

Оглавление

Предисловие.....	18
Благодарности.....	22
Об этой книге.....	24
Кому следует прочесть эту книгу.....	24
Структура издания: дорожная карта.....	25
О коде.....	27
От издательства.....	27
Об авторе.....	28
Об иллюстрации на обложке.....	29

Часть I. Преимущества функционального программирования в применении к конкурентным программам

Глава 1. Основы функциональной конкурентности.....	33
1.1. Что вы узнаете из этой книги.....	35
1.2. Начнем с терминологии.....	36
1.2.1. При последовательном программировании задачи выполняются одна за другой.....	37
1.2.2. При конкурентном программировании выполняется несколько задач в одно и то же время.....	38
1.2.3. При параллельном программировании выполняется несколько задач одновременно.....	39
1.2.4. При многозадачности выполняется несколько задач одновременно.....	41
1.2.5. Многопоточность как средство повышения производительности.....	42
1.3. Зачем нужна конкурентность.....	44

1.4.	Ловушки параллельного программирования.....	47
1.4.1.	Риски конкурентности.....	48
1.4.2.	Эволюция разделяемых состояний.....	51
1.4.3.	Простой пример из практики: параллельная быстрая сортировка	52
1.4.4.	Бенчмаркинг в F#	56
1.5.	Почему для конкурентности выбирают функциональное программирование	57
1.6.	Область применения функциональной парадигмы.....	61
1.7.	Зачем использовать F# и C# для функционального конкурентного программирования	62
	Резюме	65
Глава 2.	Технологии функционального программирования для конкурентных систем...	66
2.1.	Использование компоновки функций для решения сложных задач.....	67
2.1.1.	Функциональная компоновка в C#	68
2.1.2.	Функциональная компоновка в F#	70
2.2.	Использование замыканий для упрощения функционального мышления.....	71
2.2.1.	Захваченные переменные в замыканиях с лямбда-выражениями	72
2.2.2.	Замыкания в многопоточной среде	75
2.3.	Технология мемоизации и кэширования для ускорения программы	77
2.4.	Применение мемоизации для создания быстрого веб-бота	81
2.5.	«Ленивая» мемоизация для повышения производительности.....	85
2.6.	Эффективная конкурентная упреждающая обработка для уменьшения издержек на затратные вычисления.....	87
2.6.1.	Предварительные вычисления с естественной поддержкой функционального программирования	89
2.6.2.	Пусть победит быстрее!	91
2.7.	Лень — это хорошо	92
2.7.1.	Использование строгих языков программирования для лучшего понимания конкурентного поведения	92
2.7.2.	«Ленивое» кэширование и потокобезопасный шаблон «Одиночка»	94
2.7.3.	Поддержка «ленивых» вычислений в F#.....	96
2.7.4.	Мощное сочетание Lazy и Task.....	96
	Резюме	98
Глава 3.	Функциональные структуры данных и неизменяемость.....	99
3.1.	Практический пример: охота на потокобезопасный объект.....	100
3.1.1.	Неизменяемые коллекции .NET: надежное решение.....	104
3.1.2.	Конкурентные коллекции .NET: более быстрое решение.....	109
3.1.3.	Шаблон агента передачи сообщений: более быстрое и верное решение.....	112

3.2.	Безопасное совместное использование потоками функциональных структур данных.....	114
3.3.	Наконец-то неизменяемость!.....	115
3.3.1.	Функциональная структура данных для обеспечения их параллелизма.....	119
3.3.2.	Влияние неизменяемости на производительность.....	119
3.3.3.	Неизменяемость в C#	120
3.3.4.	Неизменяемость в F#.....	123
3.3.5.	Функциональные списки: объединение ячеек в цепочки.....	124
3.3.6.	Построение персистентной структуры данных: неизменяемое бинарное дерево	131
3.4.	Рекурсивные функции: естественный способ итерирования	134
3.4.1.	Хвост правильной рекурсивной функции: оптимизация хвостового вызова	135
3.4.2.	Оптимизация рекурсивной функции в стиле передачи продолжений	137
	Резюме.....	142

Часть II. Конкурентная программа: разные части, разные подходы

Глава 4.	Основы обработки больших данных: распараллеливание данных, часть 1	145
4.1.	Что такое распараллеливание данных.....	146
4.1.1.	Распараллеливание данных и задач.....	147
4.1.2.	Концепция «естественной параллельности»	148
4.1.3.	Поддержка распараллеливания данных в .NET	149
4.2.	Шаблон Fork/Join: параллельный алгоритм Манделъброта	150
4.2.1.	Когда узким местом является сборка мусора: структуры и объекты класса	156
4.2.2.	Оборотная сторона параллельных циклов	159
4.3.	Измерение производительности	160
4.3.1.	Определение предела повышения эффективности по закону Амдала.....	161
4.3.2.	Закон Густафсона: еще один шаг вперед в измерении повышения производительности	162
4.3.3.	Ограничения параллельных циклов: сумма простых чисел	162
4.3.4.	Что может пойти не так в простом цикле.....	164
4.3.5.	Модель декларативного параллельного программирования	166
	Резюме.....	168

Глава 5. PLINQ и MapReduce: распараллеливание данных, часть 2	169
5.1. Краткое введение в PLINQ.....	170
5.1.1. Почему язык PLINQ более функциональный.....	171
5.1.2. PLINQ и чистые функции: параллельный счетчик слов	172
5.1.3. Исключение побочных эффектов посредством чистых функций.....	174
5.1.4. Изоляция и контроль побочных эффектов: рефакторинг параллельного счетчика слов	176
5.2. Агрегирование и сокращение данных в параллельных программах	177
5.2.1. Усечение: одно из многих преимуществ свертки.....	180
5.2.2. Свертки в PLINQ: функции агрегирования	182
5.2.3. Реализация параллельной функции Reduce на PLINQ	188
5.2.4. Параллельное списковое включение в F#: PSeq	191
5.2.5. Параллельные массивы в F#.....	192
5.3. Параллельный шаблон MapReduce	193
5.3.1. Функции Map и Reduce.....	195
5.3.2. Использование MapReduce совместно с галереей пакетов NuGet	196
Резюме	203
Глава 6. Потоки событий реального времени: функциональное реактивное программирование	204
6.1. Реактивное программирование: обработка больших событий.....	206
6.2. Инструментарий .NET для реактивного программирования.....	209
6.2.1. Улучшенное решение: комбинаторы событий	210
6.2.2. Совместимость .NET с комбинаторами F#	211
6.3. Реактивное программирование в .NET: реактивные расширения (Rx)	214
6.3.1. От LINQ/PLINQ к Rx	217
6.3.2. IObservable и IEnumerable: дуальные интерфейсы	218
6.3.3. Реактивные расширения в действии	219
6.3.4. Поточковая передача в реальном времени с помощью Rx	220
6.3.5. От событий к наблюдаемым объектам F#	221
6.4. Укрощение потока событий: анализ эмоций в Twitter посредством Rx-программирования	222
6.5. Шаблон «издатель — подписчик» в Rx.....	232
6.5.1. Использование типа Subject для создания мощного концентратора в шаблоне «издатель — подписчик»	233
6.5.2. Rx и конкурентность	234
6.5.3. Реализация на Rx многоразового шаблона «издатель — подписчик»	235
6.5.4. Анализ эмоций твитов с помощью Rx-класса Pub-Sub.....	237
6.5.5. Наблюдатели в действии	240
6.5.6. Удобное выражение объектов в F#	240
Резюме	242

Глава 7. Функциональный параллелизм на основе задач.....	243
7.1. Краткое введение в параллелизм.....	244
7.1.1. Зачем нужны параллелизм задач и функциональное программирование.....	245
7.1.2. Поддержка параллелизма задач в .NET.....	246
7.2. Библиотека параллельных задач в .NET.....	248
7.2.1. Выполнение параллельных операций с помощью Parallel.Invoke из библиотеки TPL.....	250
7.3. Проблема void в C#.....	253
7.4. Стиль продолжений: функциональный поток управления.....	256
7.4.1. Зачем нужен CPS.....	256
7.4.2. Ожидание завершения задачи: модель продолжения.....	258
7.5. Стратегии компоновки операций для выполнения задач.....	263
7.5.1. Использование математических шаблонов для оптимальной компоновки.....	265
7.5.2. Рекомендации по использованию задач.....	271
7.6. Параллельный функциональный шаблон конвейера.....	271
Резюме.....	278
Глава 8. Асинхронность задач — путь к победе.....	279
8.1. Модель асинхронного программирования (APM).....	280
8.1.1. В чем ценность асинхронного программирования.....	281
8.1.2. Асинхронное программирование и масштабируемость.....	284
8.1.3. Операции с ограничениями процессора и ввода-вывода.....	285
8.2. Неограниченный параллелизм при асинхронном программировании.....	286
8.3. Поддержка асинхронности в .NET.....	287
8.3.1. Асинхронное программирование нарушает структуру кода.....	290
8.3.2. Асинхронное программирование на основе событий.....	291
8.4. Асинхронное программирование на основе задач в C#.....	291
8.4.1. Анонимные асинхронные лямбда-функции.....	295
8.4.2. Монадический контейнер Task<T>.....	296
8.5. Асинхронное программирование на основе задач: практический пример.....	299
8.5.1. Отмена асинхронных операций.....	304
8.5.2. Асинхронная компоновка на основе задач с монадическим оператором Bind.....	307
8.5.3. Отсрочка асинхронного вычисления, обеспечивающая компоновку.....	309
8.5.4. Повторная попытка в случае, если что-то пошло не так.....	310
8.5.5. Обработка ошибок в асинхронных операциях.....	312
8.5.6. Асинхронная параллельная обработка изменений фондового рынка.....	314
8.5.7. Асинхронная параллельная обработка данных фондового рынка после завершения выполнения задач.....	315
Резюме.....	317

Глава 9. Асинхронное функциональное программирование на F#	318
9.1. Аспекты асинхронного функционального программирования	319
9.2. Что такое асинхронный рабочий процесс F#	319
9.2.1. Стиль прохождения продолжений в вычислительных выражениях.....	320
9.2.2. Асинхронный рабочий процесс в действии: параллельные операции сохранения данных из Azure Blob	322
9.3. Асинхронные вычислительные выражения	328
9.3.1. Различия между вычислительными выражениями и монадами.....	330
9.3.2. AsyncRetry: построение собственных вычислительных выражений.....	331
9.3.3. Расширение асинхронного рабочего процесса	334
9.3.4. Отображение асинхронных операций: функтор Async.map.....	335
9.3.5. Распараллеливание асинхронных рабочих процессов: Async.Parallel	337
9.3.6. Поддержка отмены асинхронного рабочего процесса	343
9.3.7. Управление параллельными асинхронными операциями.....	345
Резюме	349
Глава 10. Функциональные комбинаторы для быстрого конкурентного программирования	350
10.1. Поток выполнения не всегда проходит по «счастливому пути»: обработка ошибок.....	351
10.2. Комбинаторы ошибок: Retry, Otherwise и Task.Catch в C#	355
10.2.1. Обработка ошибок в функциональном программировании: исключения при управлении потоком выполнения.....	358
10.2.2. Обработка ошибок с помощью Task<Option<T>> в C#	360
10.2.3. Тип AsyncOption в F#: сочетание Async и Option	361
10.2.4. Функциональная асинхронная обработка ошибок, свойственная F#	362
10.2.5. Сохранение семантики исключений с помощью типа Result	364
10.3. Обработка исключений при асинхронных операциях.....	368
10.3.1. Моделирование обработки ошибок в F# с помощью Async и Result.....	373
10.3.2. Расширение типа F# AsyncResult посредством операторов монадического связывания	374
10.4. Абстрагирование операций с функциональными комбинаторами.....	379
10.5. Коротко о функциональных комбинаторах	380
10.5.1. Встроенные асинхронные комбинаторы TPL.....	381
10.5.2. Использование комбинатора Task.WhenAny для избыточности и чередования	382
10.5.3. Использование комбинатора Task.WhenAll в асинхронном цикле for-each.....	384
10.5.4. Обзор математических шаблонов: что мы уже знаем?	385

10.6.	Окончательный вариант параллельной компоновки аппликативного функтора.....	389
10.6.1.	Расширение асинхронного рабочего процесса F# с помощью операторов аппликативных функторов	396
10.6.2.	Семантика аппликативных функторов и инфиксных операторов в F#.....	398
10.6.3.	Использование аппликативных функторов в гетерогенных параллельных вычислениях.....	399
10.6.4.	Компоновка и выполнение гетерогенных параллельных вычислений	401
10.6.5.	Управление потоком с помощью условных асинхронных комбинаторов	404
10.6.6.	Как работают асинхронные комбинаторы	408
	Резюме	410
Глава 11.	Реактивное программирование с использованием агентов.....	411
11.1.	Что такое реактивное программирование и чем оно полезно	413
11.2.	Программная модель асинхронной передачи сообщений.....	415
11.2.1.	Передача сообщений и неизменяемость	417
11.2.2.	Естественная изоляция	417
11.3.	Что такое агент.....	418
11.3.1.	Компоненты агента.....	419
11.3.2.	Что может делать агент	420
11.3.3.	Подход без разделения ресурсов для конкурентного программирования без блокировок.....	420
11.3.4.	Как функционирует агентное программирование.....	422
11.3.5.	Агент является объектно-ориентированным.....	422
11.4.	Агенты в F#: MailboxProcessor	423
11.5.	Как избежать узких мест при обращении к базе данных с помощью F#-типа MailboxProcessor.....	426
11.5.1.	Тип сообщения MailboxProcessor: размеченные объединения.....	429
11.5.2.	Двусторонний обмен данными посредством MailboxProcessor	430
11.5.3.	Использование AgentSQL из C#	431
11.5.4.	Распараллеливание рабочего процесса и координация групп агентов	433
11.5.5.	Как обрабатывать ошибки на F# с помощью MailboxProcessor.....	435
11.5.6.	Остановка агентов MailboxProcessor — CancellationToken.....	436
11.5.7.	Распределение работы с помощью MailboxProcessor.....	437
11.5.8.	Операции кэширования в агентном программировании	439
11.5.9.	Получение результатов от MailboxProcessor	443
11.5.10.	Использование пула потоков для сообщения о событиях MailboxProcessor.....	446
11.6.	F# MailboxProcessor: 10 000 агентов для Game of Life.....	446
	Резюме	452

Глава 12. Параллельный рабочий процесс и агентное программирование с помощью TPL Dataflow 453

- 12.1. Преимущества TPL Dataflow..... 454
- 12.2. Блоки TPL Dataflow: созданы для компоновки 455
 - 12.2.1. Использование BufferBlock<TInput> в качестве буфера FIFO..... 457
 - 12.2.2. Преобразование данных с помощью TransformBlock<TInput, TOutput> 458
 - 12.2.3. Законченный пример с ActionBlock<TInput> 459
 - 12.2.4. Связывание блоков в потоке данных 461
- 12.3. Реализация сложных шаблонов «поставщик — потребитель» с помощью TDF 461
 - 12.3.1. Реализация шаблона «несколько поставщиков — один потребитель» с помощью TDF..... 461
 - 12.3.2. Шаблон «один поставщик — несколько потребителей» 463
- 12.4. Использование агентной модели в C# с помощью TPL Dataflow..... 464
 - 12.4.1. Свертка состояний и сообщений агента: Aggregate 468
 - 12.4.2. Агентное взаимодействие: параллельный счетчик слов 468
- 12.5. Параллельный рабочий процесс для сжатия и шифрования больших потоков..... 474
 - 12.5.1. Контекст: проблема обработки большого потока данных..... 474
 - 12.5.2. Сохранение порядка в потоке сообщений 480
 - 12.5.3. Связывание, распространение и завершение 481
 - 12.5.4. Правила построения рабочего процесса TDF 483
 - 12.5.5. Реактивные расширения генерации сетки (Rx) и TDF..... 484

Резюме 486

Часть III. Современные шаблоны конкурентного программирования

Глава 13. Рецепты и шаблоны для успешного конкурентного программирования..... 489

- 13.1. Освобождение памяти, занимаемой объектами, для сокращения потребления памяти..... 490
- 13.2. Нестандартный параллельный оператор Fork/Join 494
- 13.3. Распараллеливание задач с зависимостями: разработка кода для оптимизации производительности 497
- 13.4. Шлюз для координации конкурентных операций ввода-вывода с разделяемыми ресурсами: одна операция записи, несколько операций чтения 502
- 13.5. Потокобезопасный генератор случайных чисел..... 508
- 13.6. Полиморфный агрегатор событий 510

13.7. Нестандартный Rx-планировщик для управления степенью параллелизма.....	513
13.8. Конкурентное реактивное масштабируемое приложение «клиент-сервер».....	516
13.9. Многоразовый специальный высокопроизводительный параллельный оператор фильтрации-отображения	527
13.10. Неблокирующая синхронная модель передачи сообщений	532
13.11. Координирование конкурентных заданий с использованием агентной модели программирования	537
13.12. Компоновка монадических функций	542
Резюме	546
Глава 14. Построение масштабируемого мобильного приложения методом конкурентного функционального программирования	547
14.1. Практическое применение функционального программирования для серверной части приложения	548
14.2. Как разработать высокопроизводительное приложение	550
14.2.1. Ноу-хау: ACD	551
14.2.2. Другой асинхронный шаблон: постановка в очередь для отложенного выполнения	552
14.3. Правильный выбор конкурентной модели программирования	553
14.4. Торговля акциями в реальном времени: архитектура высокого уровня для фондового рынка	557
14.5. Главные элементы приложения для фондового рынка	562
14.6. Пишем код приложения для торговли на фондовом рынке	563
Резюме	586

Приложения

Приложение А. Функциональное программирование	588
Что такое функциональное программирование	588
Преимущества функционального программирования	589
Основные принципы функционального программирования	590
Противостояние программных парадигм: от императивного к объектно-ориентированному и функциональному программированию	590
Применение функций высшего порядка для увеличения абстракции	592
Применение функций высшего порядка и лямбда-выражений для создания многократно используемого кода	593
Лямбда-выражения и анонимные функции	593
Каррирование	595
Частично примененные функции	599
Преимущества частичного применения и каррирования функций в C#	601

Приложение Б. Обзор F#	603
let-привязки.....	603
Сигнатуры функций в F#	604
Создание изменяемых типов: mutable и ref.....	604
Функции как типы первого класса	605
Компоновка: операторы конвейера и компоновки.....	605
Делегаты	606
Комментарии	606
Оператор opеr.....	606
Основные типы данных.....	607
Специальное определение строки	607
Кортежи	607
Записи	608
Размеченные объединения	609
Сопоставление с образцом	610
Активные образцы	611
Коллекции	612
Массивы	612
Последовательности (seq)	613
Списки.....	613
Множества.....	614
Словари.....	614
Циклы.....	614
Классы и наследование	615
Абстрактные классы и наследование	615
Интерфейсы	616
Объектные выражения	617
Приведение типов	617
Единицы измерения.....	618
Краткая справка по API модуля событий.....	618
Приложение В. Совместимость асинхронного рабочего процесса F# и задач .NET ...	620