СТАТЬИ

УДК 004.75

**СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ИНСТРУМЕНТОВ
ДЛЯ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ**

¹Богданов А.В., ¹Тхуреин Киав Лвин, ²Пья Сон Ко Ко, ²Чжо За

*¹Санкт-Петербургский государственный университет, Санкт-Петербург,
e-mail: bogdanov@csc.ru, trkl.mm@mail.ru;*

*²Санкт-Петербургский государственный морской технический университет,
Санкт-Петербург, e-mail: pyaesonekoko@gmail.com, kyawzaya4436@gmail.com*

Целью данной статьи является создание основной экосистемы, в которой Hadoop и Spark могут эффективно находиться в рабочем состоянии. Hadoop обеспечивает надежный, масштабируемый, отказоустойчивый сервис для крупномасштабной обработки данных на основе HDFS и MapReduce. HDFS является распределенной файловой системой и предоставляет распределенное хранилище для системы. MapReduce обеспечивает распределенную обработку для Hadoop. Однако MapReduce подходит не для всех классов приложений. Альтернативой для преодоления ограничения Hadoop являются новые системы времени исполнения в памяти, такие как Spark, которые предназначены для поддержки приложениями многократного использования рабочего набора данных в нескольких параллельных операциях. Слабость Spark заключается в том, что производительность ограничена памятью. Чтобы подчеркнуть различные аспекты двух сред выполнения больших данных, были введены PageRank. Мы измерили время работы, максимальное и среднее использование памяти и ЦП и пропускную способность, чтобы сравнить разницу в производительности между этими платформами для теста. В результате Spark работает быстрее с данными небольшого размера. Однако Spark потребляет больше памяти, а производительность Spark ограничивается размером памяти.

Ключевые слова: большие данные, распределенная система, СУРБД, Hadoop, Spark, MapReduce, HDFS

**PERFORMANCE COMPARISON USING SPARK AND HADOOP
FOR BIG DATA PROCESSING**

¹Bogdanov A.V., ¹Thurein Kyaw Lwin, ²Pyae Sone Ko Ko, ²Kyaw Zaya

¹Saint-Petersburg State University, Saint-Petersburg, e-mail: bogdanov@csc.ru, trkl.mm@mail.ru;

*²Saint-Petersburg State Marine Technical University, Saint-Petersburg,
e-mail: pyaesonekoko@gmail.com, kyawzaya4436@gmail.com*

The purpose of this article is to construct a main eco-system that supports Hadoop and Spark in a same working condition. Hadoop is developed as a solution for Big Data. It provides reliable, scalable, fault-tolerance and efficient service for large scale data processing based on HDFS and MapReduce. HDFS stands for the Hadoop distributed file system and provides the distributed storage for the system. MapReduce provides the distributed processing for Hadoop. However, MapReduce is not suitable for all classes of applications. An alternative to overcome the limitation of Hadoop is new in-memory runtime systems such as Spark, that is designed to support applications reuse a working set of data across multiple parallel operations. The weakness of Spark is that the performance is restricted by the memory. To stress the different aspects of two big data runtimes, we selected and ran PageRank benchmark on Hadoop and Spark runtime systems. We measured the running time, maximum and average memory and CPU usage, the throughput to compare the performances difference among these platforms for the benchmark. As result, Spark runs faster with small input size. However, Spark consumes more memory capacity and the performance for Spark is restricted by the memory.

Keywords: big data, distributed system, RDBMS, Hadoop, Spark, MapReduce, HDFS

Когда имеются в виду большие данные, то, скорее всего, речь идет об Apache Spark или Hadoop. В индустрии больших данных эти инструменты нередко рассматриваются как конкуренты. Тем не менее специалисты воспринимают их как дополнение друг друга.

Apache и Hadoop – это фреймворки, которые позволяют обрабатывать внушительные объемы данных, но используются для решения различных задач. Эта статья – своеобразная попытка совместить данные технологические решения, чтобы более эффективно работать с произвольными информационными данными.

