим основные свойства RAM.

Система, работающая в рамках модели RAM, состоит из процессора, устройства доступа к памяти (системной шины) и памяти, состоящей из конечного числа ячеек (рис. 2.1).

Процессор последовательно выполняет команды, заложенные в программе Π ; ему доступны основные арифметические и логические операции и чтение/запись данных в памяти. При этом постулируется, что каждая команда выполняется за фиксированное время.

Произвольное действие процессора состоит из трех этапов:

- 1) чтение данных из памяти в один из своих регистров r_i , где $1 \leq i \leq N$;
- 2) выполнение арифметической или логической операции над содержимым своих регистров;
- 3) запись данных из регистра r_j , где $1 \leq j \leq N$, в некоторую ячейку памяти.

Считается, что исполнение трех перечисленных шагов требует времени $\Theta(1)$. (Функция $\Theta(f(n))$ используется для оценки скорости работы алгоритма в зависимости от размера n его входных данных [1, 65]. Так, например, $\Theta(n^2)$ указывает на соответствующую квадратичную зависимость, $\Theta(\log_2 n)$ — на логарифмическую, а $\Theta(1)$ означает отсутствие зависимости от размера входных данных. Более подробные сведения

об оценках скорости работы алгоритмов см. в Приложении А «Методы оценки эффективности алгоритмов».)

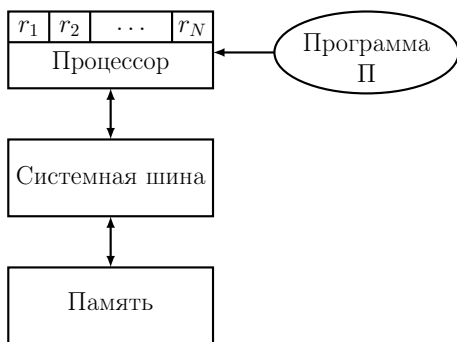


Рис. 2.1. Модель RAM

Одна из самых распространенных моделей параллельных компьютерных систем — параллельная машина с произвольным доступом к памяти (Parallel Random Access Machine, PRAM) [45, 82, 90]. В PRAM объединены p процессоров, общая память и устройство управления, которое передает команды программы Π процессорам (рис. 2.2).

Важной особенностью PRAM является ограниченное время доступа любого из процессоров системы к произвольной ячейке памяти. Как и в случае RAM, шаг алгоритма здесь также соответствует трем действиям процессора:

- 1) чтение данных процессором P_i из j -й ячейки памяти;
- 2) выполнение арифметической или логической операции процессором P_i над содержимым своих регистров;
- 3) запись данных в k -ю ячейку памяти.

Как было отмечено выше, любой шаг алгоритма выполняется за время $\Theta(1)$.

Одновременное обращение двух и более процессоров к одной и той же ячейке памяти приводит к **конфликтам доступа**. Они подразделяются на **конфликты чтения** и **конфликты записи**.

Если несколько процессоров пытаются прочесть данные из одной ячейки, то возможны два варианта дальнейших действий.

1. Исключающее чтение (Exclusive Read, ER). В данный момент времени чтение разрешено только одному процессору, в противном случае происходит ошибка выполнения программы.

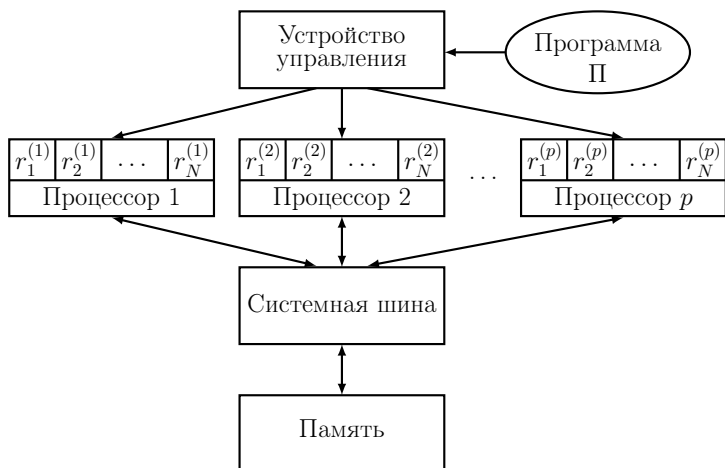


Рис. 2.2. Модель PRAM

2. Одновременное чтение (Concurrent Read, CR). Количество процессоров, получающих доступ к одной ячейке памяти, не ограничивается.

Если более одного процессора пытаются записать данные по одному адресу, разделяют два способа действия.

1. Исключающая запись (Exclusive Write, EW). Только одному процессору разрешено осуществлять запись в данную ячейку в конкретный момент времени.
2. Одновременная запись (Concurrent Write, CW). Несколько процессоров сразу получают доступ для записи к одной ячейке памяти.

Возникающие в последнем случае варианты того, по какому правилу будет определяться процессор (или процессоры), который осуществит запись, представлены ниже [72, 82, 90].

Запись общего значения. Предполагается, что все процессоры, готовые производить запись в одну и ту же ячейку памяти, обязаны записывать только общее для всех них значение, иначе инструкция записи в отведенную область памяти считается ошибочной.

Произвольный выбор. Процессор, осуществляющий запись, выбирается случайным образом.

Запись с учетом приоритетов. Каждому из конкурирующих процессоров присваивается некоторый приоритет, например, его порядковый номер, при этом сохраняется только то значение, которое поступило от процессора с заранее определенным приоритетом (например, наименьшим).

Комбинированный выбор. Все процессы выдают значения для записи, из них по определенному правилу формируется результат (например, сумма значений, максимальное значение и др.), который и записывается.

Классификация PRAM по методам разрешения конфликтов представлена на рис. 2.3.

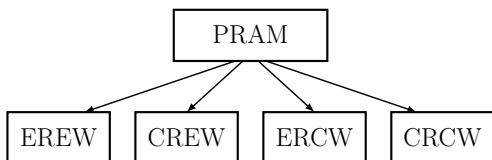


Рис. 2.3. Методы разрешения конфликтов в PRAM

Таким образом, системы EREW обладают существенными ограничениями, налагаемыми на работу с ячейками памяти. С другой стороны, системы CREW, ERCW, CRCW с большим количеством процессоров трудно построить по техническим причинам, поскольку количество вычислительных ядер, одновременно получающих доступ к некоторому участку памяти, ограничено. Однако существует важный и несколько неожиданный результат, позволяющий моделировать работу CRCW-машины на системе, построенной в соответствии с принципом EREW. Он представлен в теореме об эмуляции [94].

Теорема об эмуляции. Пусть алгоритм для CRCW-машины решает некоторую задачу с параметром размера N за время $T(N)$, используя p процессоров. Тогда существует алгоритм для той же задачи на EREW-системе с p процессорами, который может быть исполнен за время $O(T(N) \log_2 N)$. (Объем памяти PRAM должен быть увеличен в $O(p)$ раз.)

В отличие от модели RAM, основной мерой сложности алгоритмов для многопроцессорных вычислительных систем является время исполнения алгоритма. Введем обозначения: $T_1(N)$ — время, затрачиваемое последовательным алгоритмом на решение задачи, сложность которой оценивается параметром N ; $T_p(N)$ — время, затрачиваемое параллельным алгоритмом на машине с p процессорами, причем $p > 1$. Поскольку

ку, как следует из определения RAM, каждая операция требует вполне определенного времени, величина $T_1(N)$ пропорциональна числу вычислительных операций в используемом алгоритме.

Заметим, что минимальное время исполнения алгоритма наблюдается в случае $p \rightarrow \infty$. Гипотетическую вычислительную систему с бесконечно большим количеством доступных процессоров называют **паракомпьютером**. Асимптотическую сложность алгоритма для паракомпьютера обозначают через $T_\infty(N)$.

Примечание. Краткие сведения о способах математической оценки свойств алгоритмов приведены в Приложении А «Методы оценки эффективности алгоритмов».

Для анализа параллельных алгоритмов широко применяются понятия **ускорения**, **эффективности** и **стоимости**. Прежде всего следует обратить внимание на то, насколько быстрее будет решена задача по сравнению с решением на однопроцессорной машине.

Ускорение $S_p(N)$, получаемое при использовании параллельного алгоритма на машине с p процессорами, равно

$$S_p(N) = \frac{T_1(N)}{T_p(N)}.$$

Это мера прироста производительности по сравнению с *наилучшим* последовательным алгоритмом. Чем больше ускорение, тем больше отличается время решения задачи на многопроцессорной системе от продолжительности работы алгоритма на системе с одним процессором.

Эффективность $E_p(N)$ использования процессоров конкретным параллельным алгоритмом равна

$$E_p(N) = \frac{T_1(N)}{pT_p(N)} = \frac{S_p(N)}{p}.$$

Стоимость вычислений $C_p(N)$ определяется как произведение времени параллельного решения задачи и числа используемых процессоров: $C_p(N) = pT_p(N)$. **Стоимостно-оптимальный алгоритм** характеризуется стоимостью, пропорциональной сложности наилучшего последовательного алгоритма [14], и в этом случае

$$\frac{C_p(N)}{T_1(N)} = \Theta(1).$$

Пример 2.1. Время работы последовательной версии некоторого алгоритма \mathcal{A} равно $T_1(N) = 2N \log_2(N) \tau$, где N — размер входных данных, τ — время выполнения одной вычислительной операции. В предположении, что алгоритм допускает максимальное распараллеливание, т. е. время работы на вычислительной системе с p процессорами равно $T_p(N) = \frac{T_1(N)}{p}$, вычислите время работы алгоритма \mathcal{A} в случае $N = 64$, $p = 8$.

Решение.

Задача решается непосредственной подстановкой в соотношение $T_p(N) = \frac{T_1(N)}{p}$ данных из условия:

$$T_p(N) = \frac{2N \log_2(N) \tau}{p}.$$

Используя численные значения, получим

$$T_p(N) = \frac{2 \cdot 64 \log_2(64) \tau}{8} = 96\tau.$$

□

Предельные значения величин ускорения и эффективности, как непосредственно следует из их определений, равны $S_p = p$, $E_p = 1$. Максимально возможное значение S_p достигается, когда удастся равномерно распределить вычисления по всем процессорам и не требуется дополнительных действий для обеспечения взаимодействия между процессорами на этапе работы программы и для объединения результатов. Попытка увеличить ускорение, увеличив число процессоров, приведет, как правило, к уменьшению величины E_p , и наоборот. Максимальная эффективность достигается при использовании единственного процессора ($p = 1$).

Здесь не обсуждается эффект сверхлинейного ускорения [99], который заключается в том, что $S_p > p$, и может возникнуть в моделях, отличных от PRAM, по нескольким причинам:

- последовательный алгоритм, используемый для сравнения, неоптимален;
- архитектура многопроцессорной системы имеет специфические особенности, ускоряющие некоторые вычислительные операции;