ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

Обзор основных возможностей Apache Spark

Пальмов Сергей Вадимович, кандидат технических наук, доцент
Поволжский государственный университет телекоммуникаций и информатики,
Самарский государственный технический университет (г. Самара)
Поскиваткина Анастасия Анатольевна, студент 2 курса,
факультет «Информационные системы и технологии»
Поволжский государственный университет телекоммуникаций и информатики (г. Самара)

Современное информационное общество ежедневно порождает огромные массивы данных. Эффективная их обработка доступна только посредством применения специального инструментария, например, Apache Spark.

В работе рассмотрены системы для обработки больших данных Apache Spark и Hadoop, включая основные компоненты последней (YARN, HDFS и MapReduce). Приведены примеры использования библиотек Apache Spark, схема работы Spark-приложения и основные понятия архитектуры. Сформулированы выводы относительно пригодности Apache Spark для обработки больших данных.

Ключевые слова: *большие данные, Apache Spark, Apache Hadoop, анализ данных, искусственный интеллект, машинное обучение.*

Overview of Apache Spark Key Features

Palmov Sergey Vadimovich, Candidate of Science Engineering, Associate Professor
Povolzhskiy State University of Telecommunications and Informatics,
Samara State Technical University (Samara)
Poskivatkina Anastasiya Anatolyevna 2nd year student,
faculty "Information Systems and Technologies"
Povolzhskiy State University of Telecommunications and Informatics (Samara)

The modern information society daily generates huge amounts of data. Effective processing is available only through the usage of special tools, for example, Apache Spark.

The paper considers big data processing systems Apache Spark and Hadoop, including the main components of the second one (YARN, HDFS and MapReduce). Examples of using the Apache Spark libraries, the scheme of the Spark application and basic concepts of architecture are given. The conclusions regarding the suitability of Apache Spark for big data processing are formulated.

Keywords: *big data, Apache Spark, Apache Hadoop, data analysis, artificial intelligence, machine learning.*

DOI: 10.5281/zenodo.3888093

Данные непрерывно накапливаются практически во всех сферах человеческой жизни. К ним относится любая отрасль, связанная либо с человеческими взаимодействиями, либо с вычислениями. Например, астрономия, метеорология, инфокоммуникации, торговля [1, с. 25].

К основным источникам данных относят: Интернет (соцсети, форумы, блоги, СМИ и прочие ресурсы); корпоративные архивы документов; показания датчиков, приборов и других устройств [2, с. 79].

Этот огромный объем данных, часто бессистемных, которые хранятся на каком-либо цифровом носителе, определяют как Big Data.

Чтобы данные считались Big Data, они должны обладать следующими признаками (правило «трех V»):

- объем (Volume) – к Big Data относят очень большие массивы данных;
- скорость обработки (Velocity) – данные регулярно обновляются и требуют постоянной обработки;
- разнообразие (Variety) – данные в массивах могут иметь неоднородные форматы, быть структурированными полностью или частично, а также накапливаться бессистемно.

В современных системах рассматриваются два дополнительных признака:

- изменчивость (Variability) – потоки данных могут иметь сезонности, периодичность, пики и спады;
- значимость данных (Value) – данные должны обладать ценностью.

Таким образом, большие данные (Big Data) – это структурированные и неструктурированные данные огромных объемов и разнообразия, а также технологии поиска, методы их обработки, которые позволяют распределено анализировать данные [1, с. 31].

Анализ больших данных проводится с целью получения новых, ранее неизвестных знаний. Например, для того чтобы быстрее реагировать на изменения рынка, получить конкурентные преимущества, повысить эффективность производства нужно получить, обработать и проанализировать огромное количество данных, при этом человек получает конкретные и нужные ему знания для их дальнейшего применения. Следовательно, требуется внедрение все более эффективных систем хранения и манипулирования данными, то есть технологии Big Data.

ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

Основные принципы работы с Big Data:

• Горизонтальная масштабируемость: система, в которой хранятся данные, должна быть расширяемой.

• Отказоустойчивость: система должна сохранять свою работоспособность после отказа одного или нескольких вычислительных узлов.

• Локальность данных: обработка данных должна, если это возможно, производиться на той же машине, где эти данные хранятся.

К технологиям обработки данных относятся: SQL, NoSQL, SAP HANA, Hadoop, MapReduce, Apache Spark.

SQL – язык структурированных запросов, позволяющий работать с базами данных. С помощью SQL можно создавать и модифицировать данные, а управлением массива данных занимается соответствующая система управления базами данных.

NoSQL – термин расшифровывается как Not Only SQL (не только SQL). Включает в себя ряд подходов, направленных на реализацию базы данных, имеющих отличия от моделей, используемых в традиционных реляционных СУБД. Их удобно использовать при постоянно меняющейся структуре данных. Например, для сбора и хранения информации в социальных сетях.

SAP HANA – высокопроизводительная NewSQL платформа для хранения и обработки данных (NewSQL – это класс современных реляционных СУБД, стремящихся совместить в себе преимущества NoSQL и транзакционные требования классических баз данных). SAP HANA обеспечивает высокую скорость обработки запросов. Еще одним отличительным признаком является то, что SAP HANA упрощает системный ландшафт, уменьшая затраты на поддержку аналитических систем.

Остановимся на Hadoop и Apache Spark подробнее.

Hadoop

Hadoop – это набор программ с открытым исходным кодом, написанных на Java, которые используются для выполнения операций с большими объемами данных. Hadoop представляет собой масштабируемую, распределенную и устойчивую экосистему [3, с. 7]. Основными компонентами Hadoop являются:

• Hadoop YARN – управляет и планирует системные ресурсы, разделяя рабочую нагрузку на кластер машин.

• Распределенная файловая система Hadoop (HDFS) – это кластерная система хранения файлов, разработанная для обеспечения отказоустойчивости и высокой пропускной способности. Кроме того, она может хранить данные любого типа в любом возможном формате.

• Hadoop MapReduce – предназначен для проведения анализа данных [4].

MapReduce – это модель программирования для написания приложений, которые могут обрабатывать большие данные параллельно на нескольких узлах. MapReduce предоставляет аналитические возможности для обработки огромных объемов данных [5, с. 11]. Алгоритм работы MapReduce показан на рисунке 1.

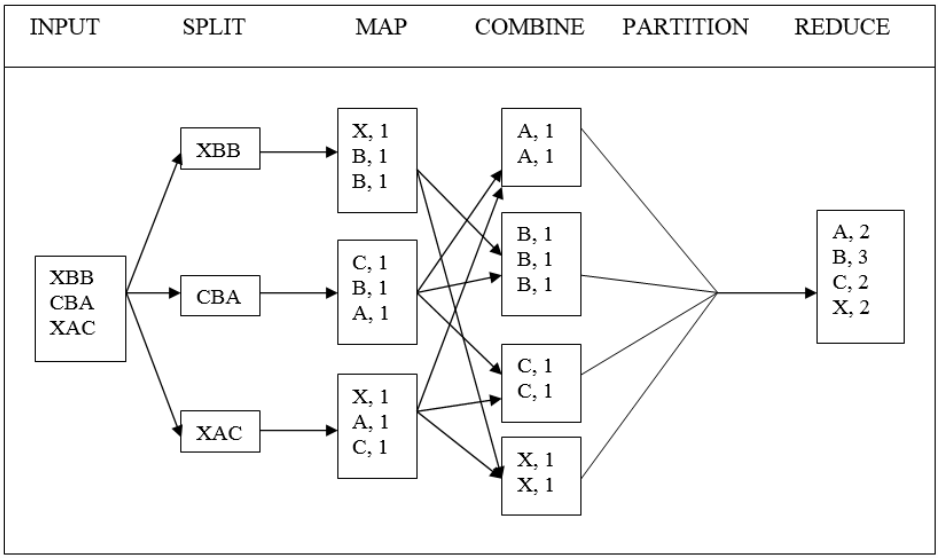

Рис. 1. **Модель MapReduce**

Пояснение шагов вычислительной модели:

• Input – главный узел кластера принимает исходные данные.

• Split – делит исходные данные на блоки данных и передает рабочим узлам кластера.

• Map – каждый рабочий узел применяет функцию «Map» к локальным данным и записывает результат в формате «ключ, значение».

• Combine – рабочие узлы перераспределяют данные на основе ключей так, что все данные, принадлежащие одному ключу, лежат на одном рабочем узле.

• Partition – происходит свертка предварительно обработанных данных.

• Reduce – главный узел получает ответы от рабочих узлов и на их основе формирует результат обработки данных [6, с. 40].

Hadoop устойчив к системным сбоям, поскольку после каждой операции данные записываются на диск.

Apache Spark

Apache Spark представляет собой платформу с открытым исходным кодом для параллельной обработки и анализа слабоструктурированных данных в оперативной памяти.

www.esa-conference.ru

Spark – это инструмент обработки данных. Он позволяет выполнять различные операции с распределенными коллекциями данных, но не предусматривает их распределенного хранения [7, с. 24].

Основными преимуществами Spark являются производительность, удобный интерфейс программирования с неявным распараллеливанием и отказоустойчивостью. Spark поддерживает четыре языка: Scala, Java, Python и R.

Фреймворк включает в себя четыре библиотеки, каждая из которых решает определенную проблему [8].

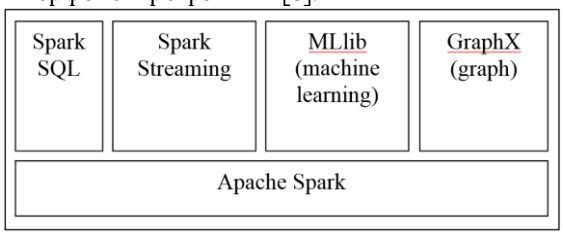

Рис. 2. **Стек библиотек Apache Spark**

Spark SQL – это модуль Apache Spark для работы со структурированными данными. Spark SQL позволяет запрашивать структурированные данные в программах Spark, используя запросы SQL.

MLlib – это масштабируемая библиотека машинного обучения Apache Spark. Spark эффективно реализует итеративные вычисления, позволяя MLlib работать быстро. MLlib содержит высококачественные алгоритмы, которые используют итерацию и могут выдавать лучшие результаты, чем однопроходные аппроксимации, используемые в MapReduce [8].

GraphX – это API-интерфейс Apache Spark для графов и параллельных вычислений.

Spark Streaming позволяет легко создавать масштабируемые отказоустойчивые потоковые приложения [7, с. 276].

Платформа Apache Spark – усовершенствованная MapReduce. В MapReduce данные считываются из кластера, выполняется необходимая операция, результаты записываются в кластер, обновленные данные считываются из кластера, выполняется следующая операция, ее результаты записываются в кластер и т. д.

Spark выполняет все аналитические операции в памяти практически в реальном времени: данные считываются из кластера, выполняются необходимые операции, затем результаты записываются в кластер, после чего процесс завершается. Пакетная обработка Spark по производительности в десять раз быстрее, чем MapReduce, а при анализе в оперативной памяти – в сто раз.

У Spark нет собственной системы управления файлами, поэтому требуется интеграция, если не с HDFS, то с какой-либо другой облачной платформой хранения, например, Amazon Web Services или Databricks [9, с. 124].

Каждое Spark-приложение состоит из управляющего процесса – драйвера (Driver) – и набора распределённых рабочих процессов – исполнителей (Executors).

Driver запускает главный метод приложения, в нем создается SparkContext (объект, который отвечает за реализацию операций с кластером). Spark Driver:

1. Запускает задание на узле в кластере или на клиенте и планирует его выполнение с помощью менеджера кластера.
2. Отвечает на пользовательскую программу или ввод.
3. Анализирует, планирует и распределяет работу между исполнителями.
4. Хранит метаданные о запущенном приложении и отображает в веб-интерфейсе.

Исполнитель (Executor) – распределенный процесс, который отвечает за выполнение задач. У каждого приложения Spark собственный набор исполнителей. Они работают в течение жизненного цикла отдельного приложения Spark.

- Исполнители делают всю обработку данных задания Spark.
- Сохраняют результаты в памяти, а на диске – только тогда, когда это специально указывается в программе-драйвере (Driver Program).
- Возвращают результаты драйверу после их завершения.

Схема работы приложения Spark показана на рисунке 3:

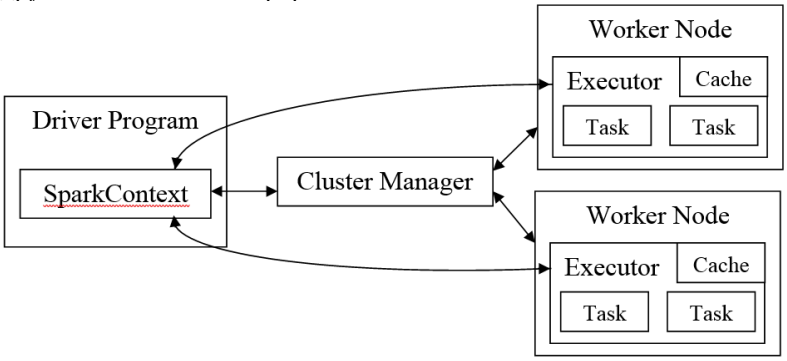

Рис. 3. **Схема работы приложения Spark**

ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

1. Приложение запускается и инициализирует SparkContext. Только при наличии SparkContext приложение называется драйвером.

2. Программа-драйвер запрашивает у менеджера кластеров (Cluster Manager) ресурсы для запуска исполнителей.

3. Менеджер кластеров запускает исполнителей.

4. Драйвер запускает код Spark.

5. Исполнители запускают задания и отправляют результаты драйверу.

6. SparkContext останавливается, а исполнители закрываются и возвращают ресурсы обратно в кластер.

Архитектура Spark основывается на двух главных понятиях:

• Устойчивые распределенные наборы данных (RDD, Resilient Distributed Datasets);

• Направленный ациклический граф (DAG, Directed Acyclic Graph).

RRD являются отказоустойчивыми. Они работают путем разделения данных на несколько разделов, которые хранятся на каждом узле-исполнителе. Каждый узел выполняет работу только на собственных разделах. Если исполнитель выходит из строя или не удается выполнить задачу, Spark восстанавливает только необходимые разделы из источника и повторно отправляет задачу для завершения.

Spark определяет набор API для работы с RDD, которые разбиты на две большие группы: Трансформации и Действия [7, с. 98]. Трансформации создают новый RDD из существующего, а Действия возвращают значение или значения программе-драйверу после выполнения вычисления над RDD.

Трансформации в Spark «ленивые». Это означает, что, когда Spark сообщается о создании RDD с помощью трансформаций существующего RDD, он не будет генерировать этот набор данных, пока не выполнится действие над ним или его дочерним элементом. Затем Spark выполнит трансформацию и действие, которое ее запустило. Поэтому Spark работает намного эффективнее [7, с. 152].

Каждый раз, когда выполняется действие над RDD, Spark создает DAG, ациклический граф операций, и последовательно запускает их на кластере согласно полученному графу. Каждая вершина в DAG – функция Spark (некоторая операция, которая выполняется над RDD). При построении DAG на основе RDD Spark проводит ряд оптимизаций, например, по возможности объединяет несколько последовательных трансформаций в одну операцию.

RDD был основной единицей взаимодействия с API Spark в версиях Spark 1.x. В Spark 2.x разработчики заявили, что теперь основным понятием для взаимодействия является Dataset. Dataset представляет собой надстройку над RDD с поддержкой SQL-like взаимодействия. При использовании Dataset API Spark позволяет задействовать широкий спектр оптимизаций, в том числе достаточно низкоуровневых. Но в целом, основные принципы, применимые к RDD, применимы также и к Dataset [10, с. 293].

Таким образом, можно сказать, что, Apache Spark хорошо подходит как для пакетной, так и для потоковой обработки, то есть – это гибридная среда обработки. Его гибкий распределенный набор данных (RDD) позволяет Spark прозрачно хранить данные в памяти и отправлять на диск только то, что важно или необходимо. В результате экономится много времени, которое тратится на чтение и запись на диск. Возможность обработки данных в режиме реального времени делает Spark лучшим выбором для анализа больших данных.

Литература:

1. Майер-Шенбергер В., Кукьер К. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим / пер. с англ. И. Гайдюк. М.: Манн, Иванов и Фербер, 2014. 240 с.

2. Фрэнкс Б. Укрощение больших данных: как извлекать знания из массивов информации с помощью глубокой аналитики / пер. с англ. А. Баранова. М.: Манн, Иванов и Фербер, 2014. 352 с.

3. Ohlhorst F. Big Data Analytics: Turning Big Data into Big Money. USA, North Carolina, Cary: SAS Institute Inc., 2012. 176 с.

4. Apache Hadoop [Электронный ресурс]. URL: http://hadoop.apache.org. (дата обращения: 13.04.20).

5. Веретенников А. В. BigData: анализ больших данных сегодня // Молодой ученый. 2017. № 32 (166). С. 9-12.

6. Лесковец Ю., Раджараман А., Ульман Д. Анализ больших наборов данных / пер. с англ. Слинкин А.А. М.: ДМК ПРЕС, 2016. 498 с.

7. Изучаем Spark. Молниеносный анализ данных / Х. Карау, Э. Конвински, П. Венделл, М. Захария. М.: ДМК Пресс, 2015. 304 с.

8. Apache Spark [Электронный ресурс]. URL: http://spark.apache.org. (дата обращения: 15.04.20).

9. Spark для профессионалов: современные паттерны обработки больших данных / С. Риза, У. Лезерсон, Ш. Оуэн, Д. Уиллс. СПб.: Питер, 2017. 272 с.

10. Уоррен Р., Карау Х. Эффективный Spark. Масштабирование и оптимизация. СПб.: Питер, 2018. 352 с.