Фактически Hadoop отвечает за формирование инфраструктуры данных распределенного типа: значительные коллекции информационных данных равномерно размещаются по большому количеству узлов, формирующих кластер, который включает в себя стандартизированные серверы. Данное обстоятельство свидетельствует об отсутствии необходимости в приобретении и обслуживании дорогостоящей техники узкого назначения. Hadoop тщательно контролирует и индексирует информационные данные, поэтому их анализ и обработка производятся с большей эффективностью, нежели прежде [1].

Spark представляет собой эффективный и очень удобный инструмент, который используется для обработки информационных данных. С его помощью можно исполнять разные операции, которые связаны с распределенными данными. Однако данный инструмент позволяет хранить эти данные распределенно. Spark функционирует с гораздо большей скоростью. По показателям производительности он существенно превосходит MapReduce, т.к. обработка данных в нем осуществляется совершенно иначе. Так, MapReduce обрабатывает данные пошагово, а Spark использует одновременно весь комплекс имеющихся данных [2].

Основная цель представленной статьи – сравнить показатели производительности Spark и Hadoop в процессе исполнения PageRank в целях обработки больших данных с применением PBVC (распределенной виртуальной вычислительной системы). Чтобы достигнуть намеченной цели, были выделены такие задачи, как:

- подборка и тщательный анализ наиболее подходящих комплексов распределенных вычислений, при учете вычислительных технологий в виртуальных системах;
- моделирование параллельного программирования, осуществляемое в целях усиления производительности и повышения уровня эффективности работы в PBVC.

Краткое описание Hadoop

Hadoop представляет собой проект, который обладает открытым исходным кодом и управляется Apache Software Foundation. Представленный фреймворк применяется в целях осуществления масштабируемых и надежных вычислений. Кроме того, Hadoop применим и для использования в качестве файлового хранилища (для файлов, имеющих общее назначение), т.к. в него вмещаются петабайты информационных данных. Большое количество коммерческих организаций применяют Hadoop в своей производственной и исследовательской деятельности.

Структуру фреймворка составляют 2 основных компонента:

- HDFS (файловая система распределенного типа), отвечающая за хранение информационных данных на Hadoop-кластере;
- MapReduce – уникальная система, позволяющая обрабатывать и вычислять значительные объемы информационных данных, которые находятся на кластере.

HDFS, являясь распространенной системой хранения информационных данных, применяется всеми Hadoop-приложениями. Она производит многочисленное копирование модулей информационных данных,

после чего распределяет сформированные копии по кластеру (его вычислительным узлам), позволяя добиться высочайшей скорости и качества вычислений [3]:

- в процессе загрузки информация распределяется по целому ряду вычислительных машин;
- система HDFS адаптирована скорее для потокового считывания информационных данных, чем для произвольных вычислений, выполняемых на нерегулярной основе;
- данные в HDFS фиксируются единовременно, после чего в них не могут быть внесены произвольные изменения;
- у приложений есть возможность записывать и считывать информационные данные NDFS непосредственно через Java-интерфейс.

MapReduce представляет собой программную модель и основу для разработки приложений, которые обеспечивают оперативную обработку информационных данных в значительных объемах на вычислительных узлах масштабных кластеров (параллельных) [4]:

- позволяет добиться эффективного распределения и распараллеливания вычислительных задач;
- обладает интегрированными инструментами поддержания работоспособности и устойчивости на случай, когда в тех или иных элементах будут возникать сбои;
- позволяет программистам достигать абсолютного уровня абстракции.

Корпоративные системы и Hadoop

RDBMS-системы (в переводе с англ. СУРБД – аббрев. от «Системы управления регуляционными базами данных») применяются в крупных коммерческих организациях по типовому принципу:

- система RDBMS интерактивного типа производит обработку запросов, которые приходят с интернет-страницы или иных приложений;
- после этого информационные данные получаются из реляционной базы, после чего зачисляются в файловое хранилище в целях последующей архивации и обработки;
- чаще всего информационные данные деформируются в так называемый куб OLAP.

К несчастью, RDBMS-системы, которые применяются сегодня, физически не имеют возможность хранить в себе значительный объем информации, формируемый в крупных коммерческих организациях. В этой связи появляется потребность в выборе компромиссного решения: либо лишь часть информации дублируется в RDBMS, либо она удаляется спустя конкретный времен-

ной интервал. Потребность в выборе подобных компромиссов исчезает, когда применяется Hadoop как промежуточный слой между файловым хранилищем и интерактивной информационной базой [5]:

- показатели производительности, которые фиксируются при обработке информации, увеличиваются в зависимости от размера информационного хранилища. Вместе с тем в серверах высокого уровня производительности увеличение объема информации и корреляция уровня производительности не имеют пропорциональной связи;

- в случае если в целях увеличения уровня производительности применяется фреймворк Hadoop, то в информационное хранилище просто интегрируются дополнительные узлы;

- с помощью фреймворка Hadoop можно обрабатывать и хранить большое количество информации. Но у него есть некоторые ограничения, по этой причине представленный фреймворк нельзя применять как операционную информационную базу;

- чтобы в Hadoop исполнить наиболее оперативную задачу, требуется несколько секунд времени;

- информацию, которая хранится в HDFS-системе, нельзя вносить коррективы;

- фреймворк Hadoop не имеет поддержки транзакционных операций.

Hadoop и СУРБД

СУРБД-системы обладают большим количеством преимуществ. Среди них [5]:

- с их помощью можно работать даже со сложными транзакциями;

- каждую секунду в СУРБД-системах обработке подвергается колоссальное количество запросов (сотни тысяч);

- информация по результатам функционирования отображается в интерактивном режиме;

- используется действенный и при этом простой язык формирования пользовательских запросов.

Тем не менее у рассматриваемых систем есть и «минусы», основными из которых являются:

- еще до импорта данных определяется их схема;

- предельная вместительность данных систем может достигать нескольких сотен Тб;

- предельный объем информации в каждом запросе СУРБД-системы достигает десятков Тб.

Hadoop и файловые хранилища

Деловая информация масштабных коммерческих организаций нередко хранится на крайне объемных серверах EMC, NetApp

и прочих типов. Эти вместительные серверы обеспечивают произвольный и крайне оперативный доступ к информации, а также имеют поддержки одновременного функционирования целого ряда пользовательских приложений.

Но если требуется хранить петабайты информации, то стоимость хранения каждого Тб может увеличиваться до существенных размеров. С данной точки зрения Hadoop – по-настоящему хорошая замена стандартизированным хранилищам файлов, при условии, что могут выбираться последовательные считывания, вместо произвольного доступа к информации, а коррективы информации можно ограничить лишь присоединяемыми записями.

ApacheSpark

Spark представляет собой один из наиболее выдающихся проектов разработчиков Apache. Он представляет собой эффективное средство, позволяющее мгновенно производить кластерные вычисления. В сравнении с проектом Hadoop Spark повышает скорость функционирования программного обеспечения на диске и в памяти (в 10 и 100 раз соответственно).

Проект Spark обладает следующими особенностями:

- сейчас проект уже предлагает API для Python, Java, Scala. Кроме того, разработчики активно готовят поддержку и других популярных программных языков, к примеру R;

- Spark великолепно сочетается с Hadoop-экосистемой и различными источниками информационных данных, включая etc, Cassandra, HBase, Hive, AmazonS3, HDFS;

- Spark позволяет оперировать кластерами, которые управляются ApacheMesos и Hadoop YARN. Кроме того, данный проект способен функционировать автономно [6].

Ядро Spark

Spark-ядро представляет собой движок-основу для масштабной распределенной и параллельной обработки информации. Оно исполняет следующие функциональные задачи [7]:

- восстановление памяти при сбоях, а также эффективное и оперативное управление ею;

- обеспечение отслеживания, распределения и планирования задач внутри кластера;

- обеспечение продуктивной работы с хранилищами информационных данных.

Spark предполагает применение RDD-концепции (стойкий распределенный комплекс информации) – не подлежащий изменению и устойчивый к отказам набор

объектов, подлежащих параллельной обработке. RDD в себя может включать любые разновидности объектов. Создается RDD посредством загрузки комплекса данных извне либо посредством распределения набора из приоритетного ПО. RDD имеет поддержку двух разновидностей операций:

- действий – операции (к примеру, расчет, редукция и пр.), способствующие возврату величины, которая формируется при ряде вычислительных процессов в RDD;
- трансформации – операции (к примеру, систематизация, фильтрация, демонстрация и пр.), которые осуществляются непосредственно над RDD. После трансформации появляется новый RDD, в котором отражен результат трансформационных процессов.

В Spark трансформационные процессы производятся в «пассивном» режиме. Это значит, что после трансформационных процессов результат исчисляется не сразу. Вместо этого они лишь фиксируют операцию, подлежащую осуществлению, и комплекс данных, которые подлежат изменению вследствие данной операции.

Трансформации вычисляются лишь в том случае, если провоцируется то или иное действие, после чего результат данного действия вновь отправляется к основному ПО. Этот дизайн позволяет повысить уровень продуктивности проекта Spark.

К примеру, когда для преобразования массивного файла применялись разные методы, после чего он был представлен первичному действию, Spark выполнит свои функции и возвратит результат лишь для первой строки, т.е. не будет обрабатывать весь объемный файл.

Изначально определено, что все RDD, подвергнутые трансформации, могут повторно вычисляться при каждом новом действии над ними. Но есть возможность хранить RDD и в течение длительного времени в памяти, применяя методики кэширования или хранения. В этой ситуации Spark обеспечит удержание на кластере необходимых компонентов, и пользователь сможет выполнять их запрос с гораздо большей скоростью [8; 9].

Конфигурация Spark в Hadoop

Все представленные выше технологические решения можно использовать и обособленно, что есть без их взаимодействия. Hadoop также в себя включает и компонент обработки данных MapReduce, и компонент хранения информации под названием Hadoop Distributed File System. В этой связи для полноценной обработки можно и вовсе не применять Spark.

Руководствуясь схожим принципом, можно использовать и Spark, не обращаясь к Hadoop. Но тут есть своя специфика. Так, в частности, структура Spark не имеет собственной системы управления файлами, поэтому при ее использовании нужно внедрять HDFS.

Вместе с тем Spark создавался под Hadoop. Поэтому мы приняли решение, что их целесообразно применять вместе.

Названные выше технологические решения подразумевают отличную реализацию функции восстановления после внезапных сбоев. Однако для этих целей в них используются абсолютно различные принципы. Фреймворк Hadoop обладает высокой устойчивостью ко внутрисетевым сбоям, потому что после завершения осуществления вычислений данные сохраняются на диск.

В случае с фреймворком Spark восстановление осуществляется благодаря сохранению объектов в наборах кластеров. RDD отвечает за восстановление информации из памяти или непосредственно с дисков, если произошли сбои или ошибки в работе [10].

Рис. 1. Конфигурация Spark в Hadoop

Тестирование и анализ производительности с использованием Spark и Hadoop для обработки больших данных

На представленном графике (рис. 2) продемонстрировано сравнение показателей производительности фреймворков Spark и Hadoop в ходе исполнения PageRank. Представленная таблица демонстрирует, насколько Spark ускоряется, в сравнении с Hadoop, при вычислении аналогичных комплексов данных.

	2M	4M	8M	16M	30M
Spark	164	414	844	1975	9963
Hadoop	415	585	936	1701	4088
Ускорение	2.53	1.41	1.10	0.86	0.41

Рис. 2. Сравнение времени работы PageRank на Hadoop и Spark

Следует привести один распространенный алгоритм ранжирования интернет-ссылок. Он используется в отношении набора документов, которые взаимосвязаны с помощью гиперссылок (к примеру, интернет-странички), и определяет для каждого такого документа определенное значение в виде чисел, которое показывает его значимость во всей массе документов.

Алгоритм применим как в отношении интернет-страничек, так и ко всяким комплексам объектов, которые имеют связь друг с другом, обеспеченную ссылками друг на друга, т.е. ко всякому графу. Данное числовое значение показывает значимость интернет-странички. Чем большее количество интернет-ссылок существует на ту или иную интернет-страничку, тем большим авторитетом обладает данная страничка. Помимо этого, масса интернет-странички А зависит от массы интернет-ссылки, которая на нее ведет.

Исходя из сказанного выше, можно заключить, что PageRank представляет собой методику расчета массы интернет-странички посредством учета и суммирования авторитетности интернет-ссылок, которые на нее ведут [11].

Объемы входных комплексов информационных данных колеблются в пределах 2–30 млн интернет-страничек, а объемы входных информационных данных – в диапазоне 1–19,9 гигабайта. Нагрузка при работе производится с тремя итерациями.

Если размер ввода лежит в интервале 2–8 млн страничек, то Spark считается более производительным, нежели Hadoop,

с предельным показателем скорости максимум 2,53 раза. Но данное достоинство сокращается по мере увеличения объема ввода. Фреймворк Spark значительно замедляется при функционировании, если вводится 16 млн интернет-страничек, в отличие от Hadoop, и показывает вдвое большее время работы в случае, если вводится 30 млн интернет-страничек.

Заключение

Каждое из рассмотренных технологических решений может применяться обособленно. Hadoop включает в себя как элемент обработки информационных данных MapReduce, так и элемент хранения информационных данных Hadoop Distributed File System, по этой причине для осуществления полноценных вычислений не требуется применение Spark.

По тому же принципу, и Spark может применяться без фреймворка Hadoop, однако тут есть определенная специфика. Spark не включает в себя независимую систему, позволяющую управлять файлами, по этой причине он требует внедрения HDFS. Наряду с этим Spark разрабатывался специально для Hadoop, по этой причине мы приняли решение, что целесообразно их задействовать вместе.

Обе технологии характеризуются хорошей опцией оперативного восстановления при нарушениях и сбоях в работе, хотя она реализована совершенно по-разному. Технология Hadoop отличается повышенной устойчивостью к нарушениям функционирования системы, т.к. после исполнения

каждой отдельно взятой операции информация фиксируется на диске. В Spark же восстановление информационных данных реализуется за счет того, что данные сохраняются в кластерных наборах. Информация может храниться на дисках или в памяти, а RDD отвечает за их полноценное восстановление, если в системе произойдут ошибки или сбои.

Список литературы

1. Том Уайт. Hadoop. Подробное руководство. Серия Бестселлеры O'Reilly. СПб.: Питер, 2013. С. 27–29.
2. Lu Liu. Performance comparison by running benchmarks on Hadoop, Spark, and HAMR. Proquest LLC. 2016. P. 11–13.
3. Vohra, Deepak. Practical Hadoop Ecosystem. A Definitive Guide to Hadoop-Related Frameworks and Tools (1st ed.). Apress. October 2016. P. 429.
4. Wang Yandong, Goldstone Robin, Yu Weikuan, Wang Teng. Characterization and Optimization of Memory-Resident MapReduce on HPC Systems. 2014 IEEE 28th International Parallel and Distributed Processing Symposium. P. 799–808.
5. Богданов А.В., Тхурейн Киав Лвин, Е Мьинт Наин. Консолидация базы данных, используемая для частного об-
лака. GRID 2012: материалы V Международной конферен-
ции. 2012. С. 52–56.
6. