ПРИМЕНЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ SCALA В РЕШЕНИИ ЗАДАЧ ОБРАБОТКИ ДАННЫХ

Иванюк А. И., Клапаток И. А., Чибисов И. В.
Кафедра программного обеспечения информационных технологий, кафедра информатики, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: ivanyuk42@gmail.com

В рамках данного исследования проведен анализ и предпринята попытка использования языка программирования Scala для обработки больших объемов данных с целью получения определённых аналитических данных. Проведено сравнение технических средств для обработки больших данных на языке Scala с уже зарекомендовавшими себя технологиями Hadoop и MapReduce. Исследованы возможности применения фреймворка Apache Spark как платформы для обработки данных.

Введение

В последнее десятилетие в связи с развитием горизонтально масштабируемых систем и эффективных хранилищ структурированных и неструктурированных данных значительно увеличилась популярность систем для обработки и анализа больших объемов данных. Развившиеся технические средства позволяли организовывать обработку огромных массивов данных, в некоторых случаях покрывающих весь мировой объем данных конкретной предметной области [1].

Одновременно с этим свойства, специфичные для обработки данных, такие как неизменяемость данных, применение операций функторов и редукции свойственны так же и функциональному программированию. Язык программирования Scala является типичным современным представителем функционального языка программирования общего назначения, который работает на платформе JVM. Эта особенность позволяет достичь достаточной гибкости при использовании языка и обеспечивает доступ к большому числу библиотек. Использование данного языка также возможно для решения задач обработки больших объемов данных, что обусловлено функциональным подходом и гибкостью языка.

I. Большие данные

Традиционно для определения принадлежности задачи к области больших данных используют набор признаков называемых «три V». Эти признаки включают следующие элементы.

1. Объем (volume) – физически большой объем обрабатываемых данных;
2. скорость (velocity) – необходимость высокой скорости обработки информации, которая обусловлена наличием современных технологических средств, а также скорость прироста объема информации;
3. многообразие (variety) – возможность обработки разнородной информации, структурированной или неструктурированной [1].

Обработка данных, которым свойственны перечисленные признаки, требует специализированных средств. Такие средства включают NoSQL системы управления базами данных, средства распределенной параллельной обработкой данных, специализированные алгоритмы (например, MapReduce) и библиотеки (например, входящие в пакет Hadoop). NoSQL системы позволяют хранить неструктурированные данные, свойственные для больших данных. В то же время комбинация из использования систем MapReduce и Hadoop позволяют преобразовывать эти данные. Рассмотрим эту типичную комбинацию зарекомендовавших себя решений.

II. Hadoop и MapReduce

Hadoop представляет собой фреймворк и набор библиотек для создания и выполнения программ, работающих распределенно на большом количестве узлов. Он позволяет организовать распределенную и параллельную обработку данных, используя мощности большого количества машин одновременно.

Структурно Hadoop разделен на несколько составных частей.

- Hadoop Common – набор утилит для работы с другими составными частями, организации инфраструктуры и управления процессом обработки данных;
- HDFS – распределенная файловая системы, которая специализирована на хранении файлов большого объема;
- YARN – модуль отвечающий за планирование задач внутри кластера и управления его ресурсами;
- Hadoop MapReduce – элемент фреймворка и шаблон, позволяющий реализацию MapReduce алгоритмов и их выполнение в среде Hadoop [2].

Алгоритмы MapReduce позволяют обрабатывать большие объемы данных параллельно на многих узлах, при этом данные должны представлять собой набор пар «ключ – значение». Работа приложения, разработанного с использованием данных алгоритмов, состоит из двух шагов: Map и Reduce. Данные шаги основаны на двух функциях высшего порядка, которые ис-

пользуются в функциональных языках программирования и носят соответствующие названия. На шаге Map происходит преобразование данных, какая-либо их предварительная обработка, при которой происходит заданное преобразование значений из пар «ключ – значение» входных данных. На шаге Reduce происходит свёртка значений, соответствующих одинаковому ключу [3]. При этом происходит последовательное выполнение заданной операции с накоплением результата.

III. Scala

Язык программирования Scala представляет собой функциональный язык программирования с набором стандартных библиотек, которые, в том числе, включают функции работы с коллекциями. Семантическая близость операций Map и Reduce с операциями преобразования коллекций в функциональных языках программирования делает язык Scala хорошим кандидатом на реализацию соответствующих функций, пригодных к выполнению параллельно и распределенно.

Сама по себе стандартная библиотека языка Scala предоставляет возможности по работе с коллекциями данных только в рамках выполнения одного процесса на одном узле. Для выполнения распределенных вычислений на большом числе узлов необходимо использование специальных технических средств, которые выходят за пределы самого языка, но используют этот язык программирования как основу. Проект, использующий язык Scala для обработки больших объемов данных носит название Apache Spark.

IV. Apache Spark

Apache Spark – это фреймворк, использующий инфраструктуру Hadoop для распределенной обработки больших объемов данных. Он является заменой MapReduce и обладает рядом преимуществ перед ним. По аналогии с MapReduce данный фреймворк позволяет проводить последовательные операции обработки и свертки, однако, если MapReduce ограничен выполнением двух шагов, то Spark позволяет комбинировать операции преобразования данных в последовательности [4]. При этом явным недостатком MapReduce является необходимость записи и чтения данных с диска после каждого выполнения пары операций Map и Reduce. В случае же использования Apache Spark обработка данных выполняется в основном в оперативной памяти, что значительно увеличивает скорость его работы. В случае со Spark основным замедляющим фактором является возникающая необходимость перераспределять данные между узлами, которая возникает, например, при изменении ключа записи. Ключ записи используется для привязки данных к определенным узлам, что необходимо для быстрого выполнения операции свертки. В случае же его изменения, данные перераспределяются.

Для работы с данными Spark предоставляет две абстракции: трансформация и действие. Трансформации задают последовательность изменений изначального набора данных. Они включают в себя, например, такие операции, как map, filter или distinct. Применение трансформаций не вызывает реальных изменений данных, а лишь описывает будущие изменения. Для применения заданных трансформаций необходимо выполнить какое-либо действие. Действие – это операция, которая позволяет материализовать результат выполнения трансформаций. Примерами действий могут служить такие примеры, как saveAsTextFile (сохранение набора данных на диск), take (получение определенного количества элементов набора данных) или reduce (операция свертки).

Таким образом, Apache Spark позволяет добиться значительного выигрыша в скорости работы за счет активного использования оперативной памяти узла. Данное преимущество часто используется для реализации алгоритмов машинного обучения с использованием данного фреймворка. Напротив, MapReduce обладает значительно меньшей производительностью, но позволяет обрабатывать гораздо большие объемы данных. Если для Spark эксперты рекомендуют одновременную обработку не более чем 1-2 Тбайт данных, то Hadoop MapReduce может обрабатывать объемы вплоть до нескольких петабайт [5].

Также Apache Spark может быть использован в отвязке от инфраструктуры Hadoop и работать с различными хранилищами данных, например, с Apache Cassandra или Amazon S3 [5].

Перечисленные выше особенности позволяют использовать Apache Spark для решения широкого класса задач обработки больших объемов данных, в том числе задач машинного обучения и статистического анализа.

Список литературы

1. Силен, Д. Основы Data Science и Big Data. Python и наука о данных / Д. Силен, А. Мейсман, М. Али // Питер, 2018. – 336 с.
2. Уайт, Т. Hadoop. Подробное руководство / Т. Уайт // Питер, 2013. – 672 с.
3. Holmes, A. Hadoop in Practice / A. Holmes // Manning Publications, 2014. – 512 p.
4. Bengfort, B. Data Analytics with Hadoop / B. Bengfort, J. Kim // O'Reilly Media, 2016. – 288 p.
5. Frampton, M. Mastering Apache Spark / M. Frampton // Packt Publishing, 2015. – 320 p.