

Моей жене Пегги, которая поддерживала не только мой путь в области высокопроизводительных вычислений, но и нашего сына Джона и дочь Рейчел. Научное программирование далеко от ее медицинских знаний, но она сопровождала меня и совершила это наше путешествие с самого начала. Моему сыну Джону и дочери Рейчел, которые вновь разожгли во мне пламя, и за ваше многообещающее будущее.

– Боб Роби

Моему мужу Рику, который поддерживал меня всю дорогу, спасибо, что брал на себя утренние смены и позволял мне работать по ночам. Ты никогда не позволял мне отказываться от самой себя. Моим родителям и родственникам, спасибо за всю вашу помощь и поддержку. И моему сыну Дереку за то, что он был одним из моих самых больших вдохновителей; ты – вся причина, почему я не просто живу, я наслаждаюсь жизнью.

– Джули Замора

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: РОЖДЕНЫ 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..... 36

1 Зачем нужны параллельные вычисления?.....	38
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

2	Планирование под параллелизацию	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
3	Пределы производительности и профилирование	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
4	Дизайн данных и модели производительности	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	Продвинутые 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

5	Параллельные алгоритмы и шаблоны	179
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

Часть II	SPU: ПАРАЛЛЕЛЬНАЯ РАБОЧАЯ ЛОШАДКА	233
-----------------	--	-----

6	Векторизация: флопы бесплатно	236
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	<i>Оптимизированные библиотеки обеспечивают производительность за счет малых усилий</i>	240
6.3.2	<i>Автоматическая векторизация: простой способ ускорения векторизации (в большинстве случаев)</i>	241
6.3.3	<i>Обучение компилятора посредством подсказок: прагмы и директивы</i>	246
6.3.4	<i>Дрянные циклы, они у нас в руках: используйте внутренние векторные функции компилятора</i>	253
6.3.5	<i>Не для слабонервных: применение ассемблерного кода для векторизации</i>	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	<i>Дополнительное чтение</i>	271
6.7.2	<i>Упражнения</i>	271
	Резюме	272

7	Стандарт OpenMP, который «рулит»	273
7.1	Введение в OpenMP	274
7.1.1	<i>Концепции OpenMP</i>	275
7.1.2	<i>Простая программа стандарта OpenMP</i>	278
7.2	Типичные варианты использования OpenMP: уровень цикла, высокий уровень и MPI плюс OpenMP	285
7.2.1	<i>OpenMP уровня цикла для быстрой параллелизации</i>	285
7.2.2	<i>OpenMP высокого уровня для улучшенной параллельной производительности</i>	286
7.2.3	<i>MPI плюс OpenMP для максимальной масштабируемости</i>	286
7.3	Примеры стандартного OpenMP уровня цикла	287
7.3.1	<i>OpenMP уровня цикла: пример векторного сложения</i>	288
7.3.2	<i>Пример потоковой триады</i>	292
7.3.3	<i>OpenMP уровня цикла: стенсильный пример</i>	293
7.3.4	<i>Производительность примеров уровня цикла</i>	295
7.3.5	<i>Пример редукции на основе глобальной суммы с использованием потокообразования OpenMP</i>	296
7.3.6	<i>Потенциальные трудности OpenMP уровня цикла</i>	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	<i>Как имплементировать OpenMP высокого уровня</i>	303
7.6.2	<i>Пример имплементирования OpenMP высокого уровня</i>	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

8	MPI: параллельный становой хребет	328
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: РОЖДЕННЫ 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	430
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 (потоков 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

11	Программирование GPU на основе директив	458
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	Продвинутые 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

12	Языки GPU: обращение к основам	510
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

13	Профилерование и инструменты GPU	558
13.1	Обзор инструментов профилерования	559
13.2	Как выбрать хороший рабочий поток.....	560
13.3	Образец задачи: симуляция мелководья	562
13.4	Образец профилероочного рабочего потока.....	566
13.4.1	Выполнение приложения симуляции мелководья.....	567
13.4.2	Профилерование 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

