1 Введение в эффективный Spark

В данной главе приведен обзор всего, что вы, надеемся, сможете узнать из нашей книги. Мы постараемся убедить вас выучить язык программирования Scala. Спокойно переходите сразу к главе 2, если знаете, что вам нужно, и уже используете Scala (или твердо решили писать на другом языке программирования).

Что такое Spark и почему производительность так важна

Фреймворк Apache Spark — высокопроизводительная универсальная распределенная система вычислений, самая активная часть проекта с открытым исходным кодом Apache более чем с 1000 участников[1]. Spark обеспечивает возможность обработки больших массивов данных, помимо тех, что могут уместиться на одной машине, с помощью высокоуровневого, относительно простого в использовании API. Spark — одна из самых быстрых систем среди аналогов, его архитектура и интерфейс уникальны. Это единственная система, которая позволяет описывать логику преобразований данных и алгоритмов машинного обучения так, чтобы не зависеть от системы, но сохранить возможность параллельного выполнения. Поэтому данный фреймворк зачастую используют для написания вычислений, которые будут работать быстро в распределенных системах хранения различных видов и размеров.

Однако, несмотря на множество преимуществ Spark и шумиху вокруг него, простейшие реализации многих распространенных стандартных операций науки о данных на Spark будут работать медленнее и менее надежно, чем оптимальные версии. А в силу того, что интересующие нас вычисления касаются обработки данных в огромном масштабе, выгоды от тонкой настройки производительности

[1] С сайта http://spark.apache.org/

кода могут быть колоссальными. Производительность — это не только быстрая работа, в подобном масштабе зачастую речь идет о том, чтобы вообще работать хоть как-то. Можно создать запрос Spark, который не будет выполняться при гигабайтных объемах данных, но после рефакторинга и внесения поправок в структуру конкретных данных и требования кластера прекрасно заработает в той же системе с терабайтами данных. Мы встречались в нашей практике написания промышленного кода Spark с тем, что одни и те же задачи на тех же кластерах работали в сотни раз быстрее после описанных здесь оптимизаций. Говоря языком обработки данных, время — деньги, и мы надеемся, эта книга окупит себя снижением стоимости информационной инфраструктуры и экономией времени разработчиков.

Далеко не все из этих методик применимы для каждого сценария использования. Spark является чрезвычайно гибким и более высокоуровневым, чем другие сравнимые по возможностям вычислительные фреймворки. Именно это позволяет извлечь колоссальную выгоду просто из более точной подгонки под форму и структуру данных. Некоторые из методов будут хорошо работать с определенными объемами данных или даже при определенных распределениях ключей, но не все. Простейший пример: во многих задачах использование функции `groupByKey` в Spark может с легкостью привести к ужасающим исключительным ситуациям из-за нехватки памяти. Но в отношении данных с незначительным дублированием эта операция будет работать столь же быстро, как и ее альтернативы, с которыми мы вас познакомим. Глубокое понимание конкретного сценария применения и системы, а также взаимодействие с ними фреймворка Spark совершенно необходимы для решения с его помощью наиболее сложных задач науки о данных.

Что даст эта книга

Мы надеемся, что, после того как вы прочтете книгу, ваши Spark-запросы станут быстрее, смогут обрабатывать бо́льшие объемы данных и будут потреблять меньше ресурсов. Издание охватывает широкий диапазон утилит и сценариев. Возможно, вы найдете здесь методики, неприменимые к вашим текущим задачам, но способные пригодиться для решения задач будущих, а также расширяющие ваше понимание фреймворка Spark. Главы этой книги содержат достаточно материала, чтобы ее можно было использовать как справочник. Однако структура издания неслучайна: читая главы по порядку, вы сможете в целом понять суть фреймворка Apache Spark и узнаете, как заставить его работать по-настоящему.

В равной степени важно отметить то, чего эта книга, вероятнее всего, не даст. Она не задумывалась как введение в Spark или Scala; для начинающих есть несколько других книг и серий видеоуроков. Возможно, мы несколько пристрастны в данном вопросе, но книга *Learning Spark*, как и великолепная серия видеоуроков для начинающих Пако Натана (Paco Nathan) (http://shop.oreilly.com/product/0636920036807.do), — отличные варианты для начинающих изучать Spark.

Хотя мы сосредоточимся на производительности, книга не предназначена для обслуживающего персонала, поэтому такие вопросы, как настройка кластера и мультиарендность, не рассматривались. Мы предполагаем, что вы уже знаете, как использовать Spark в вашей системе, вследствие чего не предложим особой помощи в вопросе принятия высокоуровневых архитектурных решений. Планируются к выходу книги других авторов, посвященные обслуживанию фреймворка Spark, которые должны уже выйти к моменту чтения вами нашей книги. Если вы как раз занимаетесь обслуживанием или в вашей организации нет выделенного персонала для данной цели, надеемся, что эти книги помогут.

Версии Spark

Spark придерживается семантики контроля версий вида [старшая версия].[младшая версия].[обновление] с сохранением неизменности API для общедоступных, не экспериментальных и не предназначенных для разработчиков API в пределах старших и младших версий. Многие экспериментальные компоненты, включая `Datasets` — новый структурированный и сильно типизированный слой абстракции Spark для SQL, — самые интересные с точки зрения производительности. Фреймворк также старается добиться бинарной совместимости API версий с помощью MiMa[1]; поэтому при использовании стабильного API для выполнения задания в новой версии Spark повторная компиляция обычно требуется разве что при смене старшей версии.

При написании данной книги использовался API Spark 2.0.1, но большая часть кода будет работать и с более ранними версиями фреймворка. Мы старались отмечать те места, в которых это не так.

Почему Scala

В этой книге мы сосредоточимся на API Scala фреймворка Spark и предполагаем, что вы немного знаете этот язык. Частично такое решение принято в целях экономии времени и места; мы верим, что читатели, желающие использовать Spark с другим языком программирования, смогут понять примененные в этой книге концепции, не прибегая к переводу примеров на Java и Python. Но важнее то, что авторы верят: «настоящую» эффективную разработку на Spark легче всего проводить на языке программирования Scala.

Для ясности: описанные причины относятся исключительно к использованию Spark со Scala, существует множество более общих доводов за (и против) приложений на Scala в других контекстах.

[1] MiMa — программа управления миграцией (Migration Manager) для языка программирования Scala, отслеживающая бинарные несовместимости разных версий.

Чтобы стать специалистом по Spark, все равно нужно хоть немного разбираться в Scala

Хотя языки программирования Python и Java более распространены, для желающих поближе познакомиться с разработкой на Spark изучение Scala вполне оправдывает себя. В документации фреймворка могут быть неточности. В то же время удобочитаемость базы кода очень высока. Вероятно, глубокое понимание кодовой базы Spark важнее для продвинутого пользователя Spark, чем в случае других фреймворков. А поскольку Spark написан на Scala, будет непросто взаимодействовать с его исходным кодом без способности по крайней мере читать код на языке Scala. Более того, методы класса *Resilient Distributed Datasets* (RDD, отказоустойчивые распределенные наборы данных) очень напоминают методы API коллекций языка Scala. Спецификации функций RDD, например `map`, `filter`, `flatMap`, `reduce` и `fold`, практически идентичны их эквивалентам языка Scala[1]. По существу, Spark — функциональный фреймворк, в значительной степени основанный на таких концепциях, как неизменяемость и лямбда-выражения, поэтому некоторые знания функционального программирования могут значительно облегчить использование API Spark.

API фреймворка Spark для языка Scala легче использовать, чем API языка Java

Разобравшись со Scala, вы быстро обнаружите, что писать на нем приложения Spark намного удобнее, чем на языке Java. Во-первых, написание таких приложений на Scala требует намного меньшего количества кода, чем на Java, поскольку Spark в значительной степени полагается на встроенные определения функций и лямбда-выражения, которые гораздо более естественно поддерживаются в языке Scala (особенно справедливо это было до выхода Java 8). Во-вторых, командная оболочка Spark — мощный инструмент отладки и разработки, а она доступна только в языках, для которых существуют REPL (Scala, Python и R).

Язык Scala производительнее, чем Python

Может показаться заманчивым создавать приложения Spark на языке Python, поскольку он прост в изучении, писать код на нем легко, это интерпретируемый язык программирования. Кроме того, для него существует обширный набор инструментария для науки о данных. Однако написанный на языке Python код Spark часто работает медленнее, чем эквивалентный код, написанный для JVM, поскольку Scala — язык программирования со статической типизацией, а затраты на взаимодействие с JVM (между Python и Scala) могут быть очень высоки. Наконец, обычно компоненты Spark сначала пишутся на Scala, а только позднее

[1] Хотя, как мы обнаружим в данной книге, их влияние на производительность и семантика вычислений сильно отличаются.

транслируются на Python, так что для использования наиболее передовых возможностей Spark необходима JVM; в частности, особенно отстает поддержка MLlib и Spark Streaming.

Почему не Scala

Существует несколько причин и для того, чтобы вести разработку приложений Spark на других языках программирования. Одна из наиболее важных — предпочтения разработчика или команды. Существующий код, как внутренний, так и библиотечный, может также оказаться серьезным доводом в пользу применения другого языка программирования. Python — один из лучше всего поддерживаемых языков на сегодняшний день. Хотя код на языке Java может быть немного громоздким и иногда слегка отстает в смысле API, производительность при написании на другом языке JVM страдает совсем незначительно (в основном за счет преобразования объектов)[1].

Хотя для итогового издания этой книги все примеры представлены на Scala, мы постараемся перенести многие образцы со Scala на Java и Python в тех случаях, когда различия в реализации существенны. Со временем они будут выложены в нашем GitHub-репозитории (https://github.com/high-performance-spark/high-performance-spark-examples). Если вы хотели бы переноса конкретного примера, то, пожалуйста, напишите нам сообщение по электронной почте или зарегистрируйте соответствующую проблему в репозитории GitHub.

Spark SQL значительно уменьшает различия в производительности при использовании не-JVM-языков. В главе 7 мы рассмотрим варианты эффективной работы со Spark на подобных языках, включая поддерживаемые Spark языки Python и R. Кроме того, в этом разделе приведены рекомендации по применению FORTRAN, C и кода, ориентированного на конкретные GPU, в целях дальнейшего повышения производительности. Даже при разработке большей части Spark-приложения на языке Scala мы не обязаны все делать только на нем, поскольку специализированные библиотеки на других языках программирования вполне могут окупить накладные расходы на выход за пределы JVM.

Изучение Scala

Если нам все же удалось убедить вас использовать Scala, то для его изучения есть несколько замечательных возможностей. Spark 1.6 основан на Scala 2.10 и кросскомпилирован с помощью Scala 2.11, а Spark 2.0 основан на Scala 2.11 и, возможно, кросскомпилирован с использованием Scala 2.10; кроме того, он, вероятно,

[1] Конечно, когда речь идет о производительности, из каждого правила есть исключения. Производительность преобразования mapPartitions в Spark 1.6 и более ранних версиях весьма ограничена, о чем мы поговорим в разделе «Выполнение преобразований “итератор — итератор” с помощью функции mapPartitions» на с. 121.

подвергнется данной операции с применением 2.12 в будущем. В зависимости от того, насколько нам удалось убедить вас изучить Scala, и от ваших возможностей, существует множество различных вариантов, начиная с книг и массовых открытых курсов дистанционного обучения (massive open online course, MOOC) и заканчивая курсами профессионального обучения.

Что касается книг, отлично подойдет *Programming Scala, 2nd Edition* (http://shop.oreilly.com/product/0636920033073.do), хотя многие из приведенных в ней системных ссылок акторов при работе в Spark не годятся. Список книг по Scala приведен и на сайте, посвященном Scala (http://www.scala-lang.org/documentation/learn.html).

Помимо книг, описывающих Scala, существуют онлайн-курсы для изучающих этот язык. На портале Coursera размещен курс Functional Programming Principles in Scala («Принципы функционального программирования на Scala») (https://www.coursera.org/learn/progfun1), преподаваемый его создателем Мартином Одерски (Martin Ordersky), а на портале edX — курс Introduction to Functional Programming («Введение в функциональное программирование») (https://www.edx.org/course/introduction-functional-programming-delftx-fp101x-0). Кроме того, несколько различных компаний тоже предлагают видеокурсы по языку Scala, но мы лично не слушали ни один из них, так что рекомендовать их не можем.

Для предпочитающих более интерактивный подход множество разных компаний, включая Lightbend (бывшая Typesafe), предлагают курсы профессионального обучения (https://www.lightbend.com/services/training). Хотя мы сами не пользовались этими курсами, они получили немало положительных отзывов и известны желанием помочь командам инженеров или группе лиц войти в курс разработки на Scala именно для работы со Spark.

Резюме

Хотя максимальной производительности Spark можно добиться только при условии хорошего понимания языка Scala, работа на Spark не требует знания этого языка. Если для ваших задач лучше подходят другие языки программирования или инструменты, то обратитесь к главе 7, где описаны методики работы с другими языками. Эта книга ориентирована на тех разработчиков, которые уже владеют основами Spark, и мы благодарны вам за то, что углубить свои знания вы решили с помощью нашего издания. В следующей главе мы представим основы архитектуры Spark и базовые принципы вычислений, необходимые для эффективного использования этого фреймворка.