Содержание

Предисловие

За очень короткое время после появления Apache Spark – фреймворк следующего поколения для быстрой обработки больших объемов данных – получил повсеместное распространение. Spark превосходит фреймворк Hadoop MapReduce, который дал импульс революции в обработке больших объемов данных, по множеству ключевых параметров: он намного быстрее, намного проще в использовании благодаря богатому API и может применяться для создания не только приложений пакетной обработки данных разной мощности, но и интерактивных приложений, приложений потоковой обработки данных, машинного обучения и обработки графов.

Я был тесно вовлечен в разработку Spark на всех этапах этого процесса, от чертежной доски до образования самого активного из современных открытых проектов и одного из самых активных проектов Apache! Мне было особенно приятно, что Матей Захария (Matei Zaharia), создатель Spark, объединился с другими давнишними разработчиками Spark – Патриком Венделлом (Patrick Wendell), Энди Конвински (Andy Konwinski) и Холденом Карау (Holden Karau), – чтобы написать эту книгу.

В связи с быстрым ростом популярности Spark на передний план вышла проблема нехватки хороших справочных руководств. Авторы книги проделали длинный путь для ее решения, написав 11 глав и представив десятки подробных примеров, чтобы помочь специалистам в области анализа данных, студентам и программистам поближе узнать Spark. Она доступна читателям, не имеющим опыта работы с «большими данными», что делает ее отличной отправной точкой для начала изучения предметной области в целом. Я надеюсь, что много лет спустя читатели со светлым чувством будут вспоминать эту книгу как проводника в новый захватывающий мир.

– *Ион Стоика* (*Ion Stoica*),
директор Databricks и содиректор AMPlab,
Калифорнийский университет Беркли

Вступление

По мере вхождения в обиход анализа данных специалисты-практики во многих областях искали все более простые инструменты для решения этой задачи. Apache Spark быстро завоевал популярность как инструмент, расширяющий и обобщающий модель MapReduce. Фреймворк Spark имеет три основных преимущества. Во-первых, простота в использовании – с его помощью можно создавать приложения на ноутбуке, используя высокоуровневый API, который позволяет сконцентрироваться на предметной стороне вычислений. Во-вторых, высокая скорость работы, что дает возможность создавать интерактивные приложения и использовать сложные алгоритмы. И в-третьих, обобщенность, позволяющая объединять разнотипные вычисления (например, выполнять SQL-запросы, обрабатывать текст и реализовывать алгоритмы машинного обучения (machine learning)), для чего прежде необходимо было применять разрозненные инструменты. Все это делает Spark отличной отправной точкой на пути изучения аспектов обработки «больших данных» (Big Data).

Цель этого вводного руководства – помочь вам быстро настроить Spark и приступить к работе с ним. Здесь вы узнаете, как загрузить и запустить Spark на своем ноутбуке, как работать с ним в интерактивном режиме, чтобы поближе познакомиться с API. Затем мы рассмотрим особенности доступных операций и распределенных вычислений. В заключение мы совершим экскурс по высокоуровневым библиотекам, входящим в состав Spark, включая библиотеки для машинного обучения, потоковой обработки данных (stream processing) и SQL. Мы надеемся, что с этой книгой вы быстро сможете приступить к решению задач, связанных с анализом данных, как на одной, так и на сотнях машин.

Кому адресована эта книга

Данная книга адресована специалистам в области анализа данных (или исследователям) и инженерам-программистам. Мы выбрали эти две группы, потому что они смогут извлечь наибольшую выгоду от привлечения фреймворка Spark для решения своих задач. Богатая коллекция библиотек (таких как MLlib), входящих в состав Spark, поможет специалистам в области анализа данных решать статистические задачи, непосильные единственному компьютеру. Программис-

ты, в свою очередь, узнают, как писать распределенные программы на основе Spark и как управлять промышленными приложениями. Программисты и исследователи по-разному будут воспринимать эту книгу, но и те, и другие смогут задействовать Spark для решения больших распределенных задач в своих областях.

Исследователи основное внимание уделяют ответам на вопросы и разработке моделей на основе данных. Они часто имеют математическую подготовку, и некоторые из них знакомы с такими инструментами, как Python, R и SQL. Мы постарались включить в книгу примеры программного кода на Python и, где это необходимо, на SQL, а также обзор библиотек и особенностей поддержки машинного обучения в Spark. Если вы – исследователь, специалист в области анализа данных, мы надеемся, что после прочтения нашей книги вы сможете использовать те же математические подходы для решения задач, только намного быстрее и в более широком масштабе.

Вторая целевая группа данной книги – инженеры-программисты, имеющие некоторый опыт программирования на Java, Python или других языках. Если вы – программист, мы надеемся, что благодаря этой книге вы научитесь настраивать кластеры Spark, пользоваться командной оболочкой Spark и писать Spark-приложения для организации параллельных вычислений. Знакомые с фреймворком Hadoop уже знают, как взаимодействовать с HDFS и управлять кластерами, но, как бы то ни было, мы все равно опишем основные понятия распределенных вычислений.

Кем бы вы ни были, исследователем или программистом, чтобы извлечь максимум из этой книги, необходимо иметь знакомство с любым из языков программирования: Python, Java, Scala или им подобным. Мы полагаем, что у вас уже реализовано решение хранилища для ваших данных, поэтому мы охватим лишь наиболее общие подходы к загрузке и сохранению данных, но не будем обсуждать вопросы их реализации. Если у вас пока нет опыта использования ни одного из перечисленных языков, не волнуйтесь: существуют великолепные книги, которые помогут в овладении ими. Некоторые из таких книг мы упомянем в разделе «Книги поддержки» ниже.

Как организована эта книга

Главы в этой книге следуют в порядке изучения материала. В начале каждой главы мы будем сообщать, какие ее разделы, по нашему мнению, больше подходят для исследователей, а какие – для программис-

тов. При этом мы надеемся, что все разделы будут доступны читателям с любым уровнем подготовки.

Первые две главы описывают порядок установки на ноутбук фреймворка Spark в базовой конфигурации и демонстрируют, чего можно достичь с его помощью. После установки и знакомства с некоторыми возможностями мы погрузимся в командную оболочку Spark – инструмент, очень удобный для разработки и прототипирования. В последующих главах подробно обсуждаются программный интерфейс Spark, порядок выполнения приложений в кластерах и высокоуровневые библиотеки, доступные в Spark (такие как Spark SQL и MLlib).

Книги поддержки

Исследователям, не имеющим опыта использования Python, отличным введением в этот язык программирования могут послужить книги «Learning Python»[1] и «Head First Python» (обе выпущены издательством O'Reilly). Имеющим некоторый опыт программирования на Python, но желающим изучить его глубже можно порекомендовать книгу «Dive into Python» (Apress).

Инженерам-программистам, а также всем, кто прочтет эту книгу, для расширения познаний в области обработки данных мы рекомендуем книги «Machine Learning for Hackers» и «Doing Data Science» (обе выпущены издательством O'Reilly).

Эта книга написана языком, доступным для начинающих. В дальнейшем мы предполагаем написать более подробную книгу для тех, кто пожелает глубже вникнуть во внутреннее устройство Spark.

Типографские соглашения

В этой книге приняты следующие типографские соглашения:

Курсив

Используется для обозначения новых терминов, адресов электронной почты, имен файлов и расширений имен файлов.

`Моноширинный шрифт`

Применяется для оформления листингов программ и программных элементов внутри обычного текста, таких как имена перемен-

[1] Лутц М. Изучаем Python. 4-е изд. М.: Символ-Плюс, 2010. ISBN: 978-5-93286-159-2. – *Прим. перев.*

ных и функций, типов данных, переменных окружения, инструкций и ключевых слов.

Моноширинный жирный

Обозначает команды или другой текст, который должен вводиться пользователем.

Моноширинный курсив

Обозначает текст, который должен замещаться фактическими значениями, вводимыми пользователем или определяемыми из контекста.

Так обозначаются советы, предложения и примечания общего характера.

Так обозначаются предупреждения и предостережения.

Использование программного кода примеров

Все примеры программного кода, что приводятся в этой книге, доступны в репозитории GitHub. Их можно получить по адресу: https://github.com/databricks/learning-spark. Примеры кода написаны на языках Java, Scala и Python.

Примеры на языке Java написаны для выполнения под управлением Java 6 и выше. В Java 8 появилась поддержка лямбда-выражений, облегчающих создание встраиваемых (inline) функций, благодаря чему код, использующий фреймворк Spark, получается намного более простым. Мы решили не использовать этот синтаксис в основных примерах, поскольку версия Java 8 пока не получила широкого распространения. Если вам интересно будет попробовать синтаксис Java 8, прочитайте статью в блоге Databricks[1]. Некоторые из примеров мы переписали на Java 8 и сохранили в репозитории GitHub.

Данная книга призвана оказать вам помощь в решении ваших задач. Вы можете свободно использовать примеры программного кода из этой книги в своих приложениях и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части программного кода. Например, если вы разрабатываете программу и используете в ней несколько отрывков программного кода из книги, вам не нужно обращаться за

[1] http://bit.ly/1ywZBs4.

разрешением. Однако в случае продажи или распространения компакт-дисков с примерами из этой книги вам необходимо получить разрешение от издательства O'Reilly. Если вы отвечаете на вопросы, цитируя данную книгу или примеры из нее, получение разрешения не требуется. Но при включении существенных объемов программного кода примеров из этой книги в вашу документацию необходимо получить разрешение издательства.

Мы приветствуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN. Например: «Learning Spark by Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia (O'Reilly). Copyright 2015 Databricks, 978-1-449-35862-4».

За получением разрешения на использование значительных объемов программного кода примеров из этой книги обращайтесь по адресу permissions@oreilly.com.

Safari® Books Online

Safari Books Online (http://www.safaribooksonline.com) – это виртуальная библиотека, содержащая авторитетную информацию[1] в виде книг и видеоматериалов, созданных ведущими специалистами в области технологий и бизнеса.

Профессионалы в области технологии, разработчики программного обеспечения, веб-дизайнеры, а также бизнесмены и творческие работники используют Safari Books Online как основной источник информации для проведения исследований, решения проблем, обучения и подготовки к сертификационным испытаниям.

Библиотека Safari Books Online предлагает широкий выбор продуктов и тарифов[2] для организаций[3], правительственных[4] и учебных[5] учреждений, а также физических лиц.

Подписчики имеют доступ к поисковой базе данных, содержащей информацию о тысячах книг, видеоматериалов и рукописей от таких издателей, как O'Reilly Media, Prentice Hall Professional, Addison-Wesly Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann,

---

1 https://www.safaribooksonline.com/explore/.
2 https://www.safaribooksonline.com/pricing/.
3 https://www.safaribooksonline.com/enterprise/.
4 https://www.safaribooksonline.com/government/.
5 https://www.safaribooksonline.com/academic-public-library/.

IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, и десятков других[1]. За подробной информацией о Safari Books Online обращайтесь по адресу: http://www.safaribooksonline.com/.

Как с нами связаться

С вопросами и предложениями, касающимися этой книги, обращайтесь в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в Соединенных Штатах Америки или в Канаде)
707-829-0515 (международный)
707-829-0104 (факс)

Список опечаток, файлы с примерами и другую дополнительную информацию вы найдете на странице книги http://bit.ly/learning-spark.

Свои пожелания и вопросы технического характера отправляйте по адресу: bookquestions@oreilly.com.

Дополнительную информацию о книгах, обсуждения, Центр ресурсов издательства O'Reilly вы найдете на сайте: http://www.oreilly.com.

Ищите нас в Facebook: http://facebook.com/oreilly.

Следуйте за нами в Твиттере: http://twitter.com/oreillymedia.

Смотрите нас на YouTube: http://www.youtube.com/oreillymedia.

Благодарности

Авторы выражают благодарность обозревателям за отзывы об этой книге: Джозефу Брэдли (Joseph Bradley), Дэйву Бриджланду (Dave Bridgeland), Чейзу Чандлеру (Chaz Chandler), Майку Дэвису (Mick Davies), Сэму ДеХорити (Sam DeHority), Виду Ха (Vida Ha), Эндрю Галу (Andrew Gal), Майклу Грегсону (Michael Gregson), Яну Иойпену (Jan Joeppen), Стефану Йоу (Stephan Jou), Джеффу Мартинесу (Jeff Martinez), Джошу Махонину (Josh Mahonin), Эндрю Ор (Andrew Or), Майку Паттерсону (Mike Patterson), Джошу Розену (Josh Rosen), Брюсу Шалвински (Bruce Szalwinski), Сянгруй Менгу (Xiangrui Meng) и Резу Заде (Reza Zadeh).

[1] https://www.safaribooksonline.com/our-library/.

Авторы выражают особую благодарность Дэвиду Анджиевски (David Andrzejewski), Дэвиду Баттлеру (David Buttler), Джульете Хогланд (Juliet Hougland), Мареку Колоджей (Marek Kolodziej), Таке Шинагаве (Taka Shinagawa), Деборе Сигель (Deborah Siegel), доктору Нормену Мюллеру (Dr. Normen Müller), Али Годши (Ali Ghodsi) и Самиру Фаруки (Sameer Farooqui). Они представили подробные отзывы для большинства глав и предложили множество существенных улучшений.

Мы также хотели бы поблагодарить экспертов, уделивших время редактированию и созданию отдельных частей глав. Татхагата Дас (Tathagata Das) работал с нами над созданием главы 10. Татхагата пошел дальше простого описания примеров, ответил на множество вопросов, а также внес большое число правок в текст. Майкл Армбруст (Michael Armbrust) помог нам проверить главу «Spark SQL». Джозеф Брэдли (Joseph Bradley) представил вводный пример для MLlib в главе 11. Реза Заде (Reza Zadeh) написал текст и примеры кода для сокращения размерности. Сянгруй Менг (Xiangrui Meng), Джозеф Брэдли (Joseph Bradley) и Реза Заде (Reza Zadeh) выполнили редактирование и рецензирование главы с описанием библиотеки MLlib.

Глава 1

Введение в анализ данных с помощью Spark

В этой главе приводится обобщенный обзор Apache Spark. Если вы уже знакомы с этим фреймворком и его компонентами, можете сразу перейти к главе 2.

Что такое Apache Spark?

Apache Spark – это *универсальная* и *высокопроизводительная* кластерная вычислительная платформа.

По производительности Spark превосходит популярные реализации модели MapReduce, попутно обеспечивая поддержку более широкого диапазона типов вычислений, включая интерактивные запросы и потоковую обработку (streaming processing). Скорость играет важную роль при обработке больших объемов данных, так как именно скорость позволяет работать в интерактивном режиме, не тратя минуты или часы на ожидание. Одно из важнейших достоинств Spark, обеспечивающих столь высокую скорость, – способность выполнять вычисления в памяти. Но даже при работе с дисковой памятью Spark выполняет операции намного эффективнее, чем известные механизмы MapReduce.

Фреймворк создавался с целью охватить как можно более широкий диапазон рабочих нагрузок, которые прежде требовали создания отдельных распределенных систем, включая приложения пакетной обработки, циклические алгоритмы, интерактивные запросы и потоковую обработку. Поддерживая все эти виды задач с помощью единого механизма, Spark упрощает и удешевляет *объединение* разных видов обработки, которые часто необходимо выполнять в едином конвейере обработки данных. Кроме того, он уменьшает бремя обслуживания, поддерживая отдельные инструменты.

Фреймворк Spark предлагает простой API на языках Python, Java, Scala и SQL и богатую коллекцию встроенных библиотек. Он также легко объединяется с другими инструментами обработки больших данных. В частности, Spark может выполняться под управлением кластеров Hadoop и использовать любые источники данных Hadoop, включая Cassandra.

Унифицированный стек

Проект Spark включает множество тесно связанных компонентов. Ядро фреймворка образует его «вычислительный механизм» (computational engine), отвечающий за планирование, распределение и мониторинг приложений, выполняющих множество вычислительных задач на множестве машин – *вычислительном кластере*. Быстрое и универсальное вычислительное ядро Spark приводит в действие разнообразные высокоуровневые компоненты, специализированные для решения разных задач, таких как выполнение запросов SQL или машинное обучение. Эти компоненты поддерживают тесную интеграцию друг с другом, давая возможность объединять их, подобно библиотекам в программном проекте.

Философия тесной интеграции имеет несколько преимуществ. Во-первых, все библиотеки и высокоуровневые компоненты стека извлекают определенные выгоды от улучшений в слоях более низкого уровня. Например, когда в ядро Spark вносятся какие-то оптимизации, библиотеки поддержки SQL и машинного обучения автоматически увеличивают производительность. Во-вторых, затраты на сопровождение стека минимальны, потому что вместо 5–10 независимых программных систем организации требуется поддерживать только одну. Эти затраты включают развертывание, сопровождение, тестирование, поддержку и т. д. Это также означает, что при добавлении в стек Spark новых компонентов все организации, использующие Spark, немедленно получают возможность опробовать эти новые компоненты. Это уменьшает затраты на опробование новых видов анализа данных, избавляя организации от необходимости загружать, развертывать и изучать новые программные проекты.

Наконец, одним из самых больших преимуществ тесной интеграции является возможность создавать приложения, прозрачно объединяющие разные модели обработки. Например, используя Spark, можно написать приложение, применяющее модель машинного обучения для классификации данных в масштабе реального времени и потребляю-

щее потоковые данные. Одновременно аналитики имеют возможность запрашивать результаты, также в масштабе реального времени, посредством SQL (например, объединяя данные с неструктурированными файлами журналов). Кроме того, опытные программисты и исследователи могут обращаться к тем же данным посредством командной оболочки на языке Python и выполнять дополнительные виды анализа. Другие могут пользоваться результатами, получаемыми от автономных приложений пакетной обработки. При этом отделу ИТ приходится обслуживать единственную систему.

Ниже мы коротко представим все основные компоненты Spark, изображенные на рис. 1.1.

Рис. 1.1 ❖ Стек Spark

Spark Core

Spark Core реализует основные функциональные возможности фреймворка Spark, включая компоненты, осуществляющие планирование заданий, управление памятью, обработку ошибок, взаимодействие с системами хранения данных и многие другие. Spark Core является также основой API *устойчивых распределенных наборов данных* (*Resilient Distributed Datasets*, *RDD*) – базовой абстракции Spark. Наборы данных RDD представляют собой коллекции элементов, распределенных между множеством вычислительных узлов, которые могут обрабатываться параллельно. Spark Core предоставляет множество функций управления такими коллекциями.

Spark SQL

Spark SQL – пакет для работы со структурированными данными. Позволяет извлекать данные с помощью инструкций на языке SQL и его диалекте Hive Query Language (HQL). Поддерживает множество ис-

точников данных, включая таблицы Hive, Parquet и JSON. В дополнение к интерфейсу SQL компонент Spark SQL позволяет смешивать в одном приложении запросы SQL с программными конструкциями на Python, Java и Scala, поддерживаемыми абстракцией RDD, и таким способом комбинировать SQL со сложной аналитикой. Подобная тесная интеграция с богатыми возможностями вычислительной среды выгодно отличает Spark SQL от любых других инструментов управления данными. Spark SQL был добавлен в стек Spark в версии 1.0.

Первой реализацией поддержки SQL в Spark стал проект Shark, созданный в Калифорнийском университете, Беркли. Этот проект представлял собой модификацию Apache Hive, способную выполняться под управлением Spark. Позднее ему на смену пришел компонент Spark SQL, имеющий более тесную интеграцию с механизмом Spark и API для разных языков программирования.

Spark Streaming

Spark Streaming – компонент Spark для обработки потоковых данных. Примерами источников таких данных могут служить файлы журналов, заполняемые действующими веб-серверами, или очереди сообщений, посылаемых пользователями веб-служб. Spark Streaming имеет API для управления потоками данных, который близко соответствует модели RDD, поддерживаемой компонентом Spark Core, что облегчает изучение самого проекта и разных приложений обработки данных, хранящихся в памяти, на диске или поступающих в режиме реального времени. Прикладной интерфейс (API) компонента Spark Streaming разрабатывался с прицелом обеспечить такую же надежность, пропускную способность и масштабируемость, что и Spark Core.

MLlib

В состав Spark входит библиотека MLlib, реализующая механизм машинного обучения (Machine Learning, ML). MLlib поддерживает множество алгоритмов машинного обучения, включая алгоритмы классификации (classification), регрессии (regression), кластеризации (clustering) и совместной фильтрации (collaborative filtering), а также функции тестирования моделей и импортирования данных. Она также предоставляет некоторые низкоуровневые примитивы ML, включая универсальную реализацию алгоритма оптимизации методом градиентного спуска. Все эти методы способны работать в масштабе кластера.

GraphX

GraphX – библиотека для обработки графов (примером графа может служить граф друзей в социальных сетях) и выполнения параллельных вычислений. Подобно компонентам Spark Streaming и Spark SQL, GraphX дополняет Spark RDD API возможностью создания ориентированных графов с произвольными свойствами, присваиваемыми каждой вершине или ребру. Также GraphX поддерживает разнообразные операции управления графами (такие как `subgraph` и `mapVertices`) и библиотеку обобщенных алгоритмов работы с графами (таких как алгоритмы ссылочного ранжирования PageRank и подсчета треугольников).

Диспетчеры кластеров

Внутренняя реализация Spark обеспечивает эффективное масштабирование от одного до многих тысяч вычислительных узлов. Для достижения такой гибкости Spark поддерживает большое многообразие *диспетчеров кластеров* (*cluster managers*), включая Hadoop YARN, Apache Mesos, а также простой диспетчер кластера, входящий в состав Spark, который называется Standalone Scheduler. При установке Spark на чистое множество машин на начальном этапе с успехом можно использовать Standalone Scheduler. При установке Spark на уже имеющийся кластер Hadoop YARN или Mesos можно пользоваться встроенными диспетчерами этих кластеров. Подробнее о разных диспетчерах кластеров и их использовании рассказывается в главе 7.

Кто и с какой целью использует Spark?

Так как Spark относится к категории универсальных фреймворков поддержки вычислений в кластерах, он применяется для реализации широкого круга приложений. Во вступлении мы определили две группы читателей, на которых ориентирована наша книга: специалисты в области анализа данных и инженеры-программисты. Давайте теперь поближе познакомимся с обеими группами и с тем, как они используют Spark. Неудивительно, что специалисты в этих двух группах используют Spark по-разному, но мы можем примерно разбить случаи использования на две основные категории – *исследование данных* и *обработка данных*.

Разумеется, это весьма условное разделение, и многие профессионалы обладают обоими навыками, иногда выступая в роли исследо-

вателей данных, а иногда создавая приложения обработки данных. Тем не менее имеет смысл в отдельности рассмотреть эти две группы и соответствующие им случаи использования.

Исследование данных

Наука о данных (data science) – относительно новая дисциплина, появившаяся несколько лет тому назад и специализирующаяся на анализе данных. Несмотря на отсутствие точного определения, мы будем пользоваться термином *специалист в области анализа данных*, или *исследователь*, для обозначения людей, основной задачей которых являются анализ и моделирование данных. Специалисты в области анализа данных могут иметь опыт использования SQL, статистических методов, прогнозирования (машинного обучения) и программирования, как правило, на Python, Matlab или R. Они также владеют приемами преобразования данных в форматы, в которых они могут быть проанализированы для проникновения в их суть (иногда это называют *выпасом данных* – *data wrangling*).

Исследователи используют свои навыки для анализа данных с целью ответить на определенные вопросы или вскрыть их суть. Часто они прибегают к специализированным методам анализа, для чего используют интерактивные оболочки (вместо создания сложных приложений), позволяющие им получать результаты запросов в кратчайшие сроки. Благодаря быстродействию и простоте API фреймворк Spark прекрасно подходит для этой цели, а его встроенные библиотеки предоставляют множество готовых алгоритмов.

Благодаря большому числу компонентов Spark поддерживает большое многообразие видов анализа данных. Командная оболочка Spark упрощает проведение анализа в интерактивном режиме, с применением Python или Scala. Spark SQL также имеет отдельную интерактивную оболочку SQL, которую можно использовать для исследования данных. Компонент Spark SQL можно также использовать в обычных программах на основе Spark или из командной оболочки Spark. Технологии машинного обучения и анализа данных поддерживаются также библиотеками MLlib. Кроме того, имеется поддержка внешних программ Matlab или на языке R. Spark позволяет исследователям данных заниматься задачами, основанными на обработке огромных объемов данных, которые прежде были недоступны при использовании простых инструментов, таких как R или Pandas.

Иногда, вслед за начальным этапом исследования данных, исследователю необходимо оформить анализ в виде законченного продук-

та, то есть создать надежное приложение, позволяющее выполнять данный анализ и способное стать частью промышленного приложения. Например, начальные исследования данных могли бы привести исследователя к мысли о необходимости создания рекомендательной системы (recommender system), интегрированной в веб-приложение и генерирующей предложения для пользователей. Нередко созданием такого законченного продукта занимается другой человек – инженер-программист.

Обработка данных

Еще один основной случай использования фреймворка Spark можно описать в контексте работы инженера-программиста. В данном случае под инженерами-программистами мы подразумеваем разработчиков программного обеспечения, использующих Spark для создания приложений обработки данных. Обычно эти разработчики знакомы с принципами создания программ, такими как инкапсуляция, дизайн интерфейса и объектно-ориентированное программирование. Они часто имеют специальное образование и используют свои знания и навыки для проектирования и создания программных систем, реализующих бизнес-логику.

Программистам Spark предоставляет простой способ распараллеливания создаваемых ими приложений в рамках кластера и скрывает сложность программирования распределенных систем, сетевых взаимодействий и устойчивости к ошибкам. Система дает им достаточно высокий уровень контроля для организации мониторинга и настройки приложений, а также быстрого создания реализаций типичных задач. Модульная природа API (на основе передачи распределенных коллекций объектов) упрощает создание библиотек многократного использования и их тестирование на локальном компьютере.

Пользователи часто выбирают фреймворк Spark в качестве основы для своих приложений обработки данных, потому что он предоставляет широкое разнообразие функциональных возможностей, простых в изучении и применении, а также благодаря его зрелости и надежности.

Краткая история развития Spark

Spark – это проект с открытым исходным кодом, поддерживаемый многочисленным сообществом разработчиков. Если вы или ваша организация пробует применить Spark впервые, вам может быть ин-

тересно узнать немного об истории этого проекта. Проект Spark начинался в 2009 году как исследовательский, в лаборатории систем быстрой разработки приложений RAD Lab Калифорнийского университета (Беркли), позднее переименованной в AMPLab. Прежде сотрудники лаборатории использовали Hadoop MapReduce и пришли к выводу, что модель MapReduce неэффективна для реализации итеративных и интерактивных вычислительных задач. Поэтому с самого начала фреймворк Spark проектировался с прицелом на достижение максимальной производительности в интерактивном режиме и при выполнении итеративных алгоритмов, что, в свою очередь, повлекло реализацию идей хранения данных в памяти и эффективной обработки ошибок.

Вскоре после начала работы над проектом в 2009 году в академических кругах появились первые статьи о Spark. Уже тогда фреймворк показывал 10–20-кратное превосходство в скорости над MapReduce на некоторых задачах.

В числе первых пользователей Spark были только лаборатории Калифорнийского университета. К их числу, например, относятся исследователи в области машинного обучения из проекта Mobile Millennium – они использовали Spark для мониторинга и прогнозирования пробок на дорогах Сан-Франциско. Однако в очень короткое время Spark стали использовать другие организации, и на сегодняшний день более 50 организаций указывают себя на странице Spark PoweredBy[1] и десятки других заявляют об использовании Spark в списках сообществ, таких как Spark Meetups[2] и Spark Summit[3]. Помимо Калифорнийского университета, в разработке Spark участвуют также Databricks, Yahoo! и Intel.

В 2011 году лаборатория AMPLab приступила к разработке высокоуровневых компонентов для Spark, таких как Shark (Hive on Spark)[4] и Spark Streaming. Эти и другие компоненты иногда называют «Стек анализа данных из Беркли» (Berkeley Data Analytics Stack, BDAS)[5].

Исходный код Spark был открыт в марте 2010 года и в июне 2013-го передан в фонд Apache Software Foundation, где продолжает развиваться и по сей день.

---

[1] http://bit.ly/1yx195p.
[2] http://www.meetup.com/spark-users/.
[3] http://spark-summit.org/.
[4] Позднее проект Shark заменил проект Spark SQL.
[5] https://amplab.cs.berkeley.edu/software/.

Версии Spark

С момента создания проект Spark активно развивается сообществом, которое продолжает разрастаться с каждым выпуском. В создании версии Spark 1.0 участвовало более 100 отдельных разработчиков. Несмотря на рост активности, сообщество продолжает выпускать обновленные версии Spark на регулярной основе. Версия Spark 1.0 вышла в мае 2014-го. Эта книга в основном охватывает версии Spark 1.1.0 и выше, хотя большинство примеров будет работать и с более ранними версиями.

Механизмы хранения данных для Spark

Spark может создавать распределенные наборы данных из любых файлов, хранящихся в распределенной файловой системе Hadoop (HDFS) или в других системах хранения данных, поддерживающих Hadoop API (включая локальную файловую систему, Amazon S3, Cassandra, Hive, HBase и др.). Важно помнить, что Spark не требует наличия Hadoop; он просто поддерживает взаимодействие с системами хранения данных, реализующих Hadoop API. Spark поддерживает текстовые файлы, файлы SequenceFile, Avro, Parquet и любые другие форматы, поддерживаемые Hadoop. Приемы использования этих источников данных будут рассматриваться в главе 5.