14	Аффинность: перемирие с вычислительным ядром	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 <i>hwloc-bind</i> для назначения аффинности	620
14.6.2	Использование <i>likwid-pin</i> : инструмент аффинности в комплекте инструментов <i>likwid</i>	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

15	Пакетные планировщики: наведение порядка в хаосе.....	633
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

16	Файловые операции для параллельного мира.....	654
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	<i>Все, что вы хотели знать о настройке параллельного файла, но не знали, как спросить</i>	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

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

17.6.2	<i>Archer: тексто-ориентированный инструмент обнаружения гоночных условий</i>	716
17.7	Устранители дефектов: отладчики для уничтожения дефектов	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	Профилирование файловых операций	721
17.9	Менеджеры пакетов: ваш персональный системный администратор	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	Modules: загрузка специализированных цепочек инструментов	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	Размышления и упражнения	730
	Резюме	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

Предисловие

От авторов

Боб Роби, Лос-Аламос, Нью-Мексико

Выходить за дверь – опасно, Фродо. Выходишь на дорогу, и, если не удержаться на ногах, неизвестно, куда тебя унесет.

– Бильбо Бэггинс

Я не мог предвидеть, куда нас приведет это путешествие в параллельные вычисления. Я говорю «мы», потому что на протяжении многих лет это путешествие делили со мной многочисленные коллеги. Мое же путешествие в параллельные вычисления началось в начале 1990-х годов, когда я учился в Университете Нью-Мексико. Я написал несколько программ по динамике сжимаемой жидкости для симулирования экспериментов с ударной трубкой и запускал их в каждой системе, которая попадала мне в руки. В результате меня вместе с Брайаном Смитом (Brian Smith), Джоном Соболевски (John Sobolewski) и Фрэнком Гилфезером (Frank Gilfeather) попросили представить предложение по созданию центра высокопроизводительных вычислений. Мы выиграли грант и в 1993 году создали Центр высокопроизводительных вычислений Maui. Мое участие в проекте состояло в том, чтобы предлагать курсы и возглавлять подготовку 20 аспирантов по разработке параллельных вычислений в Университете Нью-Мексико в Альбукерке.

1990-е годы были временем становления параллельных вычислений. Я помню выступление Эла Гейста (Al Geist), одного из первых разработчиков параллельной виртуальной машины (PVM) и члена комитета по стандартам MPI¹, как говорил о готовящемся к выпуску стандарте MPI

¹ Message Passing Interface (MPI) – программный интерфейс для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. – *Прим. перев.*

(июнь 1994 года). Он сказал тогда, что он никуда не приведет, потому что он слишком сложен. Насчет сложности Эл был прав, но, несмотря на это, он взмыл ввысь, и в течение нескольких месяцев он уже применялся почти во всех параллельных приложениях. Одна из причин успеха MPI заключается в наличии готовых имплементаций. Аргоннская национальная лаборатория разрабатывала Chameleon, инструмент переносимости, который в то время транслировал языки передачи сообщений, включая P4, PVM, MPL и многие другие. Этот проект был быстро изменен на MPICH, став первой высококачественной имплементацией MPI. На протяжении более десяти лет MPI стал синонимом параллельных вычислений. Почти каждое параллельное приложение было построено поверх библиотек MPI.

Теперь давайте перенесемся в 2010 год и увидим появление графических процессоров, GPU. Как-то я наткнулся на статью в журнале *Dr. Dobbs* об использовании суммы Кахана для компенсирования единственной арифметики с одинарной точностью, доступной на GPU. Я подумал, что, возможно, этот подход мог бы помочь решить давнюю проблему параллельных вычислений, когда глобальная сумма массива меняется в зависимости от числа процессоров. Намереваясь проверить это на практике, я подумал о программном коде по гидродинамике, который мой сын Джон написал в средней школе. Он тестировал сохранение массы и энергии в задаче с течением времени и останавливался, выходя из программы, если масса изменялась более чем на указанную величину. Пока он был дома на весенних каникулах после своего первого курса в Вашингтонском университете, мы опробовали этот метод и были приятно удивлены тем, насколько улучшилось сохранение массы. Для производственных исходных кодов влияние этого простого технического приема окажется важным. Мы рассмотрим алгоритм суммирования с повышенной точностью для параллельных глобальных сумм в разделе 5.7 этой книги.

В 2011 году я организовал летний проект с тремя студентами, Нилом Дэвисом (Neal Davis), Дэвидом Николаеффом (David Nicholaeff) и Деннисом Трухильо (Dennis Trujillo), чтобы попробовать, сможем ли разработать более сложные программные коды, такие как адаптивная детализация расчетной сетки (AMR) и неструктурированные произвольные лагранжево-эйлеровы (ALE) приложения для работы на GPU. Результатом стало CLAMR, мини-приложение AMR, которое полностью работало на GPU. Большая часть приложения была легко переносима. Самой сложной его частью было определение соседа для каждой ячейки. В исходном CPU-коде использовался алгоритм k-D-дерева, но алгоритмы, основанные на дереве, трудно переносятся на GPU. Через две недели после начала летнего проекта на холмах над Лос-Аламосом вспыхнул пожар в Лас-Кончасе, и город был эвакуирован. Мы уехали в Санта-Фе, и студенты разбежались. Во время эвакуации я встретился с Дэвидом Николаеффом в центре Санта-Фе, чтобы обсудить вопрос переноса на GPU. Он предложил нам попробовать использовать алгоритм хеширования, чтобы заменить древесный исходный код отыскания соседей. В то время я наблюдал за пылающим над городом огнем, и меня не отпускало

беспокойство, что огонь доберется до моего дома. Несмотря на все эти перипетии, я согласился попробовать, и алгоритм хеширования привел к тому, что весь код был запущен на GPU. Техника хеширования была обобщена Дэвидом, моей дочерью Рейчел, когда она училась в средней школе, и мной. Эти алгоритмы хеширования формируют основу для многих алгоритмов, представленных в главе 5.

В последующие годы технические приемы компактного хеширования были разработаны Ребеккой Тамблин (Rebecka Tumblin), Питером Аренсом (Peter Ahrens) и Сарой Харце (Sara Hartse). Более сложная задача компактного хеширования для операций перекомпоновывания (remapping) на CPU и GPU была решена Джеральдом Коломом (Gerald Collom) и Колином Редманом (Colin Redman), когда они только что закончили среднюю школу. Этими прорывами в параллельных алгоритмах для GPU были устранены препятствия для выполнения многих научных приложений на GPU.

В 2016 году я начал программу летней исследовательской стажировки по параллельным вычислениям (PCSRI) в Национальной лаборатории Лос-Аламоса (LANL) вместе с моими соучредителями Хай А Нам (Hai Ah Nam) и Гейбом Рокфеллером (Gabe Rockefeller). Целью программы параллельных вычислений было решение проблемы растущей сложности систем высокопроизводительных вычислений. Программа представляет собой 10-недельную летнюю стажировку с лекциями по различным темам параллельных вычислений с последующим исследовательским проектом под руководством сотрудников Национальной лаборатории Лос-Аламоса. У нас в летней программе участвовало от 12 до 18 студентов, и многие использовали ее в качестве трамплина для своей карьеры. С помощью этой программы мы продолжаем решать некоторые новейшие задачи, стоящие перед параллельными и высокопроизводительными вычислениями.

Джули Замора, Чикагский университет, Иллинойс

Если есть книга, которую вы хотите прочитать, но она еще не написана, то вы должны ее написать.

– Тони Моррисон

Мое знакомство с параллельными вычислениями началось со слов: «Перед тем как начнете, зайдите в комнату в конце 4-го этажа и установите вот эти процессоры Knights Corner в нашем кластере». Эта просьба профессора Корнельского университета побудила меня попробовать что-то новое. То, что я считала простым делом, превратилось в бурное путешествие в высокопроизводительные вычисления. Я начала с изучения основ – с выяснения физического принципа функционирования малого кластера, поднимая 40-фунтовые сервера для работы с BIOS и выполнения моего первого приложения, а затем оптимизации этих приложений по всем установленным мной узлам.

После короткого семейного перерыва, каким бы обескураживающим он ни был, я подала заявление на научно-исследовательскую стажировку

ку. Будучи принятым в первую программу летней исследовательской стажировки по параллельным вычислениям в Нью-Мексико, я получила возможность разведать тонкости параллельных вычислений на современном оборудовании, и именно там я познакомилась с Бобом. Я пришла в восторг от повышения производительности, которое было возможно при наличии лишь небольших знаний о правильном написании параллельного кода. Я лично провела разведывательный анализ процесса написания более эффективного кода OpenMP. Мое волнение и прогресс в оптимизации приложений открыли двери для других возможностей, таких как участие в конференциях и представление моей работы на собрании группы пользователей Intel и на стенде Intel по суперкомпьютерным вычислениям. В 2017 году меня также пригласили принять участие и выступить на конференции в Салишане. Это была прекрасная возможность обменяться идеями с несколькими ведущими провидцами высокопроизводительных вычислений.

Еще одним замечательным опытом была подача заявки на участие в хакатоне GPU и участие в нем. На хакатоне мы перенесли код на OpenACC, и в течение недели этот код был ускорен в 60 раз. Вы только подумайте – расчет, который раньше занимал месяц, теперь может выполняться за одну ночь. Полностью погрузившись в потенциал долгосрочных исследований, я подала заявление в аспирантуру и выбрала Чикагский университет, осознавая его тесную связь с Аргоннской национальной лабораторией. В Чикагском университете меня консультировали Ян Фостер (Ian Foster) и Генри Хоффман (Henry Hoffmann).

Из своего опыта я поняла, насколько ценным является личное взаимодействие, когда учишься писать параллельный код. Я также была разочарована отсутствием учебника или справочника, в котором обсуждалось бы текущее оборудование. В целях восполнения этого пробела мы написали эту книгу, чтобы сделать ее намного проще для тех, кто только начинает свою деятельность в параллельных и высокопроизводительных вычислениях. Решение задачи создания и преподавания введения в информатику для поступающих студентов Чикагского университета помогло мне получить представление о тех, кто знакомится с этой областью впервые. С другой стороны, объяснение технических приемов параллельного программирования в качестве ассистента преподавателя в курсе «Продвинутые распределенные системы» позволило мне работать со студентами с более высоким уровнем понимания. Оба этих опыта помогли мне обрести способность объяснять сложные темы на разных уровнях.

Я считаю, что у каждого должна быть возможность изучить этот важный материал по написанию высокопроизводительного кода и что он должен быть легко доступен для всех. Мне посчастливилось иметь наставников и советников, которые направляли меня по правильным ссылкам на веб-сайты или передавали мне свои старые рукописи для чтения и изучения. Хотя некоторые технические приемы, возможно, будут сложными, более серьезной проблемой является отсутствие согласованной документации или доступа к ведущим ученым в этой области в качестве наставников. Я понимаю, что не у всех есть одинаковые ре-

сурсы, и поэтому я надеюсь, что создание этой книги заполнит пустоту, которая существует в настоящее время.

Как мы пришли к написанию этой книги

Начиная с 2016 года команда ученых LANL во главе с Бобом Роби разработала лекционные материалы для летней исследовательской стажировки по параллельным вычислениям (PCSRI) в Лос-Аламосской национальной лаборатории (LANL). Большая часть этого материала посвящена новейшему оборудованию, которое быстро выходит на рынок. Параллельные вычисления меняются быстрыми темпами, и к ним прилагается мало документации. Книга, охватывающая эти материалы, была явно необходима. Именно в этот момент издательство Manning связалось с Бобом по поводу написания книги о параллельных вычислениях. У нас был лишь черновой набросок материалов, и работа над ним вылилась в двухлетние усилия по переводу всего этого в формат высокого качества.

Темы и план глав были четко определены на ранней стадии и основывались на лекциях для нашей летней программы. Многие идеи и технические приемы были позаимствованы из более широкого сообщества высокопроизводительных вычислений, поскольку мы стремимся к более высокому уровню вычислений – тысячекратному повышению производительности вычислений по сравнению с предыдущим эталоном петафлопсного масштаба. Это сообщество включает Центры передового опыта Министерства энергетики (DOE), проект по экзомасштабным вычислениям (Exascale Computing Project) и серию семинаров по производительности, переносимости и производительности. Широта и глубина материалов наших компьютерных лекций отражают глубокие проблемы сложных гетерогенных вычислительных архитектур.

Мы называем материал этой книги «глубоким введением». Она начинается с основ параллельных и высокопроизводительных вычислений, но без знания вычислительной архитектуры невозможно достичь оптимальной производительности. По ходу дела мы стараемся осветить проблематику на более глубоком уровне понимания, потому что недостаточно просто двигаться по тропе, не имея ни малейшего представления о том, где вы находитесь или куда направляетесь. Мы предоставляем инструменты для разработки карты и для того, чтобы показать удаленность цели, к которой мы стремимся.

В начале этой книги Джо Шоновер (Joe Schoonover) был привлечен для написания материалов, связанных с GPU, а Джули Замора – главы по OpenMP. Джо предоставил дизайн и компоновку разделов по GPU, но ему пришлось быстро отказаться. Джули написала статьи и предоставила массу иллюстраций о том, как OpenMP вписывается в этот дивный новый мир масштабных вычислений, поэтому указанный материал особенно хорошо подошел для главы книги об OpenMP. Глубокое понимание Джули проблем, связанных с экзомасштабными вычислениями, и ее способность разбирать их по полочкам для новичков в этой области стали решающим вкладом в создание этой книги.

Благодарности

Мы хотели бы поблагодарить всех, кто помог сформировать эту книгу. Первым в нашем списке стоит Джо Шоновер из Fluid Dynamics (Гидродинамики жидкости), который сделал все возможное в преподавании параллельных вычислений, в особенности с GPU-процессорами. Джо был одним из соруководителей нашей программы параллельных вычислений и сыграл важную роль в формулировании того, что должна охватывать эта книга. Другие наши соучредители, Хай А Нам, Гейб Рокфеллер, Крис Гарретт, Юнмо Ку, Люк Ван Рокель, Роберт Берд, Джонас Липпунер и Мэтт Тернер, внесли свой собственный вклад в успех школы параллельных вычислений и ее содержание. Создание летней программы параллельных вычислений не произошло бы без поддержки и видения директора института, Стефана Эйденбенца. Также спасибо Скотту Раннелсу и Даниэлю Израэлю, которые возглавили летнюю школу вычислительной физики LANL и стали пионерами концепции школы, дав нам модель для подражания.

Нам повезло, что нас окружают эксперты в области параллельных вычислений и книгоиздания. Благодарим Кейт Боуман, чей опыт в написании книг помогал вносить изменения в первые главы. Кейт – невероятно талантлива во всех аспектах издательского дела и уже много лет занимается составлением предметных указателей книг. У нас также были неофициальные отзывы от сына Боба Джона, дочери Рэйчел и зятя Боба Берда, часть технической работы каждого из которых упоминается в книге. Муж Джули, Рик, помог предоставить экспертные знания по некоторым темам, а Дов Шлахтер рассмотрел некоторые ранние наброски и предоставил несколько полезных отзывов.

Мы также хотели бы отметить опыт сотрудников, которые приложили свои знания в конкретных главах. Сюда входят Рао Гаримелла и Шейн Фогерти из Лос-Аламоса и Мэтт Мартино из Национальной лаборатории Лоуренса Ливермора, чья работа включена в главу 4. Особая благодарность выражается инновационным работам упомянутых ранее многих студентов, чья работа занимает большую часть главы 5. Рон Грин из Intel в течение нескольких лет руководил усилиями по документированию

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru