

УДК 004.421  
ББК 32.972  
Р58

**Роби Р., Замора Дж.**

Р58 Параллельные и высокопроизводительные вычисления / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2022. – 800 с.: ил.

**ISBN 978-5-97060-936-1**

Параллельное программирование позволяет распределять задачи обработки данных между несколькими процессорами, существенно повышая производительность. В книге рассказывается, как с минимальными трудозатратами повысить эффективность ваших программ. Вы научитесь оценивать аппаратные архитектуры и работать со стандартными инструментами отрасли, такими как OpenMP и MPI, освоите структуры данных и алгоритмы, подходящие для высокопроизводительных вычислений, узнаете, как экономить энергию на мобильных устройствах, и даже запустите масштабную симуляцию цунами на батарее из GPU-процессоров.

Издание предназначено для опытных программистов, владеющих языком высокопроизводительных вычислений, таким как C, C++ или Fortran.

УДК 004.421  
ББК 32.972

Original English language edition published by Manning Publications USA. Russian-language edition copyright © 2021 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-6172-9646-8 (англ.)  
ISBN 978-5-97060-936-1 (рус.)

© Manning Publications, 2021  
© Перевод, оформление, издание, ДМК Пресс, 2021

# Оглавление

---

Часть I ■ ВВЕДЕНИЕ В ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ .....	36
1 ■ Зачем нужны параллельные вычисления? .....	38
2 ■ Планирование под параллелизацию .....	75
3 ■ Пределы производительности и профилирование .....	102
4 ■ Дизайн данных и модели производительности .....	134
5 ■ Параллельные алгоритмы и шаблоны .....	179
Часть II ■ CPU: ПАРАЛЛЕЛЬНАЯ РАБОЧАЯ ЛОШАДКА .....	233
6 ■ Векторизация: флопы бесплатно .....	236
7 ■ Стандарт OpenMP, который «рулит» .....	273
8 ■ MPI: параллельный становой хребет .....	328
Часть III ■ GPU: РОЖДЕННЫ ДЛЯ УСКОРЕНИЯ .....	385
9 ■ Архитектуры и концепции GPU .....	389
10 ■ Модель программирования GPU .....	430
11 ■ Программирование GPU на основе директив .....	458
12 ■ Языки GPU: обращение к основам .....	510
13 ■ Профилирование и инструменты GPU .....	558
Часть IV ■ ЭКОСИСТЕМЫ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ .....	590
14 ■ Аффинность: перемирие с вычислительным ядром .....	592
15 ■ Пакетные планировщики: наведение порядка в хаосе .....	633
16 ■ Файловые операции для параллельного мира .....	654
17 ■ Инструменты и ресурсы для более качественного исходного кода .....	691

# Содержание

Оглавление .....	6
Предисловие .....	19
Благодарности .....	24
О книге .....	26
Об авторах .....	33
Об иллюстрации на обложке .....	35

## Часть I ВВЕДЕНИЕ В ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ .....

<b>1</b>	<b>Зачем нужны параллельные вычисления? .....</b>	<b>38</b>
1.1	Почему вы должны изучить параллельные вычисления? .....	41
1.1.1	Каковы потенциальные преимущества параллельных вычислений? .....	44
1.1.2	Предостережения, связанные с параллельными вычислениями .....	47
1.2	Фундаментальные законы параллельных вычислений .....	48
1.2.1	Предел на параллельные вычисления: закон Амдала .....	48
1.2.2	Преодоление параллельного предела: закон Густафсона–Барсиса ....	49
1.3	Как работают параллельные вычисления? .....	52
1.3.1	Пошаговое ознакомление с примером приложения .....	54
1.3.2	Аппаратная модель для современных гетерогенных параллельных систем .....	60
1.3.3	Прикладная/программная модель для современных гетерогенных параллельных систем .....	64
1.4	Классифицирование параллельных подходов .....	68
1.5	Параллельные стратегии .....	69
1.6	Параллельное ускорение против сравнительного ускорения: две разные меры .....	70
1.7	Чему вы научитесь в этой книге? .....	72
1.7.1	Дополнительное чтение .....	73
1.7.2	Упражнения .....	73
	Резюме .....	74

<b>2</b>	<b>Планирование под параллелизацию</b>	75
2.1	На подступах к новому проекту: подготовка	77
2.1.1	Версионный контроль: создание безопасного хранилища для своего параллельного кода	78
2.1.2	Комплекты тестов: первый шаг к созданию устойчивого и надежного приложения	80
2.1.3	Отыскание и исправление проблем с памятью	90
2.1.4	Улучшение переносимости кода	92
2.2	Профилирование: определение разрыва между способностями системы и производительностью приложения	94
2.3	Планирование: основа успеха	94
2.3.1	Разведывательный анализ с использованием сравнительных тестов и мини-приложений	95
2.3.2	Дизайн стержневых структур данных и модульность кода	96
2.3.3	Алгоритмы: редизайн для параллельности	96
2.4	Имплементация: где все это происходит	97
2.5	Фиксация: качественное завершение работы	98
2.6	Материалы для дальнейшего изучения	99
2.6.1	Дополнительное чтение	99
2.6.2	Упражнения	100
	Резюме	100
<b>3</b>	<b>Пределы производительности и профилирование</b>	102
3.1	Знание потенциальных пределов производительности вашего приложения	103
3.2	Определение возможностей своего оборудования: сравнительное тестирование	106
3.2.1	Инструменты для сбора характеристик системы	107
3.2.2	Расчет теоретических максимальных флопов	110
3.2.3	Иерархия памяти и теоретическая пропускная способность памяти	111
3.2.4	Эмпирическое измерение пропускной способности и флопов	112
3.2.5	Расчет машинного баланса между флопами и пропускной способностью	116
3.3	Характеризация вашего приложения: профилирование	117
3.3.1	Инструменты профилирования	117
3.3.2	Эмпирическое измерение тактовой частоты и энергопотребления процессора	129
3.3.3	Отслеживание памяти во время выполнения	130
3.4	Материалы для дальнейшего изучения	131
3.4.1	Дополнительное чтение	131
3.4.2	Упражнения	131
	Резюме	132
<b>4</b>	<b>Дизайн данных и модели производительности</b>	134
4.1	Структуры данных для обеспечения производительности: дизайн с ориентацией на данные	136
4.1.1	Многомерные массивы	138
4.1.2	Массив структур (AoS) против структур из массивов (SoA)	144

4.1.3	Массив структур из массивов (AoSoA) .....	150
4.2	Три категории неуспешных обращений к кешу: вынужденное, емкостное и конфликтное .....	152
4.3	Простые модели производительности: тематическое исследование .....	157
4.3.1	Полноматричные представления данных .....	160
4.3.2	Представление сжато-разреженного хранения .....	164
4.4	Продвинутое модели производительности .....	169
4.5	Сетевые сообщения .....	173
4.6	Материалы для дальнейшего изучения .....	176
4.6.1	Дополнительное чтение .....	176
4.6.2	Упражнения .....	177
	Резюме .....	177

<b>5</b>	<b>Параллельные алгоритмы и шаблоны .....</b>	<b>179</b>
5.1	Анализ алгоритмов для приложений параллельных вычислений .....	180
5.2	Модели производительности против алгоритмической сложности .....	181
5.3	Параллельные алгоритмы: что это такое? .....	186
5.4	Что такое хеш-функция? .....	187
5.5	Пространственное хеширование: высокопараллельный алгоритм .....	189
5.5.1	Использование идеального хеширования для пространственных операций с сеткой .....	192
5.5.2	Использование компактного хеширования для пространственных операций на сетке .....	208
5.6	Шаблон префиксного суммирования (сканирования) и его важность в параллельных вычислениях .....	217
5.6.1	Операция параллельного сканирования с эффективностью шагов .....	218
5.6.2	Операция параллельного сканирования с эффективностью работы .....	219
5.6.3	Операции параллельного сканирования для крупных массивов .....	220
5.7	Параллельная глобальная сумма: решение проблемы ассоциативности .....	221
5.8	Будущие исследования параллельных алгоритмов .....	229
5.9	Материалы для дальнейшего изучения .....	229
5.9.1	Дополнительное чтение .....	230
5.9.2	Упражнения .....	231
	Резюме .....	231

<b>Часть II</b>	<b>СРУ: ПАРАЛЛЕЛЬНАЯ РАБОЧАЯ ЛОШАДКА .....</b>	<b>233</b>
-----------------	--	------------

<b>6</b>	<b>Векторизация: флопы бесплатно .....</b>	<b>236</b>
6.1	Векторизация и обзор SIMD (одна команда, несколько элементов данных) .....	237

6.2	Аппаратные тренды векторизации .....	239
6.3	Методы векторизации .....	240
6.3.1	Оптимизированные библиотеки обеспечивают производительность за счет малых усилий.....	240
6.3.2	Автоматическая векторизация: простой способ ускорения векторизации (в большинстве случаев) .....	241
6.3.3	Обучение компилятора посредством подсказок: прагмы и директивы .....	246
6.3.4	Дрянные циклы, они у нас в руках: используйте внутренние векторные функции компилятора .....	253
6.3.5	Не для слабонервных: применение ассемблерного кода для векторизации.....	259
6.4	Стиль программирования для более качественной векторизации .....	261
6.5	Компиляторные флаги, относящиеся к векторизации, для различных компиляторов .....	262
6.6	Директивы OpenMP SIMD для более качественной переносимости .....	268
6.7	Материалы для дальнейшего изучения.....	271
6.7.1	Дополнительное чтение .....	271
6.7.2	Упражнения.....	271
	Резюме .....	272

<b>7</b>	<b>Стандарт OpenMP, который «рулит» .....</b>	<b>273</b>
7.1	Введение в OpenMP .....	274
7.1.1	Концепции OpenMP .....	275
7.1.2	Простая программа стандарта OpenMP .....	278
7.2	Типичные варианты использования OpenMP: уровень цикла, высокий уровень и MPI плюс OpenMP .....	285
7.2.1	OpenMP уровня цикла для быстрой параллелизации .....	285
7.2.2	OpenMP высокого уровня для улучшенной параллельной производительности .....	286
7.2.3	MPI плюс OpenMP для максимальной масштабируемости .....	286
7.3	Примеры стандартного OpenMP уровня цикла .....	287
7.3.1	OpenMP уровня цикла: пример векторного сложения .....	288
7.3.2	Пример потоковой триады .....	292
7.3.3	OpenMP уровня цикла: стенсильный пример .....	293
7.3.4	Производительность примеров уровня цикла.....	295
7.3.5	Пример редукции на основе глобальной суммы с использованием потокообразования OpenMP .....	296
7.3.6	Потенциальные трудности OpenMP уровня цикла .....	297
7.4	Важность области видимости переменной для правильности в OpenMP .....	298
7.5	OpenMP уровня функции: придание всей функции целиком свойства поточной параллельности .....	300
7.6	Усовершенствование параллельной масштабируемости с помощью OpenMP высокого уровня.....	302
7.6.1	Как имплементировать OpenMP высокого уровня .....	303
7.6.2	Пример имплементирования OpenMP высокого уровня .....	306

7.7	Гибридное потокообразование и векторизация с OpenMP .....	309
7.8	Продвинутые примеры использования OpenMP .....	312
7.8.1	Стенсильный пример с отдельным проходом для направлений $x$ и $y$ .....	312
7.8.2	Имплементация суммирования по Кахану с потокообразованием OpenMP.....	317
7.8.3	Поточная имплементация алгоритма префиксного сканирования.....	318
7.9	Инструменты потокообразования, необходимые для устойчивых имплементаций .....	320
7.9.1	Использование профилировщика Allinea/ARM MAP для быстрого получения высокоуровневого профиля вашего приложения .....	321
7.9.2	Отыскание гоночных состояний в потоках с помощью Intel® Inspector .....	322
7.10	Пример алгоритма поддержки на основе операционных задач .....	323
7.11	Материалы для дальнейшего изучения.....	325
7.11.1	Дополнительное чтение .....	325
7.11.2	Упражнения.....	326
Резюме	.....	326

<b>8</b>	<b>MPI: параллельный становой хребет .....</b>	<b>328</b>
8.1	Основы программы MPI .....	329
8.1.1	Базовые функциональные вызовы MPI для каждой программы MPI .....	330
8.1.2	Компиляторные обертки для более простых программ MPI .....	331
8.1.3	Использование команд параллельного запуска.....	331
8.1.4	Минимально работающий пример программы MPI .....	332
8.2	Команды отправки и приемки для обмена данными «из процесса в процесс» .....	334
8.3	Коллективный обмен данными: мощный компонент MPI.....	341
8.3.1	Использование барьера для синхронизирования таймеров.....	342
8.3.2	Использование широковеЩательной передачи для манипулирования данными малого входного файла .....	343
8.3.3	Использование редукции для получения одного единственного значения из всех процессов.....	345
8.3.4	Использование операции сбора для наведения порядка в отладочных распечатках.....	349
8.3.5	Использование разброса и сбора для отправки данных процессам для работы .....	351
8.4	Примеры параллельности данных.....	353
8.4.1	Потоковая триада для измерения пропускной способности на узле .....	353
8.4.2	Обмен с прозрачными ячейками в двухмерной вычислительной сетке.....	355
8.4.3	Обмен с прозрачными ячейками в трехмерной стенсильной калькуляции.....	363
8.5	Продвинутая функциональность MPI для упрощения исходного кода и обеспечения оптимизаций.....	364

8.5.1	Использование конкретно-прикладных типов данных MPI для повышения производительности и упрощения кода.....	365
8.5.2	Поддержка декартовой топологии в MPI.....	370
8.5.3	Тесты производительности вариантов обмена с прозрачными ячейками .....	376
8.6	Гибридная техника MPI плюс OpenMP для максимальной масштабируемости .....	378
8.6.1	Преимущества гибридной техники MPI плюс OpenMP.....	378
8.6.2	Пример техники MPI плюс OpenMP .....	379
8.7	Материалы для дальнейшего изучения.....	382
8.7.1	Дополнительное чтение .....	382
8.7.2	Упражнения.....	383
	Резюме .....	383

## Часть III GPU: РОЖДЕНЫ ДЛЯ УСКОРЕНИЯ ..... 385

### 9 Архитектуры и концепции GPU..... 389

9.1	Система CPU-GPU как ускоренная вычислительная платформа .....	391
9.1.1	Интегрированные GPU: недоиспользуемая опция в товарных системах .....	393
9.1.2	Выделенные GPU: рабочая лошадка .....	393
9.2	GPU и двигатель потокообразования.....	394
9.2.1	Вычислительным модулем является потоковый мультипроцессор (или подсрез) .....	397
9.2.2	Обрабатываемыми элементами являются отдельные процессоры .....	397
9.2.3	Несколько операций, выполняемых на данных каждым обрабатываемым элементом.....	398
9.2.4	Расчет пиковых теоретических флопов для некоторых ведущих GPU .....	398
9.3	Характеристики пространств памяти GPU .....	400
9.3.1	Расчет теоретической пиковой пропускной способности памяти.....	401
9.3.2	Измерение GPU с помощью приложения STREAM Benchmark .....	402
9.3.3	Модель производительности в форме контура крыши для GPU-процессоров.....	404
9.3.4	Использование инструмента смешанного сравнительного тестирования производительности для выбора наилучшего GPU для рабочей нагрузки .....	406
9.4	Шина PCI: накладные расходы на передачу данных от CPU к GPU .....	408
9.4.1	Теоретическая пропускная способность шины PCI.....	409
9.4.2	Приложение сравнительного тестирования пропускной способности PCI.....	412
9.5	Платформы с многочисленными GPU и MPI.....	416
9.5.1	Оптимизирование перемещения данных между GPU-процессорами по сети .....	416
9.5.2	Более высокопроизводительная альтернатива шине PCI .....	418

9.6	Потенциальные преимущества платформ, ускоренных за счет GPU .....	418
9.6.1	Сокращение показателя времени до решения .....	418
9.6.2	Сокращение энергопотребления с помощью GPU-процессоров .....	420
9.6.3	Снижение в затратах на облачные вычисления за счет использования GPU-процессоров .....	426
9.7	Когда следует использовать GPU-процессоры .....	427
9.8	Материалы для дальнейшего изучения .....	428
9.8.1	Дополнительное чтение .....	428
9.8.2	Упражнения .....	429
	Резюме .....	429

## 10 Модель программирования GPU .....

10.1	Абстракции программирования GPU: широко распространенная структура .....	432
10.1.1	Массовый параллелизм .....	432
10.1.2	Неспособность поддерживать координацию среди операционных задач .....	433
10.1.3	Терминология для параллелизма GPU .....	433
10.1.4	Декомпозиция данных на независимые единицы работы: NDRange или решетка .....	434
10.1.5	Рабочие группы обеспечивают оптимальную по размеру порцию работы .....	437
10.1.6	Подгруппы, варпы или волновые фронты исполняются в унисон .....	438
10.1.7	Элемент работы: базовая единица операции .....	439
10.1.8	SIMD- или векторное оборудование .....	439
10.2	Структура кода для модели программирования GPU .....	440
10.2.1	Программирование «Эго»: концепция параллельного вычислительного ядра .....	441
10.2.2	Поточные индексы: соотношение локальной плитки с глобальным миром .....	442
10.2.3	Индексные множества .....	444
10.2.4	Как обращаться к ресурсам памяти в вашей модели программирования GPU .....	444
10.3	Оптимизирование использования ресурсов GPU .....	446
10.3.1	Сколько регистров используется в моем вычислительном ядре? .....	447
10.3.2	Занятость: предоставление большего объема работы для планирования рабочей группы .....	448
10.4	Редукционный шаблон требует синхронизации между рабочими группами .....	450
10.5	Асинхронные вычисления посредством очередей (поток операций) .....	451
10.6	Разработка плана параллелизации приложения для GPU-процессоров .....	453
10.6.1	Случай 1: трехмерная атмосферная симуляция .....	453
10.6.2	Случай 2: применение неструктурированной вычислительной сетки .....	454
10.7	Материалы для дальнейшего изучения .....	455
10.7.1	Дополнительное чтение .....	456

10.7.2 Упражнения .....	457
Резюме .....	457

<b>11 Программирование GPU на основе директив .....</b>	<b>458</b>
11.1 Процесс применения директив и прагм для имплементации на основе GPU .....	460
11.2 OpenACC: самый простой способ выполнения на вашем GPU ....	461
11.2.1 Компилирование исходного кода OpenACC .....	463
11.2.2 Участки параллельных вычислений в OpenACC для ускорения вычислений .....	465
11.2.3 Использование директив для сокращения перемещения данных между CPU и GPU.....	471
11.2.4 Оптимизирование вычислительных ядер GPU.....	476
11.2.5 Резюме результирующих производительностей для потоковой триады .....	482
11.2.6 Продвинутое техники OpenACC .....	483
11.3 OpenMP: чемпион в тяжелом весе вступает в мир ускорителей ...	486
11.3.1 Компилирование исходного кода OpenMP.....	487
11.3.2 Генерирование параллельной работы на GPU с помощью OpenMP.....	488
11.3.3 Создание участков данных для управления перемещением данных на GPU с помощью OpenMP .....	493
11.3.4 Оптимизирование OpenMP под GPU-процессоры .....	497
11.3.5 Продвинутый OpenMP для GPU-процессоров.....	502
11.4 Материалы для дальнейшего изучения.....	507
11.4.1 Дополнительное чтение .....	507
11.4.2 Упражнения.....	508
Резюме .....	509

<b>12 Языки GPU: обращение к основам .....</b>	<b>510</b>
12.1 Функциональности нативного языка программирования GPU ....	512
12.2 Языки CUDA и HIP GPU: низкоуровневая опция производительности .....	514
12.2.1 Написание и сборка вашего первого приложения на языке CUDA ....	514
12.2.2 Редукционное вычислительное ядро в CUDA: жизнь становится все сложнее .....	524
12.2.3 HIP'ифицирование исходного кода CUDA.....	531
12.3 OpenCL для переносимого языка GPU с открытым исходным кодом.....	534
12.3.1 Написание и сборка вашего первого приложения OpenCL.....	536
12.3.2 Редукции в OpenCL .....	542
12.4 SYCL: экспериментальная имплементация на C++ становится магистральной .....	546
12.5 Языки более высокого уровня для обеспечения переносимости производительности .....	550
12.5.1 Kokkos: экосистема обеспечения переносимости производительности .....	550
12.5.2 RAJA для более адаптируемого слоя обеспечения переносимости производительности .....	553

12.6	Материалы для дальнейшего изучения.....	555
12.6.1	Дополнительное чтение .....	556
12.6.2	Упражнения.....	557
Резюме	.....	557

<b>13</b>	<b>Профилирование и инструменты GPU</b> .....	558
13.1	Обзор инструментов профилирования .....	559
13.2	Как выбрать хороший рабочий поток.....	560
13.3	Образец задачи: симуляция мелководья .....	562
13.4	Образец профилировочного рабочего потока.....	566
13.4.1	Выполнение приложения симуляции мелководья.....	567
13.4.2	Профилирование исходного кода CPU для разработки плана действий.....	569
13.4.3	Добавление вычислительных директив OpenACC, чтобы начать шаг имплементации .....	571
13.4.4	Добавление директив перемещения данных.....	574
13.4.5	Направляемый анализ может дать вам несколько предлагаемых улучшений .....	575
13.4.6	Комплект инструментов NVIDIA Nsight может стать мощным подспорьем в разработке.....	577
13.4.7	CodeXL для экосистемы GPU-процессоров AMD .....	578
13.5	Не утоните в болоте: сосредотачивайтесь на важных метриках .....	579
13.5.1	Занятость: достаточно ли работы? .....	580
13.5.2	Эффективность выдачи: ваши варпы прерываются слишком часто? .....	580
13.5.3	Достигнутая пропускная способность: она всегда сводится к пропускной способности .....	581
13.6	Контейнеры и виртуальные машины обеспечивают обходные пути .....	581
13.6.1	Контейнеры Docker в качестве обходного пути.....	582
13.6.2	Виртуальные машины с использованием VirtualBox .....	585
13.7	Облачные опции: гибкие и переносимые возможности.....	587
13.8	Материалы для дальнейшего изучения.....	588
13.8.1	Дополнительное чтение .....	588
13.8.2	Упражнения.....	589
Резюме	.....	589

## Часть IV ЭКОСИСТЕМЫ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ..... 590

<b>14</b>	<b>Аффинность: перемирие с вычислительным ядром</b> .....	592
14.1	Почему важна аффинность?.....	593
14.2	Нащупывание вашей архитектуры.....	595
14.3	Аффинность потоков с OpenMP .....	597

14.4	Аффинность процессов с MPI.....	606
14.4.1	Принятое по умолчанию размещение процессов с помощью OpenMPI .....	606
14.4.2	Взятие под контроль: базовые техники специфицирования размещения процессов в OpenMPI .....	607
14.4.3	Аффинность – это больше, чем просто привязывание процессов: полная картина .....	612
14.5	Аффинность для MPI плюс OpenMP .....	615
14.6	Контроль за аффинностью из командной строки.....	620
14.6.1	Использование инструмента hwloc-bind для назначения аффинности .....	620
14.6.2	Использование likwid-pin: инструмент аффинности в комплекте инструментов likwid .....	622
14.7	Будущее: установка и изменение аффинности во время выполнения .....	625
14.7.1	Настройка аффинности в исполняемом файле .....	625
14.7.2	Изменение аффинностей процессов во время выполнения .....	627
14.8	Материалы для дальнейшего исследования .....	629
14.8.1	Дополнительное чтение .....	630
14.8.2	Упражнения.....	631
	Резюме .....	632

<b>15</b>	<b>Пакетные планировщики: наведение порядка в хаосе.....</b>	<b>633</b>
15.1	Хаос неуправляемой системы.....	634
15.2	Как не быть помехой при работе в занятом работой кластере ...	636
15.2.1	Макет пакетной системы для занятых кластеров .....	636
15.2.2	Как быть вежливым на занятых работой кластерах и сайтах HPC: распространенные любимые мозоли HPC .....	636
15.3	Отправка вашего первого пакетного скрипта .....	638
15.4	Автоматические перезапуски для длительных заданий.....	645
15.5	Указание зависимостей в пакетных скриптах.....	649
15.6	Материалы для дальнейшего исследования .....	651
15.6.1	Дополнительное чтение .....	651
15.6.2	Упражнения.....	652
	Резюме .....	652

<b>16</b>	<b>Файловые операции для параллельного мира.....</b>	<b>654</b>
16.1	Компоненты высокопроизводительной файловой системы .....	655
16.2	Стандартные файловые операции: интерфейс между параллельной и последовательной обработкой.....	657
16.3	Файловые операции MPI (MPI-IO) для более параллельного мира.....	659
16.4	HDF5 как самоописывающий формат для более качественного управления данными.....	668
16.5	Другие пакеты программно-информационного обеспечения для параллельных файлов .....	676

16.6	Параллельная файловая система: аппаратный интерфейс .....	678
16.6.1	Все, что вы хотели знать о настройке параллельного файла, но не знали, как спросить .....	678
16.6.2	Общие подсказки, применимые ко всем файловым системам .....	682
16.6.3	Подсказки, относящиеся к конкретным файловым системам .....	684
16.7	Материалы для дальнейшего исследования .....	688
16.7.1	Дополнительное чтение .....	688
16.7.2	Упражнения .....	690
	Резюме .....	690

<b>17</b>	<b>Инструменты и ресурсы для более качественного исходного кода .....</b>	<b>691</b>
17.1	Системы версионного контроля: все начинается здесь .....	694
17.1.1	Распределенный версионный контроль подходит для более мобильного мира .....	695
17.1.2	Централизованный версионный контроль для простоты и безопасности исходного кода .....	695
17.2	Таймерные процедуры для отслеживания производительности исходного кода .....	696
17.3	Профилировщики: невозможно улучшить то, что не измеряется .....	698
17.3.1	Простые тексто-ориентированные профилировщики для повседневного использования .....	699
17.3.2	Высокоуровневые профилировщики для быстрого выявления узких мест .....	700
17.3.3	Среднеуровневые профилировщики для руководства разработкой приложений .....	701
17.3.4	Детализированные профилировщики обеспечивают подробные сведения о производительности оборудования .....	703
17.4	Сравнительные тесты и мини-приложения: окно в производительность системы .....	705
17.4.1	Сравнительные тесты измеряют характеристики производительности системы .....	705
17.4.2	Мини-приложения придают приложению перспективу .....	706
17.5	Обнаружение (и исправление) ошибок памяти для устойчивого приложения .....	709
17.5.1	Инструмент Valgrind Memcheck: дублер с открытым исходным кодом .....	709
17.5.2	Dr. Memory для заболеваний вашей памяти .....	710
17.5.3	Коммерческие инструменты памяти для требовательных приложений .....	712
17.5.4	Компиляторно-ориентированные инструменты памяти для удобства .....	712
17.5.5	Инструменты проверки столбов ограждения обнаруживают несанкционированный доступ к памяти .....	713
17.5.6	Инструменты памяти GPU для устойчивых приложений GPU ....	714
17.6	Инструменты проверки потоков для определения гоночных условий .....	715
17.6.1	Intel® Inspector: инструмент обнаружения гоночных состояний с графическим интерфейсом .....	715

17.6.2	<i>Archer: тексто-ориентированный инструмент обнаружения гоночных условий</i> .....	716
17.7	<b>Устранители дефектов: отладчики для уничтожения дефектов</b> .....	718
17.7.1	<i>Отладчик TotalView широко доступен на веб-сайтах HPC</i> .....	718
17.7.2	<i>DDT – еще один отладчик, широко доступный на веб-сайтах HPC</i> .....	719
17.7.3	<i>Отладчики Linux: бесплатные альтернативы для ваших локальных потребностей разработки</i> .....	719
17.7.4	<i>Отладчики GPU способны помогать устранять дефекты GPU</i> .....	720
17.8	<b>Профилирование файловых операций</b> .....	721
17.9	<b>Менеджеры пакетов: ваш персональный системный администратор</b> .....	724
17.9.1	<i>Менеджеры пакетов для macOS</i> .....	725
17.9.2	<i>Менеджеры пакетов для Windows</i> .....	725
17.9.3	<i>Менеджер пакетов Spack: менеджер пакетов для высокопроизводительных вычислений</i> .....	726
17.10	<b>Modules: загрузка специализированных цепочек инструментов</b> .....	727
17.10.1	<i>Модули TCL: изначальная система модулей для загрузки цепочек программных инструментов</i> .....	730
17.10.2	<i>Lmod: имплементация альтернативного пакета Modules на основе Lua</i> .....	730
17.11	<b>Размышления и упражнения</b> .....	730
	<b>Резюме</b> .....	731
	<i>Приложение А Справочные материалы</i> .....	732
	<i>Приложение В Решения упражнений</i> .....	740
	<i>Приложение С Глоссарий</i> .....	765
	<i>Предметный указатель</i> .....	781