

УДК 004.432.42 Haskell
ББК 32.973.28-018.1
M28

M28 Саймон Марлоу
 Параллельное и конкурентное программирование на языке Haskell / Пер. с англ. В. Н. Брагилевского. – М.: ДМК Пресс, 2017. — 372 с.: ил.

ISBN 978-5-97060-560-8

Если вы уже владеете программированием на языке Haskell, эта книга научит вас использованию множества интерфейсов и библиотек, предназначенных для написания параллельных и конкурентных программ. Вы узнаете, как распараллеливание на многоядерные процессоры позволяет ускорять вычислительно нагруженные программы и как конкурентность облегчает написание программ с активно взаимодействующими между собой и с другими программами потоками.

Автор Саймон Марлоу проведёт вас по этому пути, сопровождая его большим количеством примеров, с которыми можно самостоятельно экспериментировать, запуская, изменяя и расширяя. Книга делится на две части, посвящённые таким инструментам, как Parallel Haskell и Concurrent Haskell, включённые в неё упражнения позволят вам научиться:

- выражать параллелизм в языке Haskell средствами монады Eval и стратегий вычислений;
- распараллеливать обычный код на языке Haskell в монаде Par;
- организовывать параллельные вычисления с массивами на основе библиотеки Repa;
- использовать библиотеку Accelerate для запуска вычислений на графических процессорах;
- работать с базовыми интерфейсами для написания конкурентного кода;
- реализовывать высокопроизводительные конкурентные сетевые серверы;
- писать распределённые программы, запускающиеся на множестве машин сети.

УДК 004.432.42 Haskell
 ББ 32.973.28-018.1

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

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок всё равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несёт ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-449-33594-6 (англ.)
 ISBN 978-5-97060-560-8 (рус.)

© 2013 Simon Marlow
 © Перевод на русский язык, оформление,
 ДМК Пресс

Оглавление

Предисловие	10
1. Введение	16
Терминология: параллелизм и конкурентность	17
Инструменты и документация	19
Примеры программ	20
Часть I. Parallel Haskell	21
2. Простейший параллелизм: монада Eval	26
Ленивые вычисления и слабая головная нормальная форма	26
Монада Eval, <code>par</code> и <code>rseq</code>	33
Пример: распараллеливание решателя sudoku	37
Модуль <code>DeepSeq</code>	49
3. Стратегии вычислений	51
Параметризованные стратегии	53
Стратегия для параллельного вычисления списка	55
Пример: задача k-средних	56
Распараллеливание k-средних	62
Производительность и анализ	63
Визуализация активности нитей	68
Размеры заданий	68
Сборка мусора для нитей и спекулятивный параллелизм	71
Распараллеливание ленивых потоков посредством <code>parBuffer</code>	74
Стратегии разбиения на фрагменты	79
Свойство тождественности	79
4. Параллелизм по данным: монада <code>Par</code>	81
Пример: кратчайшие пути на графе	86
Конвейерный параллелизм	91
Ограничение скорости производителя	96
Ограничения конвейерного параллелизма	96

Пример: расписание конференции	97
Распараллеливание	103
Пример: параллельный вывод типов	107
Использование разных планировщиков	113
Сравнение монады <code>Par</code> и стратегий вычислений	113
5. Параллельное программирование с библиотекой <code>Pera</code>	115
Массивы, формы и индексы	116
Операции над массивами	119
Пример: вычисление кратчайших путей	122
Распараллеливание	125
Свёртки и полиморфизм форм	128
Пример: поворот изображения	130
Резюме	135
6. Программирование GPU с библиотекой <code>Accelerate</code>	136
Обзор	137
Массивы и индексы	138
Запуск простого <code>Accelerate</code> -вычисления	140
Скаляры как массивы	142
Индексация массивов	143
Создание массивов внутри <code>Ass</code>	143
Склеивание массивов	145
Константы	146
Пример: кратчайшие пути	146
Запуск на GPU	150
Отладка <code>CUDA</code>	151
Пример: генератор множества Мандельброта	152
Часть II. <code>Concurrent Haskell</code>	159
7. Простейшая конкурентность: потоки и изменяемые переменные	162
Простой пример: напоминания	163
Передача данных: переменные <code>MVar</code>	166
<code>MVar</code> как простой канал: служба журнализации	168
<code>MVar</code> как контейнер для разделяемого состояния	172
<code>MVar</code> как строительный блок: неограниченные каналы	175
Справедливость	180
8. Ввод-вывод в фоновом режиме	182
Исключения в <code>Haskell</code>	185
Обработка ошибок в <code>Async</code>	191
Слияние	193

9.	Аннулирование и тайм-ауты	197
	Асинхронные исключения	198
	Маскирование асинхронных исключений	201
	Функция <code>bracket</code>	205
	Безопасность по асинхронным исключениям для каналов	206
	Тайм-ауты	208
	Перехват асинхронных исключений	211
	Функция <code>mask</code> и вызов <code>forkIO</code>	213
	Асинхронные исключения: обсуждение	215
10.	Программная транзакционная память	217
	Основной пример: управление окнами	218
	Блокирование	222
	Блокирование до момента изменения условия	225
	Слияние средствами <code>STM</code>	227
	Возвращение к <code>Async</code>	228
	Реализация каналов средствами <code>STM</code>	230
	Возможность других операций	232
	Комбинирование блокируемых операций	232
	Безопасность асинхронных исключений	233
	Альтернативная реализация каналов	234
	Ограниченные каналы	237
	Чего нельзя делать с <code>STM</code> ?	239
	Производительность	241
	Итоги	243
11.	Высокоуровневые конкурентные абстракции	245
	Избежание утечки потоков	245
	Комбинаторы симметричной конкурентности	247
	Тайм-ауты с помощью <code>gase</code>	250
	Добавляем экземпляр класса <code>Functor</code>	251
	Итоги: интерфейс <code>Async</code>	253
12.	Конкурентные сетевые серверы	254
	Простейший сервер	254
	Расширяем простой сервер состоянием	259
	Первое решение: одна глобальная блокировка	259
	Второе решение: один канал на серверный поток	260
	Третье решение: широковещательный канал	261
	Четвёртое решение: использование <code>STM</code>	262
	Реализация	263
	Чат-сервер	267
	Архитектура	268

Данные клиента	269
Данные сервера	271
Сервер	271
Добавление нового клиента	272
Запуск клиента	274
Резюме	276
13. Параллельное программирование на потоках	278
Как распараллелить программу посредством конкурентности	279
Пример: поиск файлов	279
Последовательная версия	280
Параллельная версия	282
Производительность и масштабируемость	284
Ограничение количества потоков с помощью семафора	285
Монада ParIO	292
14. Распределённое программирование	295
Семейство пакетов distributed-process	296
Распределённая конкурентность или параллелизм?	298
Первый пример: пинг-понг	299
Процессы и монада Process	299
Определение типа сообщения	300
Серверный процесс	301
Ведущий процесс	303
Функция main	304
Итоги примера	305
Пинг-понг на нескольких узлах	306
Запуск нескольких узлов на одной машине	307
Запуск на нескольких машинах	308
Типизированные каналы	309
Слияние каналов	313
Обработка отказов	315
Философия распределённых отказов	318
Распределённый чат-сервер	319
Типы данных	320
Отправка сообщений	323
Широковещание	324
Распределение	325
Тестирование сервера	328
Отказы и добавление/удаление узлов	328
Упражнение: распределённое хранилище пар «ключ–значение»	330

15. Отладка, настройка и вызов внешнего кода	334
Отладка конкурентных программ	334
Проверка статуса потока	334
Запись событий в журнал и ThreadScope	335
Обнаружение тупиков	338
Настройка конкурентных (и параллельных) программ	341
Создание потоков и операции с MVar	342
Разделяемые конкурентные структуры данных	344
Настройка системы времени исполнения	346
Конкурентность и интерфейс внешних функций	348
Потоки и исходящие внешние вызовы	348
Асинхронные исключения и внешние вызовы	351
Потоки и входящие внешние вызовы	351
Предметный указатель	353