



Положительные отзывы к книге «Семь моделей конкуренции и параллелизма за семь недель»

Уже не одно десятилетие профессиональные программисты реализуют параллельное выполнение кода с применением потоков и блокировок. Но эта модель – лишь одна из многих, что ярко демонстрирует книга «Семь моделей конкуренции и параллелизма за семь недель». Если вы хотите добиться успеха в мире, где основные языки программирования постепенно обретают поддержку акторов, CSP¹, параллелизма данных, функционального программирования и заимствуют унифицированную модель последовательностей из языка Clojure, обязательно прочитайте эту книгу.

➤ **Стюарт Хэллоуэй (Stuart Halloway)**

Сооснователь, Cognitect

С появлением микропроцессоров, обладающих большим числом ядер, понимание конкуренции стало еще более важным, чем прежде. Вы познакомитесь с преимуществами функционального программирования с точки зрения конкуренции, узнаете, как применять акторы для разработки распределенного программного обеспечения, и исследуете приемы параллельной обработки огромных объемов информации на нескольких процессорах. Эта книга поможет вам приобрести новые навыки в разработке программ, благодаря чему вы будете готовы решать сложные задачи в ближайшие несколько лет.

➤ **Хосе Валим (José Valim)**

Сооснователь Plataformatec

В книге «Семь моделей конкуренции и параллелизма за семь недель», которая является поучительным обзором различных приемов параллельного/конкурентного выполнения, автору удалось найти баланс между описанием скучной теории и живыми экспериментами.

➤ **Фредерик Чанг (Frederick Cheung)**

Технический директор, Dressipi

Мир постоянно меняется, и каждый действующий программист должен изучать приемы конкурентного программирования. Теперь, ког-

¹ Communicating sequential processes – взаимодействующие последовательные процессы. – *Прим. перев.*

да меня будут спрашивать: «Как сделать это?», – я смогу предложить прочитать эту книгу. Я многое узнал из нее и с радостью могу порекомендовать ее.

➤ **Эндрю Хейли (Andrew Haley)**

Ведущий программист на Java, Red Hat

По мере того, как закон Адмала² начинает преобладать над законом Мура³, происходит переход от объектно-ориентированного к конкурентно-ориентированному программированию. Вследствие этого данная книга появилась как нельзя кстати. Пол проделал фантастическую работу, описав наиболее важные модели конкуренции и дав читателю обширный набор инструментов, из которого он сможет выбирать наиболее подходящие для его потребностей. Эту книгу должен прочитать каждый, кто разрабатывает программное обеспечение в нашу многоядерную эпоху.

➤ **Франческо Цезарини (Francesco Cesarini)**

Основатель и технический директор Erlang Solutions

В своей книге Пол дает превосходное введение в тернистую тему конкуренции и параллелизма, ясно и доходчиво описывая различные подходы.

➤ **Шон Эллис (Sean Ellis)**

Архитектор GPU, ARM

Просто о сложном. Я бы с удовольствием прослушал университетский курс по этой теме, взяв в качестве руководства «Семь моделей конкуренции и параллелизма за семь недель».

➤ **Карлос Сесса (Carlos Sessa)**

Разработчик для Android, Groupm

Пол Батчер (Paul Butcher) взял проблему, которая бросает в дрожь многих разработчиков, и наглядно продемонстрировал целую серию парадигм программирования, которые с успехом можно использовать для реализации конкуренции в программном обеспечении.

➤ **Пайди Крид (Páidí Creed)**

Программист, SwiftKey

Имея опыт сотрудничества с Полом, могу рекомендовать его как опытного эксперта в области архитектуры языков программирования. Эта книга является ясным описанием темы, часто недооцененной, но жизненно важной для разработки программного обеспечения.

➤ **Бен Медлок (Ben Medlock)**

Сооснователь и технический директор, SwiftKey

² https://ru.wikipedia.org/wiki/Закон_Адмала – *Прим. перев.*

³ https://ru.wikipedia.org/wiki/Закон_Мура – *Прим. перев.*



ОГЛАВЛЕНИЕ

Положительные отзывы к книге «Семь моделей конкуренции и параллелизма за семь недель»	5
Предисловие	13
Благодарности	15
Вступление	17
О книге	17
Чем не является эта книга.....	18
Примеры кода	18
Примечание для пользователей IDE.....	19
Примечание для пользователей Windows.....	19
Ресурсы в Сети.....	19
Глава 1. Введение.....	21
Конкуренция или параллелизм?	21
Похожие, но разные.....	21
За рамками последовательного программирования.....	23
Параллельная архитектура	23
Параллелизм на уровне битов	24
Параллелизм на уровне инструкций	24
Параллелизм данных	24
Параллелизм на уровне задач	25
Конкуренция: за рамками множества ядер	26
Конкурентные программы для конкурентного мира	26
Распределенные программы для распределенного мира	27
Надежные программы для непредсказуемого мира.....	27
Простые программы в сложном мире	28
Семь моделей	28
Глава 2. Потоки выполнения и блокировки	31
Самое простое из того, что может работать.....	31
День 1: взаимоисключение и модели памяти.....	32
Создание потока	33
Наша первая блокировка	34
Загадочная память	37

Видимость памяти.....	38
Несколько блокировок.....	39
Опасности сторонних методов.....	43
В завершение первого дня.....	44
День 2: помимо встроенных блокировок.....	46
Прерываемое блокирование.....	47
Тайм-ауты.....	49
Блокирование методом перебора.....	51
Условные переменные.....	54
Атомарные переменные.....	57
В завершение второго дня.....	58
День 3: на плечах гигантов.....	60
Еще раз о создании потоков.....	61
Копирование при записи.....	62
Законченная программа.....	64
В завершение третьего дня.....	74
В завершение.....	75
Сильные стороны.....	75
Слабые стороны.....	76
Другие языки.....	78
Напоследок.....	79

Глава 3. Функциональное программирование..... 80

Если какие-то действия вредят вам, перестаньте выполнять их.....	80
День 1: программирование без изменяемого состояния.....	81
Опасности изменяемого состояния.....	81
Краткий экскурс в язык Clojure.....	84
Первая функциональная программа.....	86
Параллелизм без усилий.....	87
Функциональный подсчет слов.....	89
Лень – это благо.....	93
В завершение первого дня.....	94
День 2: функциональный параллелизм.....	95
По одной странице за раз.....	95
Разделение данных на пакеты для увеличения производительности.....	98
Редуценты (reducers).....	99
Внутреннее устройство редуцентов.....	100
Разделяй и властвуй.....	103
Поддержка функции fold.....	104
Подсчет слов с помощью fold.....	105
В завершение второго дня.....	107
День 3: функциональная конкуренция.....	108
Та же структура, разный порядок вычислений.....	108

Ссылочная прозрачность.....	109
Потоки данных.....	110
Механизм future.....	111
Механизм promise.....	112
Функциональная веб-служба.....	113
В завершение третьего дня.....	121
В завершение.....	122
Сильные стороны.....	124
Слабые стороны.....	124
Другие языки.....	124
Напоследок.....	125

Глава 4. Путь Clojure – разделение идентичности и состояния 126

Лучшее из двух миров.....	126
День 1: атомы и сохраненные структуры данных.....	127
Атомы.....	127
Многопоточная веб-служба с изменяемым состоянием.....	129
Сохраненные структуры данных.....	130
Идентичность или состояние.....	134
Повторения.....	134
Валидаторы.....	135
Функции-наблюдатели.....	135
Гибридная веб-служба.....	136
В завершение первого дня.....	140
День 2: агенты и программная транзакционная память.....	141
Агенты.....	141
Журнал в памяти.....	145
Программная транзакционная память.....	146
Изменяемое разделяемое состояние.....	151
В завершение второго дня.....	151
День 3: погружение в глубину.....	152
Решение задачи о философах на основе STM.....	153
Решение задачи о философах без применения STM.....	155
Атомы или STM?.....	157
Собственная реализация конкуренции.....	158
В завершение третьего дня.....	160
В завершение.....	161
Сильные стороны.....	161
Слабые стороны.....	162
Другие языки.....	162
Напоследок.....	162

Глава 5. Акторы 164

Не объекты, а скорее ориентированные на объекты.....	164
--	-----

День 1: сообщения и почтовые ящики.....	166
Наш первый актер	166
Почтовые ящики и очереди	167
Прием сообщений	168
Связывание процессов.....	169
Актеры с сохранением состояния	170
Соккрытие сообщений за фасадом API.....	171
Двунаправленное взаимодействие.....	172
Именованые процессы	174
Отступление – функции первого порядка.....	176
Параллельная версия map().....	176
В завершение первого дня	177
День 2: обработка ошибок и отказоустойчивость.....	178
Актер кэширования	179
Определение момента отказа.....	182
Слежение за работой процессов	185
Тайм-ауты.....	186
Ядро ошибки	187
И пусть падает!.....	189
В завершение второго дня.....	190
День 3: распределенные приложения.....	191
ОТР	191
Узлы	196
Распределенный счетчик слов.....	200
В завершение третьего дня	206
В завершение.....	207
Сильные стороны	208
Слабые стороны	209
Другие языки.....	209
Напоследок	210

Глава 6. Взаимодействие последовательных процессов 211

Взаимодействия – это все.....	211
День 1: каналы и блоки go.....	213
Каналы	213
Блоки go.....	217
Операции с каналами	222
В завершение первого дня	225
День 2: множество каналов и ввод/вывод.....	227
Обслуживание множества каналов	227
Асинхронный опрос	230
Асинхронный ввод/вывод	233
В завершение второго дня.....	240
День 3: модель CSP на стороне клиента	241

Конкуренция – это образ жизни	242
Привет, ClojureScript	242
Обработка событий	245
Усмирение функций обратного вызова	247
Отправляемся в путь, чтобы увидеть Мастера	248
В завершение третьего дня	251
В завершение	251
Сильные стороны	252
Слабые стороны	252
Другие языки	253
Напоследок	253

Глава 7. Параллелизм данных 254

В недрах вашего ноутбука спрятан суперкомпьютер.....	254
День 1: программирование GPGPU.....	255
Обработка данных и параллелизм данных	255
Наша первая программа для OpenCL.....	258
Профилирование	264
Множество возвращаемых значений	265
Обработка ошибок.....	266
В завершение первого дня	268
День 2: многомерность и рабочие группы.....	270
Многомерные массивы рабочих элементов	270
Получение информации об устройстве	273
Модель платформы	275
Модель памяти	276
Параллельная свертка	277
Свертка с одной рабочей группой	277
В завершение второго дня.....	282
День 3: OpenCL и OpenGL – храните данные в GPU	283
Водная рябь	284
LWJGL.....	284
Отображение сетки в OpenGL	285
Доступ к буферу OpenGL из ядра OpenCL	287
Имитация ряби	288
В завершение третьего дня	291
В завершение	293
Сильные стороны	293
Слабые стороны	294
Другие языки	294
Напоследок	294

Глава 8. Лямбда-архитектура 295

Параллелизм позволяет обрабатывать гигантские объемы данных.....	295
---	-----

День 1: MapReduce	297
Практические аспекты	298
Основы Hadoop	299
Подсчет слов с помощью Hadoop	301
Опробование на Amazon EMR	305
Обработка XML	308
В завершение первого дня	310
День 2: пакетный уровень	313
Проблемы с традиционными системами данных	313
Вечные истины	315
Лучшие данные – исходные данные	315
Авторы правок в Википедии	317
Завершение картины	323
В завершение второго дня	325
День 3: уровень ускорения	327
Архитектура уровня ускорения	328
Подсчет правок с помощью Storm	335
В завершение третьего дня	341
В завершение	342
Сильные стороны	343
Слабые стороны	343
Альтернативы	343
Напоследок	343
Глава 9. В заключение	344
Куда мы идем?	344
Будущее за неизменяемостью	345
Будущее за распределенными вычислениями	346
Темы, оставшиеся за бортом	346
Fork/Join и захват задачи	347
Потоки данных	347
Реактивное программирование	347
Функциональное реактивное программирование	348
Grid-вычисления	348
Пространства кортежей	348
Выбор за вами	349
Библиография	350
Предметный указатель	352



ПРЕДИСЛОВИЕ

Эта книга рассказывает историю.

Кому-то может показаться странным, что книга начинается с этих слов, но для меня они наполнены смыслом. Видите ли, при создании книги «Семь моделей конкуренции и параллелизма за семь недель» мы отклонили десятки предложений от разных авторов, считавших, что можно собрать в кучу семь не связанных между собой статей и назвать это книгой. Мы думаем иначе.

В первой нашей истории («Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages» [Tat10]) рассказывалось, что объектно-ориентированные языки были хороши для своего времени, но под давлением всевозрастающей сложности программного обеспечения и необходимости управления конкуренцией в многоядерных архитектурах стали появляться языки функционального программирования, формирующие иные подходы к программированию. Пол Батчер (Paul Butcher) оказался самым полезным из научных редакторов этой книги. После четырехлетнего знакомства я понял, почему.

Пол многое повидал, находясь в первых рядах тех, кто внедрял по-настоящему масштабируемую поддержку конкуренции в действующие приложения. В книге «Seven Languages» он подметил, что решение многих сложных задач все чаще и чаще переносится на уровень языка. Пару лет спустя Пол пришел к нам с предложением написать свою книгу. Он говорил, что языки занимают важное место в общей истории, но они – лишь верхушка айсберга. Он хотел поведать нашим читателям намного более полную историю и простым языком рассказать о наиболее важных инструментах, которые используются в современном программном обеспечении для решения проблем, связанных с параллельным выполнением кода.

Сначала мы скептически отнеслись к его предложению. Такие книги пишутся тяжело – для них требуется больше времени, а процент неудач слишком высок – Пол решил сразиться со слишком большим драконом. Но, став настоящей командой, мы смогли превратить пер-

воначальный список глав в хорошую историю. По мере того как росла стопка страниц, мы все отчетливее понимали, что Пол не только имел обширные технические знания для раскрытия данной темы, но и азарт, чтобы взяться за нее. Мы также осознали всю особенность этой книги и ее своевременность. По мере углубления в нее вы поймете, о чем я говорю.

Вы будете ежиться вместе с нами, читая главу о потоках выполнения и блокировках – наиболее широко используемых ныне механизмах реализации конкуренции. Вы увидите, в каких ситуациях это решение дает плохие результаты, и затем приметесь за работу. Пол познакомит вас с самыми разными подходами, от лямбда-архитектуры, используемой в некоторых высоконагруженных социальных платформах, до модели акторов, лежащей в основе многих крупнейших в мире, высоконадежных телекоммуникационных систем. Вы увидите языки программирования, которыми пользуются профессионалы, от Java до Clojure и захватывающего языка Elixir, основанного на Erlang. И на каждом шаге этого сложного пути Пол будет поддерживать вас, делаясь своими глубокими знаниями посвященного.

Я рад представить вам книгу «Семь моделей конкуренции и параллелизма за семь недель» и надеюсь, что для вас она окажется такой же ценной, как и для меня.

Брюс Тейт (Bruce A. Tate)

Технический директор icanmakeitbetter.com

Редактор серии «Семь за семь»

Остин, Техас



БЛАГОДАРНОСТИ

Когда я объявил, что подписал контракт на создание этой книги, друг спросил меня: «Как, прошло столько времени, что ты забыл, как тебе далась первая книга?» Я был слишком наивен, полагая, что работать над второй книгой будет проще. Возможно, если бы я выбрал иной формат, отличный от формата серии «Семь за семь», я был бы не так уж и не прав.

Я едва ли смог бы написать эту книгу без мощной поддержки редактора серии Брюса Тейта и редактора-консультанта Джеки Картер (Jackie Carter). Спасибо вам обоим, что поддерживали меня, пока я вынашивал эту книгу, а также Дейву и Энди за возможность еще раз внести свой вклад в эту великолепную серию.

Многие делились со мной советами во время работы над рукописью, в том числе (без какого-то определенного порядка): Симон Харди-Френсис (Simon Hardy-Francis), Сэм Холлидей (Sam Halliday), Майк Смит (Mike Smith), Нил Эклс (Neil Eccles), Мэтью Руди Джекобс (Matthew Rudy Jacobs), Джо Осборн (Joe Osborne), Дейв Страусс (Dave Strauss), Дерек Ло (Derek Law), Фредерик Чанг (Frederick Cheung), Хьюго Тайсон (Hugo Tyson), Пол Грегори (Paul Gregory), Стефен Спенсер (Stephen Spencer), Алекс Никсон (Alex Nixon), Бен Коппин (Ben Coppin), Кит Смитерс (Kit Smithers), Эндрю Икотт (Andrew Eacott), Фриланд Эбботт (Freeland Abbott), Джеймс Алей (James Aley), Мэтью Уилсон (Matthew Wilson), Симон Добсон (Simon Dobson), Дуг Опп (Doug Orr), Джонас Бонер (Jonas Bonér), Стю Хэллоуэй (Stu Halloway), Рич Морин (Rich Morin), Дэвид Уайттакер (David Whittaker), Бо Ридберг (Bo Rydberg), Джейк Гулдинг (Jake Goulding), Арии Голд (Ari Gold), Хуан Мануэль Химено Илла (Juan Manuel Gimeno Illa), Стив Бассетт (Steve Bassett), Норберто Ортигоза (Norberto Ortigoza), Лучано Рамальо (Luciano Ramalho), Шива Джайараман (Siva Jayaraman), Шон Перри (Shaun Parry) и Джоэл Ван дер Верф (Joel Van der Werf).

Особое спасибо научным рецензентам книги (и снова без какого-то особого порядка): Карлос Сесса (Carlos Sessa), Денни Вудс (Danny

Woods), Венкат Субраманиам (Venkat Subramaniam), Симон Вуд (Simon Wood), Пайди Крид (Páidí Creed), Ян Ругли (Ian Roughley), Эндрю Томсон (Andrew Thomson), Эндрю Хейли (Andrew Haley), Шон Эллис (Sean Ellis), Джеффри Клементс (Geoffrey Clements), Лорен Сандс-Рамшау (Loren Sands-Ramshaw) и Пол Хадсон (Paul Hudson).

Наконец, я хочу принести слова благодарности и свои извинения моим друзьям, коллегам и родным. Спасибо за вашу поддержку и извините, что стремился к уединению последние восемь месяцев.



ВСТУПЛЕНИЕ

В 1989-м я приступил к работе над диссертацией, в которой занялся исследованием реализации параллельных и распределенных вычислений на разных языках программирования, – я был убежден, что со временем конкурентное программирование займет господствующее положение. Спустя два десятилетия моя правота была доказана – вокруг ведется масса разговоров о многоядерных архитектурах и о том, как использовать их преимущества.

Но поддержка конкуренции позволяет получить не только высокую производительность многоядерных систем. При правильном использовании конкуренция оказывается ключом к высокой отзывчивости, надежности, эффективности и простоте программного обеспечения.

О книге

Эта книга имеет ту же структуру, что и остальные книги серии «Семь за семь» издательства The Pragmatic Bookshelf: «Seven Languages in Seven Weeks» [Tat10], «Seven Databases in Seven Weeks» [RW12] и «Seven Web Frameworks in Seven Weeks» [MD14].¹ Для этой книги были отобраны семь подходов, чтобы на их примере продемонстрировать общее положение дел в области поддержки конкуренции. Мы охватим подходы, которые уже прочно заняли господствующее положение, которые вскоре могут занять такое положение, а также те, которые вряд ли получат достаточно широкое распространение, но являются фантастически мощными в своих нишах. Я надеюсь, что по прочтении этой книги вы будете точно знать, какой инструмент выбрать, столкнувшись с той или иной проблемой конкуренции.

Каждая глава спроектирована так, чтобы ее можно было прочитать за выходные, и разбита на три дня. Каждый день завершается упражнением, выполнение которых поможет закрепить знания, получен-

¹ В издательстве «ДМК Пресс» выпущено две книги этой же серии:
<http://dmkpress.com/catalog/computer/databases/978-5-94074-866-3/>
<http://dmkpress.com/catalog/computer/programming/978-5-94074-539-6/>

ные в этот день, а каждая глава завершается кратким перечислением сильных и слабых сторон подхода, изучавшегося в этой главе.

Несмотря на небольшие философские отступления, основное внимание в книге будет уделяться практическим примерам. Я настоятельно рекомендую поработать с этими примерами – нет ничего более убедительного, чем действующий код.

Чем не является эта книга

Эта книга не является справочным руководством. Я буду использовать языки, с которыми вы, возможно, плохо знакомы, такие как Elixir и Clojure. Но, поскольку главной темой этой книги является конкуренция, а не языки, я не буду подробно описывать те или иные аспекты языков, используемые в примерах. Надеюсь, что все и так будет понятно из контекста, но также надеюсь, что вы самостоятельно сможете заняться исследованием того или иного языка, если это потребуется. Возможно, удобнее будет читать эту книгу с веб-браузером под рукой, чтобы можно было оперативно обратиться к описанию языка в Интернете.

Эта книга не содержит инструкций по установке. Чтобы опробовать примеры кода, вам придется установить и настроить различные инструменты – какие-то подсказки вы найдете в файлах `README`, сопровождающих примеры, но в общем и целом вам придется проявить свою самостоятельность. Я использовал широко распространенные инструменты для всех примеров, поэтому вы не будете испытывать недостатка в руководствах по установке, которые легко можно найти в Интернете.

Наконец, эта книга не является исчерпывающим источником информации – в книге не так много места, чтобы охватить каждую тему во всех подробностях. Какие-то аспекты я буду упоминать лишь мимоходом, а какие-то просто пропущу. По возможности я преднамеренно использовал неидиоматический код, чтобы тем, кто знаком с используемыми языками программирования, было проще понять, как он действует. Если вы решите заняться более глубокими исследованиями какой-либо из представленных здесь технологий, обращайтесь к специализированным книгам, упоминаемым в тексте.

Примеры кода

Все примеры кода, демонстрируемые в книге, можно загрузить с сайта книги². Каждый пример включает не только исходный код, но и

² <http://pragprog.com/book/pb7con/>

систему сборки. Для каждого языка я выбрал наиболее популярную систему сборки (Maven – для Java, Leiningen – для Clojure, Mix – для Elixir, sbt – для Scala и GNU Make – для C).

В большинстве случаев эти системы не только создадут из примера выполняемую программу, но и автоматически загрузят дополнительные зависимости. А системы sbt и Leiningen загрузят даже соответствующие версии компиляторов Scala и Clojure, поэтому вам остается только установить соответствующий инструмент сборки, а описание процесса установки вы без труда найдете в Интернете.

Главным исключением является код на языке C, демонстрирующийся в главе 7 «Параллелизм данных», для опробования которого вам придется установить инструмент OpenCL для вашей операционной системы и графической карты (если только вы не пользуетесь Mac OS, среда разработки Xcode в которой уже включает все необходимое).

Примечание для пользователей IDE

Все системы сборки были проверены из командной строки. Если вы пользуетесь исключительно интегрированной средой разработки (Integrated Development Environment, IDE), вам придется импортировать систему сборки в свою IDE – большинство IDE уже поддерживают систему сборки Maven и имеют расширения для поддержки sbt и Leiningen. Но я не проверял этого, поэтому многим из вас проще будет научиться пользоваться командной строкой.

Примечание для пользователей Windows

Все примеры были протестированы в OS X и Linux. Они также должны работать и в Windows, но я не проверял их в этой операционной системе.

Исключением является код на языке C, демонстрирующийся в главе 7 «Параллелизм данных», для компиляции которого используется GNU Make и GCC. Должно быть, несложно перенести этот код на Visual C++, но, опять же, я не проверял этого.

Ресурсы в Сети

Приложения и примеры, демонстрирующиеся в этой книге, можно найти на веб-сайте Pragmatic Programmers.³ Там же вы найдете форум читателей и форму для отправки сообщений, с помощью которой вы

³ <http://pragprog.com/book/pb7con>

сможете сообщить нам о найденных ошибках и опечатках или внести предложения к будущим изданиям книги.

Пол Батчер (Paul Butcher)

Ten Tenths Consulting

paul@tententhsconsulting.com

Кембридж, Великобритания, июнь 2014



ГЛАВА 1.

Введение

Конкурентное программирование не является чем-то новым, но оно лишь недавно превратилось в горячую тему. Языки программирования, такие как Erlang, Haskell, Go, Scala и Clojure, получили известность отчасти благодаря превосходной поддержке конкуренции.

Основной причиной всплеска интереса стал так называемый «кризис многоядерных систем». В соответствии с законом Мура число транзисторов на одном кристалле постоянно увеличивается¹, но это приводит не к повышению быстродействия процессора, а к увеличению числа ядер.

Как заметил Герб Саттер (Herb Sutter): «Бесплатного супа больше не будет».² Вы больше не сможете получить более высокую скорость выполнения вашего кода, просто подождяв, пока появится более быстрое «железо». В наши дни, чтобы добиться более высокой производительности, требуется использовать несколько ядер, а значит, и приемы параллельного программирования.

Конкуренция или параллелизм?

Эта книга рассказывает о конкуренции, тогда почему мы говорим о параллелизме? Хотя эти два термина часто используются взаимозаменяемо, конкуренция и параллелизм суть разные вещи.

Похожие, но разные

Конкурентная программа всегда имеет несколько потоков выполнения. Потоки могут выполняться параллельно, но это не обязательно.

¹ https://ru.wikipedia.org/wiki/Закон_Мура

² <http://www.gotw.ca/publications/concurrency-ddj.htm> (Перевод на русский язык: <http://habrahabr.ru/post/145432/> – Прим. перев.)

Параллельная программа обычно действует намного быстрее, чем последовательная, решая сразу (параллельно) несколько задач. При этом она необязательно имеет несколько потоков выполнения.

Конкуренцию можно считать аспектом предметной области, например когда программе требуется обрабатывать одновременно (или почти одновременно) возникающие события. Параллелизм, напротив, является аспектом решения, например с целью увеличить скорость работы за счет параллельной обработки разных блоков данных.

Вот как обозначает различия Роб Пайк (Rob Pike),³

- Конкуренция – это способ одновременного решения множества разных задач.
- Параллелизм – это способ одновременного выполнения разных частей одной задачи.

Так о чем эта книга, о конкуренции или о параллелизме?



Джо спрашивает: в чем разница между конкуренцией и параллелизмом?

Моя жена – учитель. Как и большинство учителей, она может заниматься сразу несколькими делами. В каждый конкретный момент она может делать только что-то одно, но способна делать несколько дел конкурентно. Слушая, как читает кто-то из ее учеников, она может прервать его, чтобы успокоить расшумевшийся класс или ответить на чей-то вопрос. Это – конкурентная работа, а не параллельная (она может делать только что-то одно).

Если ей в помощь дать помощника (один из них будет слушать ученика, читающего текст, а другой – отвечать на вопросы), мы получим вариант работы в паре, который можно назвать конкурентным и параллельным.

Представьте, что класс придумал поздравительные открытки и решает заняться их массовым выпуском. Решить эту проблему можно, поручив каждому ребенку нарисовать пять открыток. Это параллельная организация работы (если отбросить несущественные мелочи), но не конкурентная – выполняется только одна задача.

³ <http://concur.rspace.googlecode.com/hg/talk/concur.html>

За рамками последовательного программирования

Параллелизм и конкуренцию объединяет то, что обе эти парадигмы выходят за рамки модели последовательного программирования, в которой все процессы протекают последовательно, друг за другом. В этой книге мы будем рассматривать обе парадигмы, параллельного и конкурентного программирования (если бы я был педантом, я дал бы этой книге название «Семь моделей конкурентного и/или параллельного программирования за семь недель», но такое название едва ли уместилось бы по ширине обложки).

Конкуренцию и параллелизм часто путают между собой, потому что традиционные потоки выполнения и блокировки не имеют прямого отношения к параллелизму. Если вы пожелаете задействовать несколько ядер с помощью потоков и блокировок, вы вынуждены будете создать конкурентную программу и запускать ее на параллельной архитектуре.

Такое ограничение весьма досадно, потому что программы, реализующие конкурентную модель, часто недетерминированы – они дают разные результаты в зависимости от точности синхронизации событий. При решении по-настоящему конкурентной задачи недетерминированность является естественным и вполне ожидаемым следствием. Параллелизм, напротив, необязательно подразумевает недетерминированность – операция удвоения всех чисел в массиве не приведет (по крайней мере, не должна) к недетерминированности только потому, что удвоение половины чисел выполняется одним ядром, а другая половина – другим. Языки с явной поддержкой параллелизма позволяют писать параллельный код, не внося недетерминированности.

Параллельная архитектура

Большинство из нас привыкло связывать параллелизм с наличием множества ядер, однако в действительности современные компьютеры обеспечивают параллелизм на многих других уровнях. Причина, до недавнего времени объяснявшая увеличение скорости работы ядер с каждым годом, заключается в параллельном использовании всех тех дополнительных транзисторов, предсказанных законом Мура, – как на уровне отдельных битов, так и на уровне инструкций.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru