

ОГЛАВЛЕНИЕ

Введение	16
Глава 1. Общая характеристика параллельных вычислительных систем	20
1.1. Классификация вычислительных систем	21
1.1.1. Мультипроцессоры	22
1.1.2. Мультикомпьютеры	24
1.2. Показатели эффективности параллельного алгоритма	26
1.3. Краткий обзор главы	28
1.4. Обзор литературы	29
1.5. Контрольные вопросы	29
1.6. Задачи и упражнения	30
Глава 2. Основы параллельного программирования	31
2.1. Основные понятия	31
2.1.1. Концепция процесса	31
2.1.2. Определение потока	33
2.1.3. Понятие ресурса	34
2.1.4. Организация параллельных программ как системы потоков	34
2.2. Взаимодействие и взаимоисключение потоков	38
2.2.1. Разработка алгоритма взаимоисключения	39
2.2.2. Семафоры	45
2.2.3. Мониторы	46
2.3. Синхронизация потоков	49
2.3.1. Условные переменные	50
2.3.2. Барьерная синхронизация	51
2.4. Взаимоблокировка потоков	51
2.4.1. Условные переменные	53
2.4.2. Описание возможных изменений состояния программы	55
2.4.3. Обнаружение и исключение тупиков	56

2.5. Классические задачи синхронизации	56
2.5.1. Задача «Производители – Потребители»	57
2.5.2. Задача «Читатели – Писатели»	59
2.5.3. Задача «Обедающие философы»	61
2.5.4. Задача «Спящий парикмахер»	64
2.6. Методы повышения эффективности параллельных программ	66
2.6.1. Оптимизация количества потоков	67
2.6.2. Минимизация взаимодействия потоков	68
2.6.3. Оптимизация работы с памятью	69
2.6.4. Использование потоко-ориентированных библиотек	70
2.7. Краткий обзор главы	71
2.8. Обзор литературы	72
2.9. Контрольные вопросы	73
2.10. Задачи и упражнения	74

Часть I. Базовые технологии параллельного программирования

Глава 3. Параллельное программирование с использованием OpenMP	76
3.1. Основы технологии OpenMP	80
3.1.1. Понятие параллельной программы	80
3.1.2. Организация взаимодействия параллельных потоков	81
3.1.3. Структура OpenMP	82
3.1.4. Формат директив OpenMP	83
3.2. Выделение параллельно выполняемых фрагментов программного кода	83
3.2.1. Директива <code>parallel</code> для определения параллельных фрагментов	83
3.2.2. Пример первой параллельной программы	84
3.2.3. Основные понятия параллельной программы: фрагмент, область, секция	85
3.2.4. Параметры директивы <code>parallel</code>	86
3.2.5. Определение времени выполнения параллельной программы	86
3.3. Распределение вычислительной нагрузки между потоками (распараллеливание по данным для циклов)	87
3.3.1. Управление распределением итераций цикла между потоками	91
3.3.2. Управление порядком выполнения вычислений	92

3.3.3. Синхронизация вычислений по окончании выполнения цикла	93
3.3.4. Введение условий при определении параллельных фрагментов (параметр <code>if</code> директивы <code>parallel</code>)	94
3.4. Управление данными для параллельно выполняемых потоков	95
3.4.1. Определение общих и локальных переменных	95
3.4.2. Совместная обработка локальных переменных (операция редукции)	96
3.5. Организация взаимоисключения при использовании общих переменных.	98
3.5.1. Обеспечение атомарности (неделимости) операций	98
3.5.2. Использование критических секций.	100
3.5.3. Применение переменных семафорного типа (замков)	101
3.6. Распределение вычислительной нагрузки между потоками (распараллеливание по задачам при помощи директивы <code>sections</code>)	103
3.7. Расширенные возможности OpenMP.	106
3.7.1. Определение однопоточковых участков для параллельных фрагментов (директивы <code>single</code> и <code>master</code>)	106
3.7.2. Выполнение барьерной синхронизации (директива <code>barrier</code>)	107
3.7.3. Синхронизация состояния памяти (директива <code>flush</code>)	107
3.7.4. Определение постоянных локальных переменных потоков (директива <code>threadprivate</code> и параметр <code>copyin</code> директивы <code>parallel</code>)	108
3.7.5. Управление количеством потоков	110
3.7.6. Задание динамического режима при создании потоков	111
3.7.7. Управление вложенностью параллельных фрагментов	111
3.8. Дополнительные сведения	112
3.8.1. Разработка параллельных программ с использованием OpenMP на языке Fortran	112
3.8.2. Сохранение возможности последовательного выполнения программы	114
3.8.3. Краткий перечень компиляторов с поддержкой OpenMP	115
3.9. Краткий обзор главы	116
3.10. Обзор литературы	118
3.11. Контрольные вопросы.	118
3.12. Задачи и упражнения.	120

Глава 4. Параллельное программирование на основе MPI	126
4.1. MPI: основные понятия и определения	128
4.1.1. Понятие параллельной программы	128
4.1.2. Операции передачи данных	129
4.1.3. Понятие коммутаторов	129
4.1.4. Типы данных	130
4.1.5. Виртуальные топологии	130
4.2. Введение в разработку параллельных программ с использованием MPI	131
4.2.1. Основы MPI	131
4.2.2. Первая параллельная программа с использованием MPI	136
4.2.3. Определение времени выполнения MPI-программы	139
4.2.4. Начальное знакомство с коллективными операциями передачи данных	140
4.3. Операции передачи данных между двумя процессами	147
4.3.1. Режимы передачи данных	147
4.3.2. Организация неблокирующих обменов данными между процессорами	149
4.3.3. Одновременное выполнение передачи и приема	151
4.4. Коллективные операции передачи данных	152
4.4.1. Обобщенная передача данных от одного процесса всем процессам	152
4.4.2. Обобщенная передача данных от всех процессов одному процессу	154
4.4.3. Общая передача данных от всех процессов всем процессам	155
4.4.4. Дополнительные операции редукции данных	156
4.4.5. Сводный перечень коллективных операций данных	157
4.5. Производные типы данных в MPI	158
4.5.1. Понятие производного типа данных	159
4.5.2. Способы конструирования производных типов данных	161
4.5.3. Объявление производных типов и их удаление	165
4.5.4. Формирование сообщений при помощи упаковки и распаковки данных	165
4.6. Управление группами процессов и коммутаторами	168
4.6.1. Управление группами	169
4.6.2. Управление коммутаторами	170
4.7. Виртуальные топологии	172

4.7.1. Декартовы топологии (решетки)	173
4.7.2. Топологии графа	176
4.8. Дополнительные сведения о MPI	178
4.8.1. Разработка параллельных программ с использованием MPI на алгоритмическом языке Fortran	178
4.8.2. Общая характеристика среды выполнения MPI-программ	179
4.8.3. Дополнительные возможности стандарта MPI-2	180
4.9. Краткий обзор главы	181
4.10. Обзор литературы	183
4.11. Контрольные вопросы	183
4.12. Задачи и упражнения	184

Часть II. Технологии на основе параллельного расширения существующих языков программирования

Глава 5. UPC: расширение языка C для параллельного программирования	187
5.1. Понятие параллельной программы	189
5.2. Управление общими данными	190
5.3. Управление вычислениями	194
5.4. Синхронизация параллельно выполняемых потоков	196
5.4.1. Модели синхронизации доступа к общей памяти	197
5.4.2. Барьерная синхронизация	198
5.4.3. Синхронизация при помощи семафоров	199
5.5. Коллективные операции для управления данными	200
5.5.1. Операции копирования и инициализации	201
5.5.2. Коллективные операции передачи данных	202
5.5.3. Коллективные операции обработки данных	203
5.6. Использование указателей в UPC	204
5.7. Пример параллельной программы на UPC	207
5.8. Оценка эффективности параллельных программ на UPC	210
5.9. Краткий обзор главы	214
5.10. Обзор литературы	216
5.11. Контрольные вопросы	216
5.12. Задачи и упражнения	217

Глава 6. CAF: расширение языка Fortran для параллельного программирования	219
6.1. Модель параллельного программирования	219
6.1.1. Управление вычислениями.	219
6.1.2. Первая программа на языке Co-Array Fortran	220
6.2. Управление общими данными.	222
6.2.1. Одномерные распределенные массивы	225
6.2.2. Многомерные распределенные массивы	226
6.2.3. Определение положения исполнителя программы.	228
6.2.4. Динамические распределенные массивы	230
6.3. Синхронизация выполнения исполнителей программы	230
6.4. Ввод/вывод	232
6.5. Стандарт CAF 2.0	232
6.6. Примеры программ на языке Co-Array Fortran	235
6.6.1. Вычисление значения числа π методом Монте-Карло	235
6.6.2. Умножение матриц	238
6.9. Анализ эффективности CAF программ на примере набора тестов HPV.	243
6.10. Краткий обзор главы	244
6.11. Обзор литературы	246
6.12. Контрольные вопросы.	246
6.13. Задачи и упражнения.	246

Часть III. Новые языки параллельного программирования

Глава 7. Новый язык параллельного программирования Chapel	248
7.1. Основы языка	250
7.1.1. Загрузка и установка Chapel	250
7.1.2. Первая программа на языке Chapel	250
7.1.3. Основные элементы языка	251
7.1.4. Типы данных и потоки управления	256
7.1.5. Структура программы на языке Chapel	259
7.2. Параллелизм по данным.	266
7.2.1. Регулярные домены и массивы.	267
7.2.2. Операции редукции и сканирования	271
7.2.4. Переменные синхронизации.	272
7.2.4. Управление параметрами параллельного запуска	273

7.2.5.	Примеры параллельных программ на языке Chapel.	274
7.2.6.	Нерегулярные домены	283
7.3.	Параллелизм задач.	285
7.3.1.	Создание одиночных задач: оператор begin	285
7.3.2.	Одновременное создание нескольких задач: операторы sobegin и soforall	286
7.3.3.	Синхронизация задач	287
7.3.4.	Пример программы, использующей параллелизм задач	289
7.4.	Управление распределенностью вычислений	292
7.4.1.	Понятие исполнителя (locale)	292
7.4.2.	Распределение доменов между исполнителями.	294
7.4.3.	Управления исполнителями	297
7.5.	Анализ эффективности Chapel-программ на примере набора тестов NPCC.	299
7.5.1.	Тест STREAM Triad	300
7.5.2.	Тест Random Access	303
7.6.	Краткий обзор главы	306
7.7.	Обзор литературы	307
7.8.	Контрольные вопросы.	307
7.9.	Задачи и упражнения.	309
Глава 8.	Новый язык параллельного программирования X10	310
8.1.	Основы подхода.	311
8.1.1.	Понятие исполнителя (place).	311
8.1.2.	Определение активности (activity).	312
8.1.3.	Первая программа на языке X10	312
8.1.4.	Загрузка и установка X10	313
8.2.	Управление данными	313
8.2.1.	Переменные	313
8.2.2.	Индексы (points)	314
8.2.3.	Регионы (regions)	315
8.2.4.	Массивы.	316
8.3.	Управление параллельными вычислениями	316
8.3.1.	Создание активностей (async)	317
8.3.2.	Ожидание завершения активностей (finish)	317
8.3.3.	Обеспечение неделимости (атомарности) действий (atomic)	318
8.3.4.	Организация таймерной синхронизации (clock)	318
8.3.5.	Создание функциональных активностей (future)	320

8.3.6. Выполнение неделимых (атомарных) действий с предусловием (when)	321
8.4. Управление распределенными массивами	321
8.4.1. Создание распределений	322
8.4.2. Определение распределенных массивов	322
8.4.3. Доступ к элементам распределенных массивов	322
8.4.4. Организация распределенных вычислений	323
8.5. Примеры параллельных программ на языке X10	324
8.6. Анализ эффективности X10-программ на примере набора тестов HPCC	328
8.7. Краткий обзор главы	329
8.8. Обзор литературы	333
8.9. Контрольные вопросы	333
8.10. Задачи и упражнения	334

Часть IV. Технологии параллельного программирования для графических процессоров

Глава 9. CUDA: технология параллельного программирования для графических процессоров	336
9.1. Возможные преимущества вычислений на графическом процессоре	337
9.2. Средства разработки для графического процессора	339
9.2.1. Интерфейсы программирования графики и шейдерные языки	339
9.2.2. Специализированные средства от производителей	341
9.2.3. Сторонние специализированные средства	342
9.2.4. OpenCL – открытый стандарт параллельных вычислений на гетерогенных системах	344
9.3. Использование технологии CUDA для вычислений	345
9.3.1. Установка и настройка программного обеспечения	346
9.3.2. Модель программирования CUDA	349
9.3.3. Интерфейс программирования CUDA	353
9.3.4. Общие принципы вычислений на базе CUDA	360
9.4. Исследование производительности в задаче N тел	363
9.5. Краткий обзор главы	364
9.6. Обзор литературы	366
9.7. Контрольные вопросы	367
9.8. Задачи и упражнения	368

Глава 10. OpenCL – открытый стандарт параллельного программирования для гетерогенных систем	369
10.1. Архитектура OpenCL	369
10.1.1. Модель платформы	369
10.1.2. Модель исполнения	370
10.1.3. Модель памяти	373
10.1.4. Модель программирования	375
10.2. Общие принципы вычислений на базе OpenCL	375
10.3. Перенос приложения CUDA на OpenCL	386
10.4. Исследование производительности в задаче N тел.	386
10.5. Одновременное использование центральных и графических процессоров для вычислений	388
10.6. Заключение	390
10.7. Краткий обзор главы	391
10.8. Обзор литературы	392
10.9. Контрольные вопросы	392
10.10. Задачи и упражнения	393
Список литературы	394