# Оглавление


Часть I ▪ Теория, разбавленная превосходными примерами ..... 35
*1* ▪ Так что же такое Spark? ..... 36
*2* ▪ Архитектура и рабочий процесс ..... 56
*3* ▪ Важнейшая роль фрейма данных ..... 72
*4* ▪ Природная лень ..... 112
*5* ▪ Создание простого приложения для развертывания ..... 138
*6* ▪ Развертывание простого приложения ..... 165

Часть II ▪ Потребление данных ..... 190
*7* ▪ Потребление данных из файлов ..... 192
*8* ▪ Потребление из баз данных ..... 226
*9* ▪ Более сложный процесс потребления: поиск источников данных и создание собственных ..... 255
*10* ▪ Потребление через структурированные потоки ..... 288

Часть III ▪ Преобразлвание данных ..... 313
*11* ▪ Работа с языком SQL ..... 314
*12* ▪ Преобразование данных ..... 329
*13* ▪ Преобразование документов в целом ..... 364
*14* ▪ Расширенные преобразования с помощью функций, определенных пользователем ..... 378
*15* ▪ Агрегирование данных ..... 396

Часть IV ▪ Продолжаем изучение Spark ..... 424
*16* ▪ Кеширование и копирование данных в контрольных точках: улучшение производительности Spark ..... 426
*17* ▪ Экспорт данных и создание полноценных конвейеров обработки данных ..... 455
*18* ▪ Описание ограничений процесса развертывания: объяснение экосистемы ..... 478

# Содержание

Оглавление ..... 5
Словарь терминов ..... 15
Вступительное слово ..... 17
Предисловие ..... 19
Благодарности ..... 21
О чем эта книга ..... 24
Об авторе ..... 32
Иллюстрация на обложке ..... 33

**Часть I Теория, разбавленная превосходными примерами** ..... 35

*1 Так что же такое Spark?* ..... 36
1.1 Общая картина: что такое Spark и что он делает ..... 37
*1.1.1 Что такое Spark* ..... 37
*1.1.2 Четыре столпа маны* ..... 40
1.2 Как можно использовать Spark ..... 41
*1.2.1 Spark в процессе обработки данных / инженерии данных* ..... 41
*1.2.2 Spark в научных исследованиях в области обработки данных* ..... 44
1.3 Что можно делать с помощью Spark ..... 45
*1.3.1 Spark прогнозирует качество пунктов питания Северной Каролины* ..... 46
*1.3.2 Spark обеспечивает быструю передачу данных для Lumeris* ..... 47
*1.3.3 Spark анализирует журналы наблюдения за оборудованием CERN* ..... 48
*1.3.4 Другие варианты использования* ..... 48

1.4 Почему вам очень понравится фрейм данных ..... 48
*1.4.1 Фрейм данных с точки зрения Java* ..... 49
*1.4.2 Фрейм данных с точки зрения СУРБД* ..... 49
*1.4.3 Графическое представление фрейма данных* ..... 50
1.5 Первый пример ..... 51
*1.5.1 Рекомендуемое программное обеспечение* ..... 51
*1.5.2 Скачивание исходного кода* ..... 52
*1.5.3 Запуск первого приложения* ..... 52
*1.5.4 Первый исходный код для вас* ..... 53
Резюме ..... 54

**2 *Архитектура и рабочий процесс*** ..... 56
2.1 Создание собственной мысленной (когнитивной) модели ..... 57
2.2 Использование кода Java для создания мысленной (когнитивной) модели ..... 58
2.3 Подробный разбор приложения ..... 61
*2.3.1 Установление соединения с ведущим узлом* ..... 62
*2.3.2 Загрузка или потребление содержимого CSV-файла* ..... 63
*2.3.3 Преобразование данных* ..... 66
*2.3.4 Сохранение работы, сделанной в фрейме данных, в базе данных* ..... 68
Резюме ..... 71

**3 *Важнейшая роль фрейма данных*** ..... 72
3.1 Чрезвычайно важная роль фрейма данных в Spark ..... 73
*3.1.1 Внутренняя организация фрейма данных* ..... 74
*3.1.2 Неизменяемость – это не клятва* ..... 75
3.2 Использование фреймов данных на примерах ..... 77
*3.2.1 Фрейм данных после простой операции потребления CSV-файла* ..... 79
*3.2.2 Данные хранятся в разделах* ..... 84
*3.2.3 Подробнее о схеме* ..... 86
*3.2.4 Фрейм данных после потребления формата JSON* ..... 87
*3.2.5 Объединение двух фреймов данных* ..... 94
3.3 Фрейм данных как структура `Dataset<Row>` ..... 99
*3.3.1 Повторное использование простых старых объектов Java (POJO)* ..... 100
*3.3.2 Создание набора данных из строк* ..... 101
*3.3.3 Преобразование фрейма данных в набор данных и обратно* ..... 103
3.4 Предшественник фрейма данных: RDD ..... 109
Резюме ..... 110

4 *Природная лень* ..... 112
4.1 Пример рациональной лени из реальной жизни ..... 113
4.2 Пример рациональной лени в Spark ..... 114
*4.2.1 Рассмотрение результатов преобразований и действий* ..... 115
*4.2.2 Процесс преобразования шаг за шагом* ..... 116
*4.2.3 Код реализации процесса преобразования/действия* ..... 119
*4.2.4 Загадка создания 7 миллионов точек данных за 182 мс* ..... 123
*4.2.5 Загадка, связанная с измерением времени для действий* ..... 125
4.3 Сравнение с СУРБД и обычными приложениями ..... 130
*4.3.1 Обработка набора данных с коэффициентами рождаемости для подростков* ..... 130
*4.3.2 Анализ различий между обычным приложением и приложением Spark* ..... 131
4.4 Spark великолепно подходит для приложений, ориентированных на обработку данных ..... 133
4.5 Catalyst – катализатор приложения ..... 133
Резюме ..... 137

5 *Создание простого приложения для развертывания* ..... 138
5.1 Пример без операции потребления данных ..... 139
*5.1.1 Вычисление π* ..... 139
*5.1.2 Исходный код для вычисления приближенного значения π* ..... 142
*5.1.3 Что такое лямбда-функции в Java* ..... 148
*5.1.4 Приближенное вычисление π с использованием лямбда-функций* ..... 150
5.2 Взаимодействие со Spark ..... 152
*5.2.1 Локальный режим* ..... 153
*5.2.2 Режим кластера* ..... 154
*5.2.3 Интерактивный режим Scala и Python* ..... 158
Резюме ..... 163

6 *Развертывание простого приложения* ..... 165
6.1 Подготовка к изучению примера: роль компонент ..... 168
*6.1.1 Краткий обзор компонент и взаимодействий между ними* ..... 168
*6.1.2 Рекомендации по устранению проблем в архитектуре Spark* ..... 172
*6.1.3 Дополнительная информация для изучения* ..... 173
6.2 Создание кластера ..... 174
*6.2.1 Создание собственного кластера* ..... 174
*6.2.2 Настройка среды кластера* ..... 176

6.3 Создание приложения для работы в кластере ... 179
*6.3.1 Создание файла uberJAR для приложения* ... 180
*6.3.2 Создание приложения с использованием Git и Maven* ... 182
6.4 Выполнение приложения в кластере ... 185
*6.4.1 Передача файла uberJAR* ... 185
*6.4.2 Выполнение приложения* ... 186
*6.4.3 Анализ пользовательского интерфейса Spark* ... 187
Резюме ... 188

**Часть II Потребление данных** ... 190

7 ***Потребление данных из файлов*** ... 192
7.1 Общее поведение парсеров ... 194
7.2 Сложная процедура потребления данных из CSV-файла ... 194
*7.2.1 Требуемый вывод результата* ... 196
*7.2.2 Код* ... 197
7.3 Потребление CSV-данных с известной схемой ... 198
*7.3.1 Требуемый вывод результата* ... 199
*7.3.2 Код* ... 200
7.4 Потребление данных из JSON-файла ... 201
*7.4.1 Требуемый вывод результата* ... 203
*7.4.2 Код* ... 204
7.5 Потребление данных из многострочного JSON-файла ... 205
*7.5.1 Требуемый вывод результата* ... 207
*7.5.2 Код* ... 207
7.6 Потребление данных из файла XML ... 208
*7.6.1 Требуемый вывод результата* ... 210
*7.6.2 Код* ... 211
7.7 Потребление данных из текстового файла ... 213
*7.7.1 Требуемый вывод результата* ... 214
*7.7.2 Код* ... 214
7.8 Форматы файлов для больших данных ... 215
*7.8.1 Проблема с обычными форматами файлов* ... 215
*7.8.2 Avro – формат сериализации на основе схемы* ... 217
*7.8.3 ORC – формат хранения данных в столбцах* ... 217
*7.8.4 Parquet – еще один формат хранения данных в столбцах* ... 218
*7.8.5 Сравнение форматов Avro, ORC и Parquet* ... 218
7.9 Потребление данных из файлов Avro, ORC и Parquet ... 218
*7.9.1 Потребление данных в формате Avro* ... 219
*7.9.2 Потребление данных в формате ORC* ... 221
*7.9.3 Потребление данных в формате Parquet* ... 222
*7.9.4 Справочная информация по организации потребления данных в форматах Avro, ORC, Parquet* ... 224
Резюме ... 224

8 *Потребление из баз данных* ..... 226
8.1 Потребление из реляционных баз данных ..... 228
*8.1.1 Контрольный перечень операций при установлении соединения с базой данных* ..... 228
*8.1.2 Объяснение происхождения данных, используемых в следующих примерах* ..... 229
*8.1.3 Требуемый вывод результата* ..... 231
*8.1.4 Код* ..... 232
*8.1.5 Другая версия кода* ..... 234
8.2 Роль диалекта ..... 236
*8.2.1 Что такое диалект* ..... 236
*8.2.2 Диалекты JDBC, предоставляемые в Spark* ..... 237
*8.2.3 Создание собственного диалекта* ..... 237
8.3 Расширенные запросы и процесс потребления ..... 240
*8.3.1 Фильтрация с использованием ключевого слова WHERE* ..... 240
*8.3.2 Соединение данных в базе данных* ..... 243
*8.3.3 Выполнение потребления и распределение данных* ..... 246
*8.3.4 Итоги изучения расширенных функциональных возможностей* ..... 249
8.4 Потребление данных из Elasticsearch ..... 249
*8.4.1 Поток данных* ..... 249
*8.4.2 Набор данных о ресторанах Нью-Йорка, извлекаемый Spark* ..... 250
*8.4.3 Исходный код для потребления набора данных о ресторанах из Elasticsearch* ..... 252
Резюме ..... 253

9 *Более сложный процесс потребления: поиск источников данных и создание собственных* ..... 255
9.1 Что такое источник данных ..... 257
9.2 Преимущества прямого соединения с источником данных ..... 259
*9.2.1 Временные файлы* ..... 260
*9.2.2 Скрипты для улучшения качества данных* ..... 260
*9.2.3 Данные по запросу* ..... 261
9.3 Поиск источников данных на сайте Spark Packages ..... 261
9.4 Создание собственного источника данных ..... 261
*9.4.1 Обзор примера проекта* ..... 262
*9.4.2 Интерфейс API специализированного источника данных и его параметры* ..... 264
9.5 Что происходит внутри: создание самого источника данных ..... 267
9.6 Использование файла регистрации и заявочного класса ..... 268

9.7 Объяснение взаимоотношения между данными и схемой ... 270
*9.7.1 Источник данных создает отношение* ... 271
*9.7.2 Внутри отношения* ... 274
9.8 Создание схемы из JavaBean ... 277
9.9 Создание фрейма данных – манипуляции с утилитами ... 280
9.10 Другие классы ... 286
Резюме ... 286

10 ***Потребление через структурированные потоки*** ... 288
10.1 Что такое потоковая обработка ... 290
10.2 Создание первого потока данных ... 292
*10.2.1 Генерация потока данных* ... 293
*10.2.2 Потребление записей* ... 296
*10.2.3 Считывание записей, а не строк* ... 302
10.3 Потребление данных из сетевых потоков ... 303
10.4 Работа с несколькими потоками ... 306
10.5 Различия между дискретизированными и структурированными потоками ... 311
Резюме ... 312

**Часть III Преобразлвание данных** ... 313

11 ***Работа с языком SQL*** ... 314
11.1 Работа со Spark SQL ... 315
11.2 Различия между локальными и глобальными представлениями ... 319
11.3 Совместное использование API фрейма данных и Spark SQL ... 321
11.4 Не удаляйте (DELETE) данные ... 324
11.5 Рекомендации для дальнейшего изучения SQL ... 327
Резюме ... 327

12 ***Преобразование данных*** ... 329
12.1 Что такое преобразование данных ... 330
12.2 Процесс и пример преобразования данных на уровне записи ... 331
*12.2.1 Обследование данных для оценки их сложности* ... 333
*12.2.2 Отображение данных для создания схемы процесса* ... 335
*12.2.3 Написание исходного кода преобразования* ... 338
*12.2.4 Итоговый обзор результата преобразования данных для обеспечения качества обработки* ... 345
*12.2.5 Несколько слов о сортировке* ... 347

*12.2.6 Завершение первого процесса преобразования с использованием Spark* ..... 347
12.3 Соединение наборов данных ..... 348
*12.3.1 Более подробно о соединяемых наборах данных* ..... 348
*12.3.2 Создание списка вузов по округам* ..... 350
*12.3.3 Выполнение соединений* ..... 356
12.4 Выполнение других преобразований ..... 362
Резюме ..... 362

**13 *Преобразование документов в целом*** ..... 364
13.1 Преобразование документов в целом и их структура ..... 365
*13.1.1 Упрощение структуры документа в формате JSON* ..... 365
*13.1.2 Создание документов с вложенной структурой для передачи и сохранения* ..... 371
13.2 Секреты статических функций ..... 376
13.3 Выполнение других преобразований ..... 377
Резюме ..... 377

**14 *Расширенные преобразования с помощью функций, определенных пользователем*** ..... 378
14.1 Расширение функциональности Apache Spark ..... 379
14.2 Регистрация и вызов UDF ..... 381
*14.2.1 Регистрация UDF в Spark* ..... 384
*14.2.2 Использование UDF совместно с API фрейма данных* ..... 385
*14.2.3 Использование UDF совместно с SQL* ..... 387
*14.2.4 Реализация UDF* ..... 388
*14.2.5 Написание кода сервиса* ..... 390
14.3 Использование UDF для обеспечения высокого уровня качества данных ..... 392
14.4 Ограничения использования UDF ..... 394
Резюме ..... 395

**15 *Агрегирование данных*** ..... 396
15.1 Агрегирование данных в Spark ..... 397
*15.1.1 Краткое описание агрегаций* ..... 397
*15.1.2 Выполнение простых агрегаций с использованием Spark* ..... 400
15.2 Выполнение агрегаций с оперативными данными ..... 403
*15.2.1 Подготовка набора данных* ..... 403
*15.2.2 Агрегация данных для получения более точной информации о школах* ..... 408
15.3 Создание специализированных агрегаций с использованием UDAF ..... 415
Резюме ..... 422

**Часть IV Продолжаем изучение Spark** ..... 424

***16 Кеширование и копирование данных в контрольных точках: улучшение производительности Spark*** ..... 426

16.1 Кеширование и копирование данных в контрольных точках могут повысить производительность ..... 427
*16.1.1 Полезность кеширования в Spark* ..... 429
*16.1.2 Изысканная эффективность механизма копирования данных в контрольных точках в Spark* ..... 431
*16.1.3 Использование кеширования и копирования данных в контрольных точках* ..... 431
16.2 Кеширование на практике ..... 442
16.3 Дополнительные материалы по оптимизации производительности ..... 452
Резюме ..... 453

***17 Экспорт данных и создание полноценных конвейеров обработки данных*** ..... 455

17.1 Экспорт данных ..... 456
*17.1.1 Создание конвейера с наборами данных NASA* ..... 456
*17.1.2. Преобразование столбцов в метки времени datetime* ..... 459
*17.1.3 Преобразование процентов степени достоверности в уровень достоверности* ..... 461
*17.1.4 Экспорт данных* ..... 462
*17.1.5 Экспорт данных: что происходит в действительности* ..... 465
17.2 Delta Lake: удобная база данных прямо в системе ..... 466
*17.2.1 Объяснение, почему необходима база данных* ..... 467
*17.2.2 Использование Delta Lake в конвейере обработки данных* ..... 468
*17.2.3 Потребление данных из Delta Lake* ..... 473
17.3 Доступ к сервисам облачного хранилища из Spark ..... 475
Резюме ..... 477

***18 Описание ограничений процесса развертывания: объяснение экосистемы*** ..... 478

18.1 Управление ресурсами с использованием YARN, Mesos и Kubernetes ..... 479
*18.1.1 Встроенный автономный режим управления ресурсами* ..... 480
*18.1.2 YARN управляет ресурсами в среде Hadoop* ..... 481
*18.1.3 Mesos – автономный диспетчер ресурсов* ..... 483
*18.1.4 Kubernetes управляет оркестровкой контейнеров* ..... 484
*18.1.5 Правильный выбор диспетчера ресурсов* ..... 486

18.2 Совместное использование файлов с помощью Spark ..... 486
*18.2.1 Доступ к данным, содержащимся в файлах* ..... 487
*18.2.2 Совместное использование файлов с помощью распределенных файловых систем* ..... 488
*18.2.3 Доступ к файлам на совместно используемых накопителях или на файловом сервере* ..... 490
*18.2.4 Работа с сервисами совместного использования файлов для распределения файлов* ..... 491
*18.2.5 Другие варианты обеспечения доступа к файлам в Spark* ..... 492
*18.2.6 Гибридное решение совместного использования файлов в Spark* ..... 492
18.3 Уверенность в безопасности приложения Spark ..... 492
*18.3.1 Безопасность сетевых компонентов инфраструктуры* ..... 493
*18.3.2 Безопасность при использовании диска Spark* ..... 494
Резюме ..... 495

*Приложение А* Установка Eclipse ..... 496
*Приложение Б* Установка Maven ..... 502
*Приложение В* Установка Git ..... 506
*Приложение Г* Загрузка исходного кода и начало работы в Eclipse ..... 508
*Приложение Д* Хронология корпоративных данных ..... 514
*Приложение Е* Справочная информация по реляционным базам данных ..... 519
*Приложение Ж* Статические функции упрощают преобразования ..... 524
*Приложение З* Краткий справочник по Maven ..... 533
*Приложение И* Справочник по преобразованиям и действиям ..... 538
*Приложение Й* Немного Scala ..... 548
*Приложение К* Установка Spark в реальной эксплуатационной среде и несколько рекомендаций ..... 550
*Приложение Л* Справочник по операциям потребления ..... 563
*Приложение М* Справочник по соединениям ..... 574
*Приложение Н* Установка Elasticsearch и пример набора данных ..... 586
*Приложение О* Генерация потоковых данных ..... 592
*Приложение П* Справочник по обработке потоковых данных ..... 597
*Приложение Р* Справочник по экспорту данных ..... 608
*Приложение С* Где искать помощь при затруднениях ..... 616

Предметный указатель ..... 621

# *Словарь терминов*

Краткое описание терминов Spark, используемых в процессе развертывания.

| Термин на английском языке | Термин на русском языке | Определение |
|---|---|---|
| Application | Приложение | Программа, которая создается в рабочей среде Spark и для функционирования в Spark.Состоит из программы-драйвера и исполнителей в кластере |
| Application JAR | JAR-приложение | Архивный файл Java (JAR), содержащий приложение Spark. Это может быть файл uber-JAR, включающий все зависимости |
| Cluster manager | Диспетчер кластера | Внешний сервис для распределения ресурсов в кластере. Возможно использование встроенного в Spark диспетчера кластера |
| Deploy mode | Режим развертывания | Определяет, где запускается и работает процесс драйвера. В режиме кластера (cluster mode) фреймворк запускает драйвер внутри кластера. В режиме клиента (client mode) ведомый (подчиненный) запускает драйвер за пределами кластера. Можно определить, в каком режиме вы находитесь, выполнив метод `deployMode()`. Метод возвращает свойство, защищенное от изменения (только для чтения) |
| Driver program | Программа-драйвер | Процесс, выполняющий основную функцию `main()` приложения и создающий контекст `SparkContext`. Все начинается именно здесь |
| Executor | Исполнитель | Процесс, запускаемый для приложения на рабочем узле. Исполнитель выполняет задачи и сохраняет промежуточные данные в памяти или на диске. У каждого приложения имеются собственные исполнители. |
| Job | Задание | Параллельное вычисление (выполнение), состоящее из нескольких задач, которые порождаются в ответ на некоторое действие Spark (например, `save()` или `collect()`, подробнее см. приложение И) |
| Stage | Этап | Каждое задание разделяется на более мелкие наборы задач, называемые этапами, зависящими друг от друга (подобно этапам отображения и сокращения в MapReduce) |
| Task | Задача | Минимальная единица работы, которая передается одному исполнителю |
| Worker node | Рабочий узел | Любой узел, который может выполнять код приложения в кластере |


APACHE Spark
элементы
Процессы и ресурсы приложения
Задание: параллельные задачи, запускаемые после вызова какого-либо действия
Архивный файл JAR приложения
Программа-драйвер
Сеанс SparkSession (контекст SparkContext)
Диспетчер кластера
Задания разделяются на этапы
Доступ к программе-драйверу возможен в соответствующем режиме развертывания
Рабочий узел
Исполнитель
Кеш
За-дача
За-дача
Рабочий узел
Исполнитель
Кеш
За-дача
За-дача
Код конкретного приложения в пакете JAR
Узлы

# Вступительное слово

## Аналитическая операционная система

В XX веке масштабы деятельности в деловой сфере значительно укрупнились благодаря глобальному расширению и распространению. Любая компания, выполняющая бизнес-операции по всему миру, получила вытекающее из этого преимущество в стоимости и степени распространения, что привело к появлению более конкурентоспособной продукции. Предприятия розничной торговли с глобальной сетью магазинов получили преимущество в распространении товаров, несравнимое с деятельностью более мелких компаний. Результаты этого глобального укрупнения определили преимущества в конкуренции на несколько десятилетий вперед.

Всю ситуацию в целом изменил интернет. В настоящее время существует три главных результата глобального укрупнения:

- сеть – замкнутость, управляемая надежной сетью (Facebook, Twitter, Etsy и т.д.);
- экономика глобального укрупнения – более низкая цена единицы товара, управляемая объемом (Apple, TSMC и т.д.);
- данные – всеобъемлющее машинное обучение и интуиция на основе динамически изменяющихся совокупностей данных.

В книге Big Data Revolution (Wiley, 2015 г.) я исследовал несколько компаний, которые извлекали реальную выгоду из данных в результате глобального укрупнения бизнес-деятельности. Но сейчас, в 2019 году, большие данные все еще остаются в основном неисследованной областью в организациях по всему миру. Spark – аналитическая операционная система – представляет собой средство, способное изменить существующее положение.

Spark стал инструментом, изменившим инновационную политику компании IBM. Spark – это аналитическая операционная система, обеспечивающая стандартизацию и унификацию источников данных и средства доступа к данным. Стандартизированная программная мо-

дель Spark делает эту систему самым лучшим вариантом выбора для разработчиков, создающих приложения анализа огромных объемов данных. Spark сокращает время и снижает сложность создания аналитических рабочих потоков, позволяя разработчикам сосредоточиться на задачах машинного обучения и создания экосистемы на основе Spark. Мы наблюдали ранее и видим сейчас, как проект с открытым исходным кодом быстро становится источником инноваций в широком масштабе.

Эта книга погружает вас в мир Spark. В ней рассматривается мощь этой технологии и гибкость ее экосистемы, а также практические приложения Spark для работы в конкретной современной компании. Вне зависимости от того, являетесь ли вы инженером по обработке данных, научным исследователем в области обработки данных, разработчиком прикладных программ или инженером поддержки ИТ-операций, эта книга расскажет об инструментальных средствах и откроет секреты, которые вы должны знать, чтобы управлять инновациями в своей компании или сообществе.

Наша стратегия в IBM – создание собственных проектов на основе и с использованием успешных открытых платформ, но эти проекты должны быть значительными и должны выделяться на общем фоне. Spark является именно такой платформой. В IBM можно найти множество примеров применения этой стратегии, и вы получите такой же результат в своей компании, если выберете этот путь.

Spark можно считать инновацией – это аналитическая операционная система, в которой будут успешно развиваться новые решения, устраняющие сдерживающий фактор обработки больших данных. Кроме того, Spark – это сообщество высококвалифицированных ученых и аналитиков в области обработки данных, хорошо знающих Spark, которые могут быстро превратить любые сегодняшние задачи в завтрашние решения. Spark представляет собой один из самых быстро развивающихся проектов с открытым исходным кодом в истории ИТ. Присоединяйтесь к этому движению.

**– Роб Томас (Rob Thomas)**,
первый вице-президент,
подразделение Cloud and Data Platform, IBM

# *Предисловие*

Не думаю, чтобы для Apache Spark требовалось какое-то предварительное представление. Если вы читаете эти строки, то, вероятнее всего, уже кое-что знаете о теме этой книги: инженерия данных и наука о данных, применяемая в крупных масштабах с использованием распределенной обработки. Но Spark – это нечто большее, и об этом вы скоро узнаете, прочитав вступительное слово Роба Томаса и главу 1.

Как Обеликс упал в котел с магическим зельем[1], я «упал в котел» Spark в 2015 году. В то время я работал во французской компании, производящей аппаратные компоненты для компьютеров, где участвовал в проектировании высокопроизводительных систем для анализа данных. Вполне естественно, первое впечатление от Spark было скептическим. Затем я начал постоянно работать со Spark и понял, что результат зависит от самого пользователя. Первоначальный скептицизм превратился в настоящее глубокое увлечение превосходным инструментом, который позволяет обрабатывать данные чрезвычайно простым способом – это мое искреннее убеждение.

Я начал работу над несколькими проектами с использованием Spark, и это позволило мне выступить с сообщениями на Spark Summit, IBM Think, а также сблизиться с инициативами All Things Open, Open Source 101. Кроме того, через местную пользовательскую группу Spark я принял участие в анимации региона Роли-Дурэм в Северной Каролине. Это позволило мне встретиться с замечательными людьми и наблюдать за многочисленными проектами, связанными со Spark. В результате я увлекся Spark еще больше.

В этой книге я попытался поделиться с вами своим увлечением.

---

[1] Обеликс (Obelix) – известный персонаж комиксов и мультфильмов. Обеликс – неизменный спутник Астерикса (Asterix). Когда галл Астерикс выпивает магическое зелье, он получает суперсилу, которая помогает ему постоянно побеждать римлян (и пиратов). В раннем детстве Обеликс упал в котел, в котором варилось магическое зелье, воздействие которого на Обеликса стало постоянным. В Европе комиксы (и мультфильмы) про Астерикса (и Обеликса) весьма популярны. Больше информации см. на www.asterix.com/en/.

Примеры (или домашние задания) в этой книге основаны на использовании языка Java, но в репозитории книги содержится также код на языках Scala и Python. Сразу после выпуска версии Spark 3.0 сотрудники издательства Manning и я решили убедиться в том, что книга соответствует самым свежим версиям, а не отстает от действительности.

Как вы могли догадаться, мне очень нравятся комиксы. Я вырос на комиксах. Мне нравится такой способ передачи информации, в чем вы можете убедиться сами, читая эту книгу. Это не книжка комиксов, но около 200 иллюстраций в ней должны помочь вам понять великолепный инструмент, который называется Apache Spark.

Спутником Астерикса является Обеликс, точно так же и у книги «Spark на практике» (второе издание) есть спутник – постоянные средства поддержки, которые можно загрузить бесплатно из раздела ресурсов сайта издательства Manning по быстрой ссылке: http://jgp.net/sia. Эти средства поддержки содержат справочную информацию о статических функциях Spark и в итоге должны превратиться в более полезные справочные ресурсы.

Неважно, понравится вам книга или нет, в любом случае отправьте мне сообщение в твиттер на @jgperrin. Если книга понравилась, напишите отзыв на Amazon. Если книга не понравилась, то, как говорят при бракосочетании, (скажите сейчас) или храните молчание вечно. И все же я надеюсь, что книга вам понравится.

Alea iacta est – Жребий брошен[1].

[1] По-английски это высказывание звучит так: «The die is cast». Эту фразу приписывают Юлию Цезарю (заклятому врагу Астерикса), так как именно Цезарь перевел свою армию через Рубикон: событие произошло, и отменить его невозможно – как в случае, когда эта книга вышла из печати и стала доступна читателям.

# О чем эта книга

Когда я начинал работу над этим проектом, который стал книгой «Spark на практике», второе издание, моими целями были:

- помощь сообществу Java в использовании Apache Spark с наглядной демонстрации того, что нет необходимости дополнительно изучать языки Scala и/или Python;
- описание главных концепций, лежащих в основе Apache Spark, инженерии (больших) данных и науке о данных, требующих только знаний о реляционных базах данных и основ языка SQL, и ничего больше;
- тщательное разъяснение того, что Spark – это операционная система, специально предназначенная для распределенных вычислений и анализа.

Я верю в метод обучения по любой теме из области информационных технологий с использованием большого количества примеров. Примеры в этой книге являются чрезвычайно важной частью процесса обучения. Я создавал примеры так, чтобы они были как можно ближе к ситуациям, возникающим в настоящей профессиональной деятельности. Предлагаемые здесь наборы данных взяты из реальных ситуаций, со всеми присущими им качественными недостатками. Это не идеализированные учебные наборы данных, которые «работают всегда». Именно поэтому, объединяя примеры и такие наборы данных, вы будете работать и учиться более практическим способом по сравнению со «стерилизованным» подходом к обучению. Я называю эти примеры лабораторными работами (labs) с надеждой, что они окажутся интересными для вас и вы захотите поэкспериментировать с ними.

В книге очень много иллюстраций. Основываясь на широко известном высказывании «Рисунок заменяет тысячу слов», я избавил вас от чтения 183 000 дополнительных слов.

## Для кого предназначена эта книга

Трудно связать название профессии с названием книги, поэтому если ваша профессия называется инженер по обработке данных, ученый-ис-

следователь в области обработки данных, инженер программного обеспечения или архитектор программного обеспечения/данных, то книга определенно подходит вам. Если вы архитектор корпоративных приложений, ну тогда, вероятно, вам все это известно, ведь архитектор корпоративных приложений знает все обо всем, не так ли? А если говорить более серьезно, эта книга будет полезной, если вы хотите получить больше знаний по любой из следующих тем:

- использование Apache Spark для создания конвейеров данных и их анализа: потребление (ingestion), преобразование и экспортирование/публикация;
- использование Spark без необходимости изучения Scala или Hadoop: изучение Spark с Java;
- понимание различий между реляционной базой данных и Spark;
- главные концепции, заложенные в основу больших данных, включая основные компоненты Hadoop, которые можно встретить в рабочей среде Spark;
- позиционирование Spark в корпоративной архитектуре;
- использование имеющихся навыков и умений по работе с Java и СУРБД в среде больших данных;
- понимание API фрейма данных (dataframe);
- объединение реляционных баз данных с процедурами потребления данных в Spark;
- сбор и объединение данных из потоков;
- понимание развития конкретной области промышленности и причин полного соответствия Spark этому процессу;
- понимание и практическое использование центральной роли фрейма данных;
- знание того, что представляют собой устойчивые распределенные наборы данных (resilient distributed datasets – RDD) и почему их не следует использовать (в дальнейшем);
- понимание процесса взаимодействия с рабочей средой Spark;
- понимание различных компонент Spark: драйвер, исполнители, ведущий узел, рабочие узлы, Catalyst, Tungsten;
- изучение роли ключевых технологий, производных от Hadoop, таких как YARN или HDFS;
- понимание роли диспетчера ресурсов, например YARN, Mesos и встроенного диспетчера;
- потребление данных из различных файлов в пакетном режиме и из потоков;
- использование SQL совместно со Spark;
- работа со статическими функциями, предоставляемыми Spark;
- понимание, что такое неизменяемость (immutability) и почему она так важна;
- расширение Spark с помощью функций, определяемых пользователем (UDF), в языке Java;
- расширение Spark с помощью новых источников данных;

- линеаризация данных из формата JSON, чтобы можно было использовать язык SQL;
- выполнение агрегаций и объединений в фреймах данных;
- расширение агрегации с помощью функций агрегации, определяемых пользователем (UDAF);
- понимание различий между кешированием и механизмом копирования данных в контрольных точках (checkpointing) и увеличение производительности конкретных приложений Spark;
- экспорт данных в файлы и базы данных;
- понимание процесса развертывания в средах AWS, Azure, IBM Cloud, GCP и в собственных локальных кластерах;
- потребление данных из файлов с форматами CSV, XML, JSON, текст, Parquet, ORC и Avro;
- расширение источников данных с примером: как потребляются метаданные фотографий с использованием EXIF и сосредоточением внимания на прикладном интерфейсе Data Source API v1;
- использование Delta Lake совместно со Spark при создании конвейеров.

## *Чему вы научитесь, читая эту книгу*

Цель этой книги – научить вас практическому использованию Spark в конкретных приложениях или созданию приложений специально для Spark.

Эта книга предназначена для инженеров по обработке данных и для инженеров программного обеспечения Java. Когда я начинал изучать Spark, все программное обеспечение было написано на Scala, почти вся документация находилась на официальном сайте проекта, а вопросы по Spark появлялись чрезвычайно редко. В документации утверждалось, что для Spark имеется Java API, но подробных развернутых примеров было совсем немного. В то время мои соратники находились в замешательстве, вызванном необходимостью изучения Spark и Scala, а наше руководство требовало результатов. Члены моей группы тогда стали основным стимулом для написания этой книги.

Предполагается, что читатель обладает основами знаний о языке Java и о СУРБД (о системах управления реляционными базами данных). Во всех примерах я использовал версию Java 8, хотя уже существует версия Java 11.

Для чтения этой книги не нужны какие-либо знания о Hadoop, но, поскольку вам все же потребуются некоторые компоненты Hadoop (лишь немногие), здесь они рассматриваются. Если вы уже знаете Hadoop, то книга поможет освежить эти знания. Нет необходимости в знании языка Scala, так как эта книга о Spark и Java.

Когда я был ребенком (должен признать, и сейчас тоже), то читал много французских комиксов (bandes dessinées), это нечто среднее между обычными комиксами и романом в комиксах. Поэтому мне очень нравятся иллюстрации, и я включил огромное количество иллюстраций в эту книгу. На рис. 1 показана обычная для этой книги графическая схе-

ма, демонстрирующая несколько компонент, значков, условных обозначений и описаний.

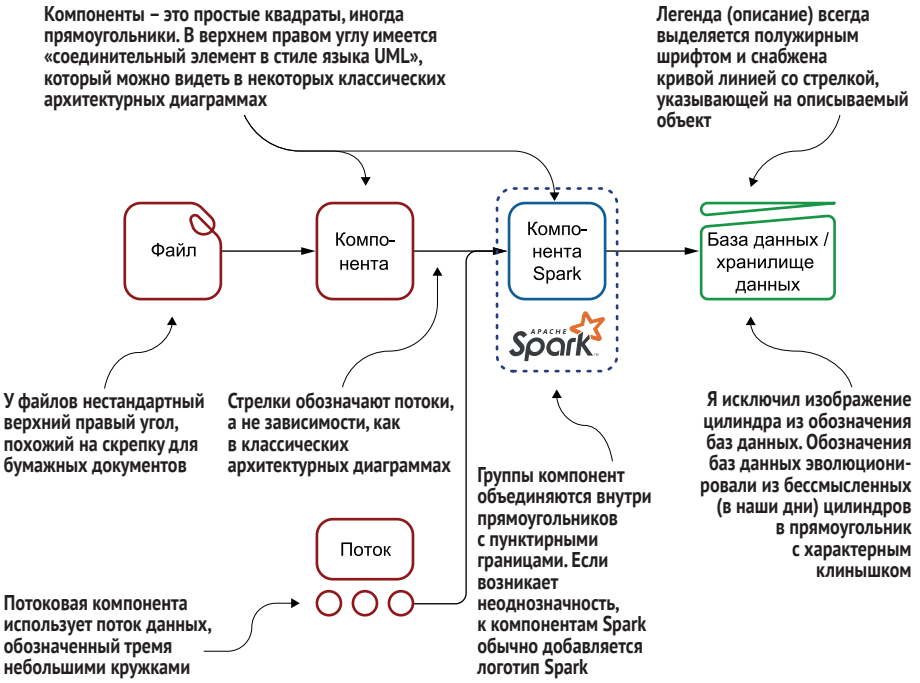


**Рис. 1 Графические элементы и описания, используемые на иллюстрациях в этой книге**

## Как организована книга

Эта книга разделена на четыре части и 18 приложений.

Часть I предлагает основополагающую информацию о Spark. Вы будете изучать теорию и общие концепции, но не спешите впадать в отчаяние (пока еще рано) – здесь представлено множество примеров и графических схем. Эта часть читается почти как книжка комиксов. Часть I содержит следующие главы:

- глава 1 – общее введение с простым примером. Вы узнаете, почему Spark представляет собой распределенную аналитическую операционную систему;
- глава 2 – подробное описание простого процесса Spark;
- глава 3 – объяснение великолепных свойств фрейма данных, который объединяет API и возможности сохранения данных Spark;
- глава 4 – вознесение хвалы лени (ленивым, или отложенным, вычислениям), сравнение Spark и СУРБД, а также необходимая общая информация о направленном ациклическом графе (НАГ);
- главы 5 и 6 взаимосвязаны – вы напишете небольшое приложение, создадите кластер и развернете это приложение. В главе 5 описано

создание небольшого приложения, а в главе 6 рассматривается его развертывание.

В части 2 начинается разделение практических и прагматических примеров, демонстрирующих потребление данных. Потребление (ingestion)[1] – это процесс переноса данных в Spark. Это несложный процесс, но для него существует огромное количество возможностей и комбинаций. Часть II содержит следующие главы:

- глава 7 – описание процесса потребления данных из файлов: CSV, текст, JSON, XML, Avro, ORC и Parquet. Для каждого формата файла приводится отдельный пример;
- глава 8 – рассматривается потребление из баз данных: данные берутся из реляционных баз данных и других хранилищ данных;
- глава 9 – потребление из источников данных, устанавливаемых пользователем;
- глава 10 – обработка потоковых данных.

В части 3 описано преобразование данных: я назвал бы это «поднятием тяжелых данных». Рассматривается качество данных, их преобразование и публикация обработанных данных. Эта самая большая часть книги рассказывает об использовании фрейма данных в совокупности с языком SQL, а также с собственным прикладным интерфейсом API, агрегатами, кешированием и расширением Spark с помощью функций, определяемых пользователем (UDF):

- глава 11 – описание широко известного языка запросов SQL;
- глава 12 – рассматривается выполнение преобразований;
- глава 13 – расширение преобразований до уровня всего документа в целом. В этой главе также описываются статические функции, которые являются одним из многих замечательных инструментов Spark;
- глава 14 – расширение Spark с помощью функций, определяемых пользователем (UDF);
- агрегаты также представляют собой широко известную концепцию баз данных и могут оказаться весьма важными для анализа. В главе 15 рассматриваются агрегаты, в том числе включенные в Spark и специализированные агрегаты.

Заключительная часть IV приближает читателя к реальной производственной среде и рассматривает более продвинутые темы. Вы узнаете о распределении данных в кластере, об экспорте данных, об ограничениях процесса развертывания (включая развертывание в облаке) и об оптимизации:

- глава 16 – описание методик оптимизации: кеширование и механизм копирования данных в контрольных точках;
- глава 17 – экспорт данных в файлы и базы данных. В этой главе так-

[1] Дословный перевод термина «ingestion» – потребление, прием пищи, поглощение, всасывание. – *Прим. перев.*

же описано использование Data Lake, базы данных, размещаемой рядом с ядром Spark;
- глава 18 – подробная справочная информация об архитектурах и методах обеспечения безопасности, необходимых для развертывания. Эта глава в меньшей степени связана с практикой, но содержит весьма важную информацию.

Приложения не так важны, но содержат большой объем информации: об установке, об устранении возникающих проблем и ошибок, а также о создании рабочего контекста. Почти все приложения содержат справочную информацию по Apache Spark в контексте Java.

## Исходный код примеров

Как уже было отмечено ранее, каждая глава (за исключением 6 и 18) содержит лабораторные работы, в которых объединен исходный код и данные. Исходный код приводится в пронумерованных листингах и в отдельных строках обычного текста. В обоих случаях исходный код выделяется моноширинным шрифтом, как здесь, чтобы отделить его от обычного текста. Иногда некоторые фрагменты кода кроме моноширинного обозначаются еще и **полужирным** шрифтом, чтобы выделить ту часть кода, которая является более важной в блоке кода.

Весь исходный код свободно и бесплатно доступен в репозитории GitHub под лицензией Apache 2.0. Данные могут быть защищены различными лицензиями. Для каждой главы создан собственный отдельный репозиторий GitHub: для главы 1 – https://github.com/jgperrin/net.jgp.books.spark.ch01, для главы 15 – https://github.com/jgperrin/net.jgp.books.spark.ch15 и т. д. Два исключения:

- в главе 6 используется исходный код из главы 5;
- в главе 18, где подробно рассматривается процесс развертывания, нет исходного кода.

Так как инструментальные средства управления исходным кодом позволяют создавать ответвления версий, главная ветвь содержит исходный код самой последней готовой к реальному использованию версии, тогда как каждый репозиторий содержит ветви для специализированных или пробных версий, когда это необходимо.

Лабораторные работы пронумерованы трехзначными числами, начиная со 100. Предлагается два типа лабораторных работ: лабораторные работы, описанные в книге, и дополнительные лабораторные работы, доступные в режиме онлайн:

- лабораторные работы, описанные в книге, нумеруются по разделам соответствующей главы. Таким образом, лабораторная работа #200 главы 12 находится в главе 12 в разделе 2. А лабораторная работа #100 главы 17 описывается в первом разделе главы 17;
- лабораторные работы, не описанные в книге, нумеруются числами, начинающимися с цифры 9, т. е. 900, 910 и т. д. Группа с номерами, начиная с 900, постоянно увеличивается: я добавляю в нее новые

лабораторные работы. Нумерация этих лабораторных работ не является непрерывной – как нумерация строк кода Basic.

В репозиториях GitHub вы найдете исходный код на Python, Scala и Java (кроме тех случаев, когда код на каком-либо из этих языков неприменим). Но в книге используется только Java для сохранения ясности.

Во многих случаях исходный код, приведенный в книге, требует переформатирования: в код добавлены разрывы строк и изменено выравнивание строк для соответствующего размещения исходного кода на доступном пространстве страницы книги. В редких случаях, когда даже вышеописанных мер недостаточно, в листинги включены маркеры продолжения строки (➥). Кроме того, комментарии к исходному коду были удалены из листингов в тех случаях, когда код описывается в тексте. Во многих листингах к исходному коду приложены примечания, объясняющие важные концепции.

## Форум для обсуждения книги liveBook

Приобретение книги «Spark на практике» дает право на свободный доступ на частный веб-форум издательства Manning Publications, где вы можете написать комментарии по этой книге, задать технические вопросы и получить помощь от автора и других пользователей. Этот форум расположен по адресу https://livebook.manning.com/#!/book/spark-in-action-second-edition/discussion. Вы можете также узнать больше о форумах Manning и правилах их использования по адресу https://livebook.manning.com/#!/discussion.

К обязательствам издательства Manning по отношению к читателям относится предоставление места, где обеспечивается разумный диалог между отдельными читателями, а также между читателями и автором книги. Но при этом автор не обязан обеспечивать какой-либо конкретный объем своего участия в обсуждении, его участие в этом форуме является добровольным (и не оплачивается). Мы предлагаем читателям попытаться задать автору действительно трудные и интересные вопросы, чтобы его интерес к форуму не угас. Форум и архивы предыдущих обсуждений будут доступными на сайте издателя сразу после выхода книги из печати.

## Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу

http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

## Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

## Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

## Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Wiley очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

# *Об авторе*

Жан-Жорж Перрен (Jean-Georges Perrin) страстно увлечен инженерией программного обеспечения и всем, что касается использования данных. В своих последних проектах он в еще большей степени устремлен к идее инженерии распределенной обработки данных. В этих проектах Жан-Жорж активно использует Apache Spark, Java и другие инструментальные средства в гибридных облачных средах. Он гордится тем, что первым во Франции был удостоен звания IBM Champion на двенадцатом году непрерывной работы в этой компании. Как признанный эксперт в области обработки данных и инженерии программного обеспечения, в настоящее время Жан-Жорж работает во многих странах мира, но чаще всего в США, где он проживает. Своим более чем 25-летним опытом в области информационных технологий Жан-Жорж делится как участник конференций и автор статей в печатных изданиях и в интернете. Вы можете посетить его блог по адресу http://jgp.net.

# Часть I

# Теория, разбавленная превосходными примерами

При освоении любой технологии необходимо хотя бы немного знать и понимать «скучную» теорию, прежде чем приступать к практическому использованию. Я включил в эту часть шесть глав, которые предоставят подробный обзор теоретических концепций, объясняемых на примерах.

В главе 1 представлено общее введение с простым примером. Вы узнаете, почему Spark представляет собой не просто обычный набор инструментов, а настоящую распределенную аналитическую операционную систему. После прочтения первой главы вы будете готовы к выполнению простой процедуры потребления данных в Spark.

Глава 2 продемонстрирует, как работает Spark на высоком (внешнем) уровне. Вы постепенно выстроите представление о компонентах Spark, формируя мысленную модель (представляющую ваш собственный мыслительный процесс) шаг за шагом. Лабораторная работа в этой главе продемонстрирует, как экспортировать данные в базу данных. В этой главе очень много иллюстраций, которые должны способствовать процессу обучения, дополняя текст и исходный код.

Глава 3 помещает вас в совершенно новое измерение: исследование мощного фрейма данных, который объединяет прикладной интерфейс API и возможности хранения данных Spark. В лабораторной работе этой главы вы загрузите два набора данных и объедините их.

В главе 4 воздается хвала лени (ленивым, или отложенным, вычислениям) и объясняется, почему Spark использует ленивую, или отложенную, оптимизацию. Будет описан направленный ациклический граф (НАГ; DAG), потом приведено сравнение Spark и СУРБД. Лабораторная работа научит вас основам обработки данных с использованием API фрейма данных.

Главы 5 и 6 взаимосвязаны: вы напишете небольшое приложение, создадите кластер и развернете в нем это приложение. В этих двух главах содержится только практическая информация.

# 1 Так что же такое Spark?

***Краткое содержание главы:***

- что такое Spark и где он используется;
- основы технологии распределенной обработки данных;
- четыре столпа Spark;
- хранилище данных и прикладные интерфейсы API: великолепный фрейм данных.

Когда я был ребенком в 1980-х годах, открывающим для себя мир программирования с помощью языка Basic и домашнего компьютера Atari, я не мог понять, почему невозможно автоматизировать основные законы, регулирующие деятельность, как, например, ограничение скорости, нарушения правил на светофоре и счетчик оплаты за стоянку автомобиля. Все выглядело достаточно простым – в книге, которую я читал, было сказано: чтобы стать хорошим программистом, нужно избегать использования операторов `GOTO`. Именно так я и поступил, пытаясь структурировать свой код, начиная с 12 лет. Но я никоим образом не мог вообразить реальный объем обрабатываемых данных (и появление и стремительное развитие интернета вещей – Internet of Things, IoT), когда разрабатывал игру в стиле Монополии. Поскольку моя игра умещалась в 64 Кб оперативной памяти, я даже и представить себе не мог, что наборы данных должны будут увеличиваться (с гигантским коэффициентом увеличения) или что данные могут обладать скоростью, и терпеливо ждал, когда моя игра сохранится на магнитной ленте в накопителе Atari 1010.

Быстро пролетели 35 лет, и все способы автоматизации, о которых я мечтал, стали доступными (а моя игра стала пустяковым «детским лепетом»). Объемы данных росли быстрее, чем технологии аппаратного

обеспечения для поддержки их обработки[1]. Кластер из небольших (настольных) компьютеров мог стоить дешевле, чем один большой компьютер (мейнфрейм). Память подешевела в два раза по сравнению с 2005 годом, а память в 2005 году была в пять раз дешевле, чем в 2000-м[2]. Сети стали быстрее в несколько раз, а современные центры обработки данных (datacenters) предлагают скорость до 100 Гб/с (Gbps), это приблизительно в 2000 раз быстрее, чем ваш домашний Wi-Fi пять лет назад. Это были лишь некоторые из причин, заставивших людей задать следующий вопрос: как можно использовать вычисления с распределенной памятью для анализа больших объемов данных?

Когда вы читаете техническую литературу или ищете в вебе информацию об Apache Spark, возможно, вам встретятся утверждения о том, что это инструмент для обработки больших данных, преемник Hadoop, платформа для выполнения анализа, фреймворк (рабочая среда) на основе кластера компьютеров, и тому подобные высказывания. Que nenni[3]!

> **ЛАБОРАТОРНАЯ РАБОТА** Лабораторная работа в этой главе доступна в репозитории GitHub по адресу https://github.com/jgperrin/net.jgp.books.spark.ch01. Это лабораторная работа #400. Если вы не знакомы с сервисом GitHub и интегрированной средой разработки Eclipse, то в приложениях А, Б, В, Г вы найдете справочное руководство.

## 1.1 Общая картина: что такое Spark и что он делает

Подобно тому, как Маленький Принц сказал Антуану де Сент-Экзюпери: «Нарисуй мне барашка», можно было бы сказать: «Нарисуй мне Spark». В этом разделе вы узнаете о том, что такое Spark, а затем – что может делать Spark, на нескольких конкретных примерах его использования. Завершается этот раздел описанием того, как Spark интегрируется в качестве стека программного обеспечения с другими программными комплексами и используется учеными-исследователями в области обработки данных.

### 1.1.1 Что такое Spark

Spark – это не просто стек программного обеспечения для научных работников в области обработки данных. При создании приложений вы строите их «поверх» или на основе операционной системы, как показано на рис. 1.1. Операционная система предоставляет сервисы, упрощающие

---

[1] См. статью «Intel Puts the Brakes on Moore's Law» Тома Симоните (Tom Simonite) в MIT Technology Review, март 2016 года (http://mng.bz/gVj8).

[2] См. статью «Memory Prices (1957–2017)» Джона С. МакКаллума (John C. McCallum) (https://jcmit.net/memoryprice.htm).

[3] Средневековое французское выражение, означающее: «Разумеется, нет!»

разработку приложения, другими словами, вам не нужно формировать файловую систему или писать сетевой драйвер для каждого разрабатываемого приложения.

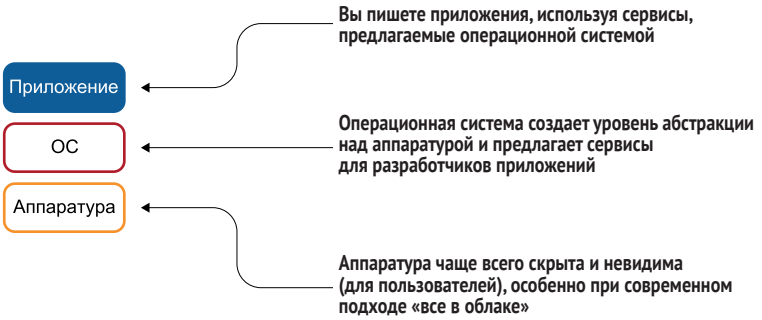


**Рис. 1.1 Когда вы пишете приложения, то используете сервисы, предлагаемые операционной системой, которая изолирует вас от прямого взаимодействия с аппаратурой**

Вместе с потребностью в большей вычислительной мощности возникает и постоянно возрастающая потребность в распределенных вычислениях (в распределенной обработке данных). После появления технологии распределенных вычислений в распределенные приложения должны были включаться соответствующие функции. На рис. 1.2 показано увеличение сложности при добавлении дополнительных компонент в распределенное приложение.

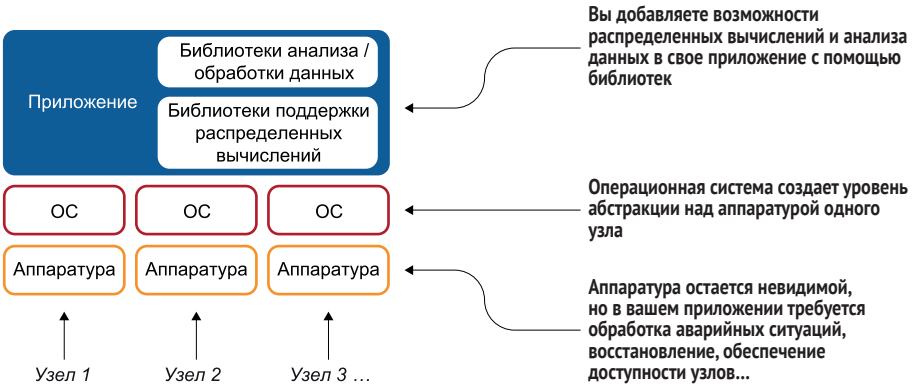


**Рис. 1.2 Один из способов создания распределенных приложений, специализированных для обработки данных, – встраивание всех управляющих элементов на уровне приложения с использованием библиотек или других программных компонент. В результате размер приложений увеличивается и их сопровождение усложняется**

После этого необходимо добавить, что Apache Spark может показаться сложной системой, для применения которой потребуется огромный

объем предварительных знаний. Я уверен, что вам необходимы только навыки работы с Java и системами управления реляционными базами данных (СУРБД) для понимания, использования и создания приложений, а также для расширения самой системы Spark.

Приложения тоже стали более интеллектуальными – генерируют отчеты и выполняют анализ данных (в том числе агрегацию данных, линейную регрессию или просто выводят кольцевые диаграммы). Таким образом, если нужно добавить аналитические функциональные возможности подобного рода в приложение, то необходимо подключить (связать) соответствующие библиотеки или написать свои. Из-за этого приложение увеличивается в размерах (или становится «толще», как «толстый клиент»), затрудняется его сопровождение, и само приложение становится более сложным и, как следствие, более дорогим для предприятия.

Вы можете спросить: «Так почему бы не переместить всю эту функциональность на уровень операционной системы?» Преимуществ перемещения этих функциональных свойств на более низкий уровень, например на уровень операционной системы, несколько, в том числе следующие:

- предоставление стандартного способа работы с данными (в определенной степени это похоже на язык структурированных запросов – Structured Query Language, SQL для реляционных баз данных);
- снижение стоимости разработки (и сопровождения) приложений;
- возможность сосредоточиться на том, как использовать конкретный инструмент, а не на том, как он работает (например, Spark выполняет распределенное потребление данных, и вы можете узнать, как извлечь преимущества из этого процесса без необходимости тщательного изучения того, как именно Spark выполняет эту задачу).

Это как раз то, чем Spark стал для меня: аналитической операционной системой. На рис. 1.3 показан этот программный стек в упрощенной форме.

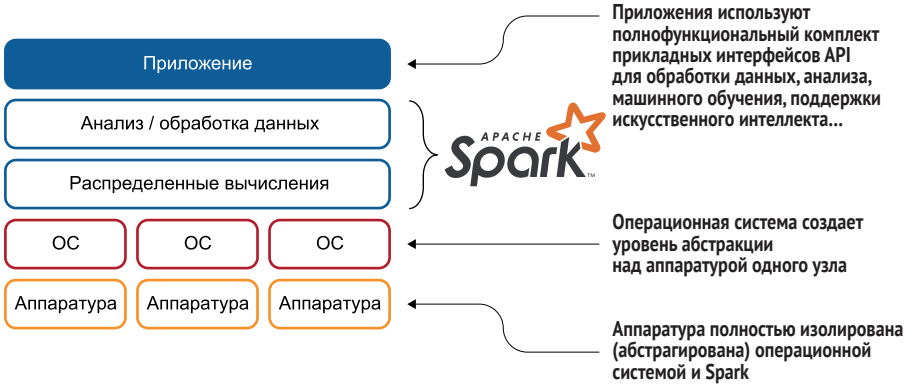


**Рис. 1.3 Apache Spark упрощает разработку приложений, специально предназначенных для анализа данных, предлагая сервисы для этих приложений точно так же, как операционная система**

В этой главе мы рассмотрим несколько конкретных примеров использования Apache Spark в различных отраслях промышленности и в проектах разных масштабов. Эти примеры помогут получить начальное представление о том, чего можно достичь.

Я твердо уверен в том, что для лучшего понимания современности мы должны обращаться к истории. Это в полной мере применимо и к сфере информационных технологий (ИТ): см. приложение Д, если вы принимаете мою точку зрения.

Теперь, когда декорации размещены на сцене, можно начать углубленное изучение Spark. Начнем с общего обзора, рассмотрим хранилища данных и прикладные интерфейсы API, а в конце главы разберем первый пример.

### 1.1.2 Четыре столпа маны

По представлениям полинезийцев, мана (mana) – это энергия основных сил природы, заключенная в объекте или в человеке. Это определение соответствует общей схеме, которую можно найти в комплекте документации Spark. Схема изображает четыре столпа, приносящие природные силы в Spark: Spark SQL, потоковый механизм Spark Streaming, Spark MLlib (для машинного обучения) и GraphX, стоящие на прочном фундаменте Spark Core. Это точное представление программного стека Spark, но я считаю его не совсем полным. Представление стека должно быть расширено, чтобы показать аппаратуру, операционную систему и само приложение, как на рис. 1.4.

Разумеется, кластер(ы), в котором работает Spark, возможно, не используется исключительно вашим приложением, но в своей работе вы будете использовать следующие компоненты:

- Spark SQL – для выполнения операций с данными, таких как обычные задания на языке SQL в СУРБД. Spark SQL предлагает прикладные интерфейсы API и SQL для обработки данных. Spark SQL будет рассматриваться в главе 11, а дополняться эта тема будет в большинстве последующих глав. Spark SQL – это краеугольный камень Spark;
- Spark Streaming – потоковый механизм Spark и особенно структурированный потоковый механизм Spark – предназначен для анализа потоковых данных. Стандартизированный API Spark поможет обрабатывать данные в одинаковом стиле независимо от того, являются ли данные потоковыми или пакетными. Подробнее о потоковой обработке данных вы узнаете в главе 10;
- Spark MLlib – для машинного обучения и дальнейших расширений для глубокого обучения. Машинное обучение, глубокое обучение и искусственный интеллект заслуживают отдельной книги;
- GraphX – для интенсивного использования графовых структур данных. Чтобы узнать больше о GraphX, вы можете прочесть книгу «Spark GraphX in Action» Майкла Малака (Michael Malak) и Робина Иста (Robin East) (Manning, 2016).

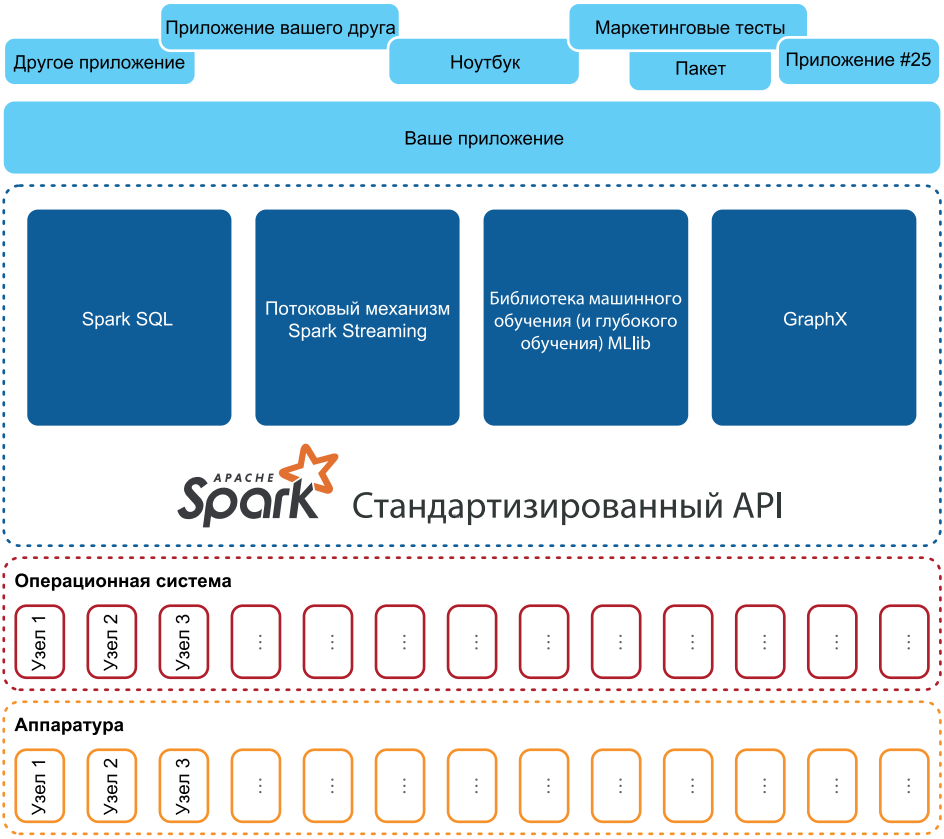


**Рис. 1.4 Ваше приложение, как и другие приложения, имеет дело с четырьмя столпами Spark – SQL, потоковым механизмом, машинным обучением и графами – через стандартизированный прикладной интерфейс API. Spark защищает вас от ограничений, присущих операционной системе и аппаратуре: нет необходимости заботиться о том, где запускается приложение или предоставлены ли ему правильные данные. Spark позаботится об этом. Но приложение может получить доступ к операционной системе или к аппаратуре, если это действительно необходимо**

## 1.2 Как можно использовать Spark

В этом разделе подробно рассматривается, как можно использовать Spark, и основное внимание сосредоточено на обычных вариантах обработки данных, а также на подходе с использованием науки о данных. Неважно, являетесь ли вы инженером по обработке данных или ученым-исследователем в области обработки данных, в любом случае вы сможете воспользоваться Apache Spark в своей работе.

### 1.2.1 Spark в процессе обработки данных / инженерии данных

Spark может обрабатывать данные несколькими различными способами. Но он превосходен при обработке больших данных, когда данные

потребляются, очищаются, преобразовываются и публикуются результаты обработки.

Я предпочитаю определять инженеров по обработке данных как специалистов по подготовке данных и по обеспечению и организации процесса обработки данных. Они обеспечивают доступность и готовность данных, надлежащее применение правил по соблюдению качества обработки данных, успешное выполнение преобразований и доступность данных для других систем и подразделений, включая бизнес-аналитиков и ученых-исследователей в области обработки данных. Инженеры по обработке данных также могут быть теми специалистами, которые принимают результаты работы ученых-исследователей и внедряют их в производственную практику.

Spark – превосходный инструмент для инженеров по обработке данных. Ниже перечислены четыре этапа обычного варианта использования Spark (с большими данными) в процессе инжиниринга данных.

1. Потребление данных.
2. Улучшение качества данных (data quality – DQ).
3. Преобразование данных.
4. Публикация результатов обработки.

На рис. 1.5 показан этот процесс.


**Рис. 1.5 Spark в обычном варианте обработки данных. Первый этап – потребление данных. На этом этапе исходные данные «сырые» (необработанные), возможно, далее потребуется применение некоторых методов улучшения качества данных (DQ). После этого данные готовы к преобразованию. После преобразования данные становятся «обогащенными». Наступает время их публикации или совместного использования, чтобы сотрудники вашей организации могли выполнять какие-либо действия с этими данными и принимать решения на их основе**

Показанный на рис. 1.5 процесс включает четыре этапа, и после каждого этапа данные перемещаются в соответствующую зону (zone):

1. потребление данных – Spark может потреблять данные из разнообразных источников (о потреблении данных см. главы 7, 8 и 9).

Если невозможно найти поддерживаемый формат, то можно создать собственные источники данных. На этом этапе я называю данные исходными («сырыми», необработанными – raw data). Встречаются также следующие названия этой зоны: сосредоточение (staging), посадочная площадка (landing), бронзовая (bronze) зона или даже болото (swamp);

2. улучшение качества данных – перед обработкой данных, вероятно, потребуется проверка их качества. Пример обеспечения качества данных (DQ) – проверка с целью убедиться в том, что все даты рождения указаны в прошлом. Частью этого процесса также может стать выборочное скрытие некоторых данных: при обработке номеров социальной страховки (SSN) в сфере здравоохранения можно обеспечить недоступность SSN для разработчиков и для неавторизованных сотрудников[1]. Этап после улучшения качества данных я называю зоной очищенных данных (pure data). Встречаются также следующие названия этой зоны: рафинировочная (refinery), серебряная (silver), бассейн (pond), песочница (sandbox) или зона обследования (exploration zone);
3. преобразование данных – следующий этап – обработка данных. Можно объединять данные с другими наборами данных, применять специализированные функции, выполнять операции агрегирования, реализовывать методы машинного обучения и много другое. Цель этого этапа – получение «обогащенных» (обработанных) данных (rich data) как результата аналитической работы. В большинстве глав этой книги обсуждается преобразование данных. Эта зона может также называться продукционной (production), золотой (gold), зоной улучшенных данных (refined), лагуной (lagoon) или эксплуатационной зоной (operationalization zone);
4. загрузка и публикация – как и в процессе ETL[2], на завершающем этапе можно загрузить данные в хранилище данных (data warehouse), используя инструментальное средство бизнес-интеллекта (business intelligence – BI), вызывая прикладные интерфейсы API или сохраняя данные в файле. Результатом являются данные, готовые к использованию в конкретной организации (на предприятии).

---

[1] Если вы проживаете не в США, то должны понять, насколько важен номер социальной страховки SSN. Фактически SSN управляет всей жизнью владельца. С его первоначальной целью – идентификатор для получения социальных льгот – связи почти не осталось: в настоящее время SSN стал идентификатором для уплаты налогов и для обеспечения анонимности финансовых операций, а также для отслеживания деятельности людей. Злоумышленники специально ищут номера SSN и другие личные данные людей, чтобы открывать от их имени банковские счета или получить доступ к существующим финансовым счетам.

[2] ETL – Extract, Transform, Load – извлечение, преобразование, загрузка – основной (классический) процесс в крупном хранилище данных (data warehouse).

### 1.2.2 Spark в научных исследованиях в области обработки данных

Ученые-исследователи в области обработки данных применяют методики, немного отличающиеся от подхода инженеров программного обеспечения или инженеров по обработке данных, поскольку главное внимание ученых сосредоточено на этапе преобразования данных, осуществляемом в интерактивном режиме. Для этой цели исследователи в области обработки данных используют разнообразные инструментальные средства, например такие как виртуальные блокнотные среды (notebook) Jupyter, Zeppelin, IBM Watson Studio и Databricks Runtime.

Результаты исследований в области обработки данных, несомненно, будут иметь значение для вас, но в исследовательских проектах обработки данных будут использоваться корпоративные данные, так что на завершающем этапе данные могут передаваться ученым, а результаты их исследований (такие как модели машинного обучения) применяются в корпоративных хранилищах данных или в процессе внедрения в производство исследовательских методов.

Таким образом, диаграмма последовательностей в стиле UML, показанная на рис. 1.6, немного точнее объясняет, как ученые-исследователи в области обработки данных используют Spark.

Если вы хотите больше знать о Spark и науке о данных, то можете ознакомиться со следующими книгами:

- «PySpark in Action», автор Джонатан Риу (Jonathan Rioux) (Manning, 2020, www.manning.com/books/pyspark-in-action?a_aid=jgp);
- «Mastering Large Datasets with Python», автор Джон Т. Уолохан (John T. Wolohan) (Manning, 2020, www.manning.com/books/mastering-large-datasets-with-python?a_aid=jgp).

В примере использования, показанном на рис. 1.6, данные загружаются в Spark, затем пользователь работает с ними, применяя методы преобразования, и выводит (предъявляет) часть этих данных. Вывод данных не означает завершение процесса. Пользователю предоставляется возможность продолжать процесс в интерактивном режиме, как в обычном бумажном блокноте, где записываются рецепты, заметки и т. п. На завершающем этапе процесса пользователь виртуальной блокнотной среды может сохранить данные в файлах или в базах данных или сформировать (интерактивные) отчеты.

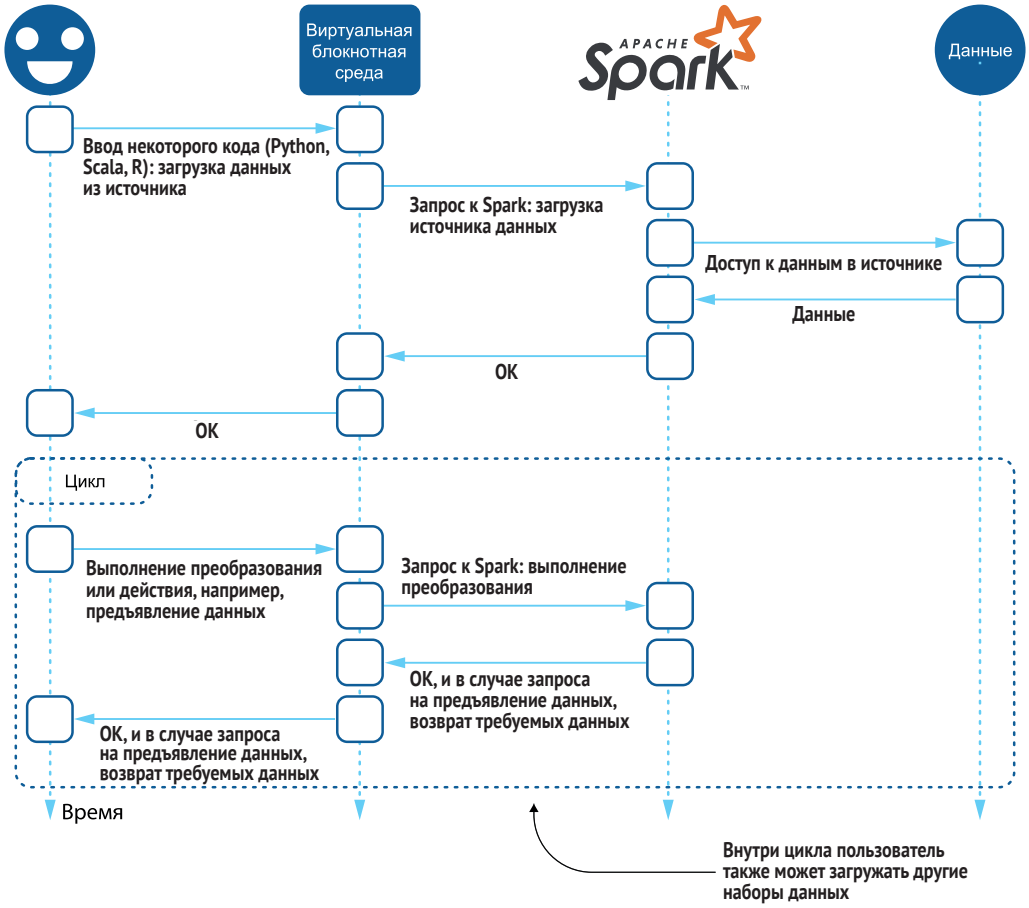


**Рис. 1.6 Диаграмма последовательностей для ученого-исследователя в области обработки данных, использующего Spark: пользователь «разговаривает» с виртуальной блокнотной средой (notebook), которая вызывает Spark при необходимости. Spark напрямую выполняет операции потребления данных. Каждый квадрат обозначает конкретный этап, каждая стрелка представляет последовательность действий. Такую диаграмму следует читать в хронологическом порядке, начиная с ее вершины**

## 1.3 Что можно делать с помощью Spark

Spark используется в проектах разнообразных типов, поэтому мы рассмотрим лишь некоторые такие проекты. Во всех вариантах использования подразумеваются данные, которые невозможно хранить и обрабатывать на одном компьютере (т. е. большие данные), таким образом, требуется кластер компьютеров, следовательно, необходима распределенная операционная система, специализированная для анализа данных.

Определение больших данных (big data) со временем изменялось – от данных с характеристиками, известных как «пять V»[1], до определения «данных, которые невозможно разместить на одном компьютере». Мне не нравится это определение – вероятно, вам известно, что многие СУРБД разделяют хранимые данные между несколькими серверами. Как и для многих теоретических концепций, скорее всего, потребуется собственное определение. Надеюсь, что книга поможет вам в этом.

Для меня большие данные — это комплект наборов данных, доступных в любом пункте корпоративной среды, собранных (агрегированных) в одной локации, в которой можно выполнять основные аналитические операции для реализации более сложных аналитических функций, таких как машинное обучение и глубокое обучение. Такие укрупненные наборы данных могут стать основой для методик искусственного интеллекта (ИИ; artificial intelligence – AI). Для этой концепции абсолютно не важны конкретные технологии, размер и количество компьютеров.

Spark, благодаря своим аналитическим функциональным возможностям и собственной распределенной архитектуре, может обрабатывать большие данные независимо от того, считаете ли вы данные действительно большими, и от возможности их размещения на одном или на нескольких (или даже многих) компьютерах. Просто нужно помнить о том, что вывод обычного отчета на широком 132-знаковом матричном принтере – это не самый типичный вариант использования Spark. Рассмотрим несколько примеров из реальной практической деятельности.

### 1.3.1 Spark прогнозирует качество пунктов питания Северной Каролины

Почти везде в США для ресторанов (и прочих пунктов питания) обязательны инспекторские проверки местными отделами здравоохранения для проверки работы этих заведений и оценки на основе таких проверок. Более высокая оценка означает не лучшее качество пищи, а позволяет узнать, не подвергаете ли вы опасности свою жизнь, съев барбекю в какой-то забегаловке во время поездки в южные штаты. Оценивается чистота кухни, правильность хранения продуктов и многие другие критерии, для того чтобы (как все мы надеемся) избежать пищевых отравлений и прочих недомоганий.

В Северной Каролине рестораны оцениваются по шкале от 0 до 100. Каждый округ предоставляет доступ к ресторанным оценкам, но централизованный пункт доступа к информации по всему штату отсутствует.

---

[1] Пять V (five Vs) – это volume – объем (количество генерируемых и сохраняемых данных), variety – разнообразие (тип и природа данных), velocity – скорость (с которой данные генерируются и обрабатываются), variability – непостоянство (несогласованность или отсутствие целостности в наборе данных) и veracity – достоверность (качество данных может изменяться в весьма широких пределах); определение сформировано на основе информации из «Википедии» и от компании IBM.

NCEatery.com – это ориентированный на клиентов веб-сайт, на котором расположен список ресторанов с инспекторскими оценками за некоторый интервал времени. Цель этого сайта – централизованный доступ к информации и выполнение прогностического анализа по ресторанам, чтобы клиенты могли найти образцы качественных ресторанов. «*Неужели то место, которое так понравилось мне два года назад, скатывается все ниже и ниже?*»

Во внутренней компоненте этого сайта Apache Spark потребляет наборы данных о ресторанах, инспекторских проверках и данные о нарушениях норм и правил, приходящие из различных округов, перемалывает эти данные и публикует сводные отчеты о результатах обработки на сайте. На этапе обработки применяются некоторые методики улучшения качества данных, а также методы машинного обучения, чтобы попытаться спрогнозировать результаты инспекторских проверок и оценки. Spark обрабатывает $1{,}6\times10^{21}$ элементов (точек) данных и публикует около 2500 страниц каждые 18 ч, используя для этого небольшой кластер. Этот постоянно развивающийся проект включает в процесс обработки все большее количество округов Северной Каролины.

## 1.3.2 Spark обеспечивает быструю передачу данных для Lumeris

Lumeris – это компания информационного обеспечения служб здравоохранения, расположенная в Сент-Луисе, штат Миссури. Компания постоянно помогает органам здравоохранения получить больше полезной информации из имеющихся данных. Для высокотехнологичной ИТ-системы компании потребовалось существенное ускорение, чтобы обслуживать больше клиентов и обеспечить более мощные средства внутренней обработки имеющихся данных.

В компании Lumeris Apache Spark, как часть процессов инженерии данных, потребляет тысячи файлов в формате CSV (comma-separated values – значения, разделяемые запятыми), хранящихся на сайте Amazon Simple Storage Service (S3), создает ресурсы по стандарту здравоохранения HL7 FHIR[1] и сохраняет их в специализированном хранилище документов, где они могут использоваться как существующими приложениями, так и новым поколением клиентских приложений.

Такой стек технологий позволяет компании Lumeris продолжать свой рост и развитие в плане объемов обрабатываемых данных и приложений. В дальнейшем с помощью этой технологии компания Lumeris надеется обеспечить сохранность многих жизней.

---

[1] Health Level Seven International (HL7) – это некоммерческие, утвержденные ANSI стандарты для организаций, занимающихся обменом, сбором и объединением, совместным использованием и извлечением информации в области здравоохранения, представленной в электронном виде. Стандарт HL7 поддерживают более 1600 участников из более чем 50 стран. Fast Healthcare Interoperability Resources (FHIR) – одна из самых последних спецификаций стандартов по обмену информацией в области здравоохранения.

### 1.3.3 Spark анализирует журналы наблюдения за оборудованием CERN

CERN, или Европейская организация ядерных исследований, была основана в 1954 году. Здесь находится Большой адронный коллайдер (LHC – Large Hadron Collider), 27-километровое кольцо, расположенное на глубине 100 м под землей на границе Франции и Швейцарии, в Женеве.

Здесь проводятся крупномасштабные физические эксперименты, генерирующие 1 Петабайт (Пб) данных в секунду. После значительной фильтрации объем данных сокращается до 900 Гб в день.

После экспериментов с Oracle, Impala и Spark группа CERN на основе Spark создала сервис Next CERN Accelerator Logging Service (NXCALS) в собственном локальном облаке под управлением OpenStack, в котором насчитывается до 250 000 ядер. Пользователями этой впечатляющей архитектуры являются ученые (использующие специализированные приложения и виртуальные блокнотные среды Jupyter), разработчики и различные приложения. Дальнейшая цель CERN – предоставление постоянного доступа к еще большему объему данных и увеличение общей скорости обработки данных.

### 1.3.4 Другие варианты использования

Spark применялся во многих других вариантах использования, включая следующие:

- создание интерактивных инструментальных средств так называемого выпаса данных (data-wrangling), таких как виртуальные блокнотные среды IBM Watson Studio и Databricks;
- наблюдение за качеством видео, передаваемого по телевизионным каналам, таким как MTV или Nickelodeon[1];
- наблюдение за участниками онлайновой видео-игры с целью выявления некорректного поведения и регулирования взаимодействия игроков в псевдореальном времени для достижения всеми игроками максимально возможного положительного опыта; компания Riot Games.

## 1.4 Почему вам очень понравится фрейм данных

Моя цель в этом разделе – сделать так, чтобы вам очень понравился фрейм данных. Вы узнаете только то, что стимулирует вас к дальнейшему изучению, и вы узнаете больше в главе 3 и последующих главах книги. Фрейм данных (dataframe) – это совокупность контейнера данных и прикладного интерфейса API.

Концепция фрейма данных чрезвычайно важна в Spark. Тем не менее эту концепцию понять не так уж сложно. Вы будете постоянно исполь-

[1] См. статью «How MTV and Nickelodeon Use Real-Time Big Data Analytics to Improve Customer Experience», автор Бернард Марр (Bernard Marr), Forbes, January 2017 (http://bit.ly/2ynJvUt).

зовать фреймы данных. В этом разделе объясняется, что такое фрейм данных с точки зрения Java (инженера по программному обеспечению) и с точки зрения СУРБД (инженера по обработке данных). После ознакомления с этими точками зрения с приведением некоторых аналогий в конце раздела будет приведена диаграмма.

> **Примечание. Вопрос правописания**
>
> В литературе вам в большинстве случаев будет встречаться другой вариант написания термина: DataFrame. Я решил придерживаться варианта, общепринятого в англоязычной литературе, который, соглашусь, может казаться непривычным для людей, говорящих по-французски. Как бы то ни было, несмотря на свое «королевское величие», фрейм данных – dataframe – остается обычным существительным, поэтому нет причин использовать то тут, то там буквы верхнего регистра. Здесь вам не какая-нибудь забегаловка!

## 1.4.1 Фрейм данных с точки зрения Java

Если вы хорошо знакомы с языком программирования Java и имеете некоторый опыт использования Java Database Connectivity (JDBC), то фрейм данных для вас будет выглядеть приблизительно так: `ResultSet`. Он содержит данные, у него есть API…

Фрейм данных и `ResultSet` обладают следующими одинаковыми свойствами:

- доступ к данным осуществляется через простой API;
- можно получить доступ к схеме.

Но есть и некоторые отличия фрейма данных от `ResultSet`:

- нет возможности последовательного просмотра с помощью метода `next()`;
- API расширяется с помощью функций, определенных пользователем (UDF). Можно написать свой код или обертку для существующего кода и добавить в Spark. Далее этот код будет доступным в распределенном режиме. Функции, определенные пользователем (UDF), рассматриваются в главе 16;
- если необходим доступ к данным, то сначала нужно получить строку `Row`, затем перемещаться по столбцам этой строки с помощью методов типа `get` (это похоже на `ResultSet`);
- метаданные достаточно просты, так как они не являются главными или внешними ключами или индексами в Spark.

В Java фрейм данных реализован как `Dataset<Row>` (произносится как «a dataset of rows» – набор данных, состоящий из строк).

## 1.4.2 Фрейм данных с точки зрения СУРБД

Если вы имеете опыт работы с СУРБД, то, возможно, считаете, что фрейм данных – это нечто похожее на таблицу. Фрейм данных и таблица действительно имеют следующие одинаковые свойства:

- данные размещены в столбцах и строках;
- столбцы строго типизированы.

Фрейм данных отличается от таблицы СУРБД следующими свойствами:

- данные могут быть вложенными, как в документе в формате JSON или XML. В главе 7 описано потребление таких документов, и вы будете использовать вложенные конструкции такого типа в главе 13;
- невозможно обновлять или удалять целые строки; вы создаете новый фрейм данных;
- можно с легкостью добавлять или удалять столбцы;
- в фрейме данных нет ограничений, индексов, главных и внешних ключей, триггеров.

## 1.4.3 Графическое представление фрейма данных

Фрейм данных – это мощный инструмент, которым вы будете пользоваться на протяжении всей книги и во время работы со Spark. Его весьма эффективный API и возможности хранения данных выделяют фрейм данных как главный элемент среди всего, что его окружает. На рис. 1.7 показан один из вариантов графического представления API, реализации и хранилища фрейма данных.

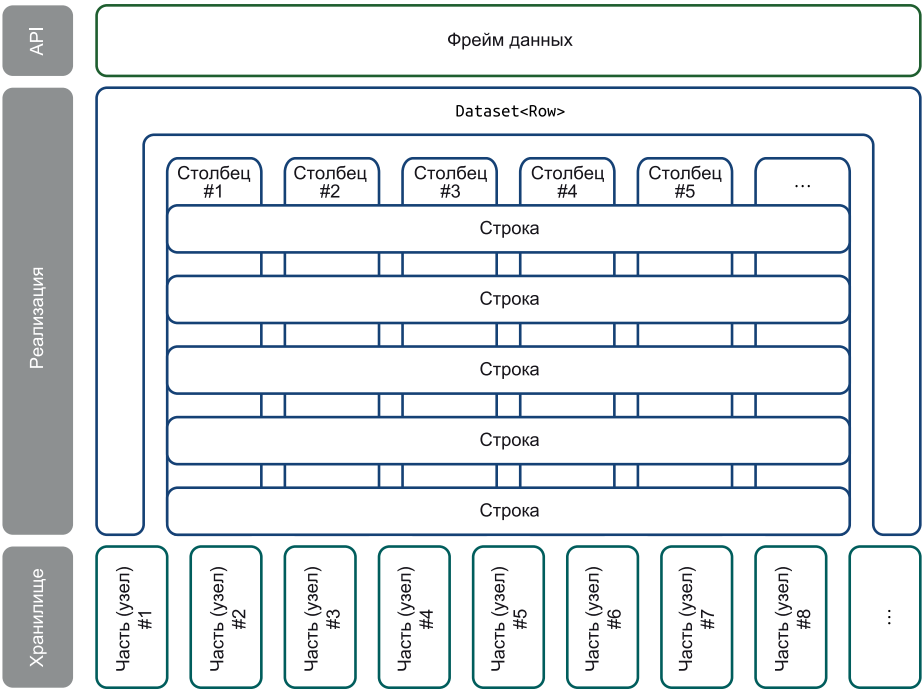


**Рис. 1.7 Графическое представление фрейма данных, его реализации на языке Java (Dataset<Row>), схемы и распределенного хранилища данных. Как разработчик вы будете использовать API набора данных, который позволит работать со столбцами и строками. К хранилищу, распределенному по разделам (узлам кластера), также можно получить доступ, в основном для оптимизации. О распределении по узлам кластера вы подробнее узнаете в главе 2**

## 1.5 Первый пример

Пора рассмотреть первый пример. Ваша задача – запустить Spark с простым приложением, которое считывает содержимое файла, сохраняет его в фрейме данных и выводит результат. Вы узнаете, как настраивать рабочую среду, которая будет использоваться на протяжении всей книги. Вы также научитесь взаимодействовать с рабочей средой Spark и выполнять основные операции.

В дальнейшем вы увидите, что в большинстве глав содержатся лабораторные работы по темам соответствующей главы, в которых можно экспериментировать с исходным кодом. Для каждой лабораторной работы предлагается набор данных (по возможности взятый из реальной практической деятельности), а также один или несколько листингов с исходным кодом.

Чтобы начать изучение примера, необходимо выполнить следующие действия:

- установить требуемое программное обеспечение, возможно, оно уже установлено в вашей системе: Git, Maven, Eclipse;
- загрузить исходный код методом клонирования из репозитория GitHub;
- выполнить пример, который загрузит простой файл в формате CSV (значения, разделенные запятыми) и выведет несколько строк.

### 1.5.1 Рекомендуемое программное обеспечение

В этом разделе представлен список программ, которые вы будете использовать на протяжении всей книги. Подробные инструкции по установке требуемого программного обеспечения содержатся в приложениях А и Б.

В этой книге используется следующее программное обеспечение:

- Apache Spark 3.0.0;
- в основном ОС macOS Catalina, но примеры работают и в Ubuntu версий с 14 по 18, и в Windows 10;
- Java 8 (хотя не используется множество конструкций, введенных в версии 8, таких как лямбда-функции). Мне известно, что уже доступна версия Java 11, но большинство организаций слишком медленно переходят на более новую версию (кроме того, мне кажется немного запутанной новейшая стратегия развития Java компанией Oracle). На данный момент только Spark v3 сертифицирован для работы с Java 11.

В примерах будет использоваться либо командная строка, либо интегрированная среда разработки Eclipse. Для работы в командной строке можно использовать следующее программное обеспечение:

- Maven – в книге используется версия 3.5.2, но любая более поздняя версия также будет работать;
- Git версии 2.13.6, но любая более поздняя версия также будет работать. В macOS можно воспользоваться версией, представленной

в пакете Xcode. В Windows можно скачать пакет Git с сайта https://git-scm.com/download/win. Если вы предпочитаете графические пользовательские интерфейсы (GUI) для работы с Git, то рекомендую программу Atlassian Sourcetree, которую можно загрузить с сайта www.sourcetreeapp.com.

## 1.5.2 Скачивание исходного кода

Исходный код размещен в открытом репозитории на сайте GitHub. URL репозитория https://github.com/jgperrin/net.jgp.books.spark.ch01. В приложении Г подробно описано, как использовать Git в командной строке и в Eclipse для загрузки исходного кода.

## 1.5.3 Запуск первого приложения

Теперь вы готовы к запуску самого первого приложения. Если при запуске возникли какие-то проблемы, то вам должно помочь приложение С.

### Командная строка

В командной строке необходимо перейти в рабочий каталог:

```
$ cd net.jgp.books.spark.ch01
```

Затем выполните следующую команду:

```
$ mvn clean install exec:exec
```

### Eclipse

После импорта проекта (см. приложение Г) поместите файл приложения *CsvToDataframeApp.java* в Project Explorer. Щелкните правой кнопкой мыши по имени этого файла, затем выберите из контекстного меню пункт **Run As > 2 Java Application**, как показано на рис. 1.8. Посмотрите на полученный результат в консоли.

Независимо от того, использовали вы командную строку или Eclipse, после нескольких секунд внутренней обработки вы должны увидеть результат, похожий на приведенный ниже:

```
+---+--------+--------------------+-----------+--------------------+
| id|authorId|               title|releaseDate|                link|
+---+--------+--------------------+-----------+--------------------+
|  1|       1|Fantastic Beasts ...|   11/18/16|http://amzn.to/2k...|
|  2|       1|Harry Potter and ...|    10/6/15|http://amzn.to/2l...|
|  3|       1|The Tales of Beed...|    12/4/08|http://amzn.to/2k...|
|  4|       1|Harry Potter and ...|    10/4/16|http://amzn.to/2k...|
|  5|       2|Informix 12.10 on...|    4/23/17|http://amzn.to/2i...|
+---+--------+--------------------+-----------+--------------------+
здесь показано только 5 верхних строк
```

Теперь попробуем разобраться, что здесь произошло.

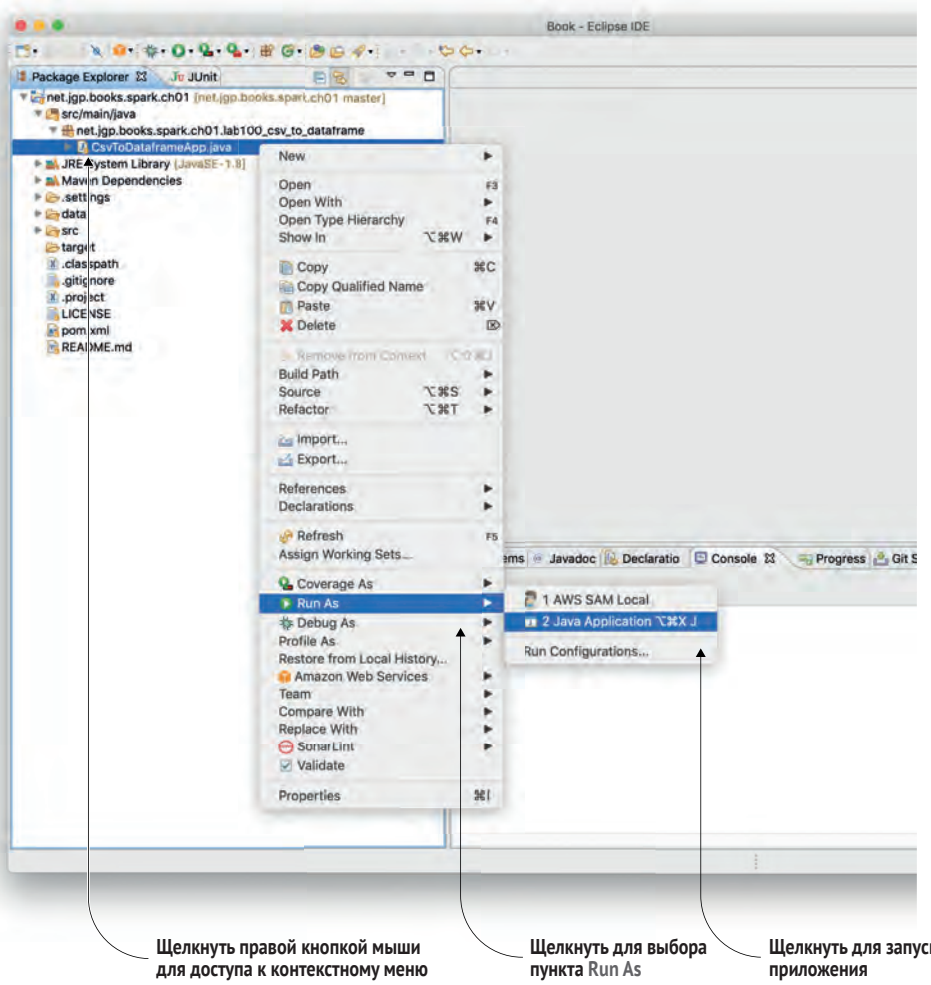


**Рис. 1.8 Интегрированная среда разработки Eclipse с деревом проекта в Project Explorer**

## 1.5.4 Первый исходный код для вас

Наконец-то вы добрались до кода! В предыдущем разделе вы видели вывод результата. Пора запустить первое приложение, которое создает сеанс, обращается к Spark с запросом на загрузку файла в формате CSV, а затем выводит пять строк (не более) полученного набора данных. В листинге 1.1 приведен полный код программы.

Когда дело доходит до представления исходного кода, следует вспомнить о существовании двух направлений мышления: одно направление придерживается абстрактного подхода в представлении кода, второе направление предпочитает представлять весь исходный код полностью, как есть. Я сторонник второго направления: мне нравится полный, за-

вершенный код примеров больше, чем частичный. Мне не хотелось бы, чтобы вам пришлось самостоятельно домысливать отсутствующую часть кода или требуемые программные пакеты, даже если все это вполне очевидно.

**Листинг 1.1 Потребление данных в формате CSV**

```
package net.jgp.books.spark.ch01.lab100_csv_to_dataframe;

import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

public class CsvToDataframeApp {
    public static void main(String[] args) {                          ❶
        CsvToDataframeApp app = new CsvToDataframeApp();
        app.start();
    }

    private void start() {
        SparkSession spark = SparkSession.builder()                   ❷
                .appName("CSV to Dataset")
                .master("local")
                .getOrCreate();

        Dataset<Row> df = spark.read().format("csv")                  ❸
                .option("header", "true")
                .load("data/books.csv");

        df.show(5);                                                   ❹
    }
}
```

❶ Функция `main()` – точка входа в приложение.
❷ Создание сеанса на локальном ведущем узле.
❸ Считывание файла *books.csv* в формате CSV и сохранение его содержимого в фрейме данных.
❹ Вывод не более пяти строк из созданного фрейма данных.

Хотя это простой пример, вы все же выполнили следующие действия:

- установили все компоненты, необходимые для работы со Spark (да, это так просто);
- создали сеанс, в котором может выполняться программный код;
- загрузили файл с данными в формате CSV;
- вывели пять строк из полученного набора данных.

Теперь вы готовы к более глубокому изучению Apache Spark и пониманию того, что происходит внутри этой операционной системы.

## Резюме

- Spark – это аналитическая операционная система, которую можно использовать для обработки рабочей нагрузки и реализации алгоритмов в распределенном режиме. Spark пригоден не только для анализа:

можно использовать его для передачи данных, для крупномасштабных преобразований данных, для анализа журналов и для решения многих других задач.

- Spark поддерживает SQL, Java, Scala, R и Python как интерфейсы программирования, но в этой книге рассматривается только Java (иногда Python).
- Основным внутренним хранилищем данных Spark является фрейм данных. Фрейм данных объединяет возможность хранения данных с прикладным интерфейсом API.
- Если у вас есть опыт разработки ПО с использованием JDBC, то вы обнаружите некоторые свойства, похожие на структуру JDBC `ResultSet`.
- Если у вас есть опыт разработки ПО для реляционных баз данных, то вы можете сравнить фрейм данных с таблицей с менее сложными метаданными.
- В Java фрейм данных реализован как `Dataset<Row>`.
- Можно быстро настроить Spark с помощью Maven и Eclipse. Spark не требует специализированной процедуры установки.
- Spark не ограничен применением алгоритма MapReduce: его API позволяет применять множество разнообразных алгоритмов для обработки данных.
- Потоковая обработка (streaming) все чаще и чаще используется в корпоративных приложениях, так как в деловой сфере требуется доступ к методам анализа в реальном времени. Spark поддерживает потоковую обработку данных.
- Анализ данных уже вышел за рамки простых объединений и агрегаций. В производственной сфере необходимо, чтобы «компьютеры думали за нас». Поэтому Spark поддерживает методы машинного обучения и глубокого обучения.
- Графы – специализированная методика применения анализа, но Spark поддерживает и графы.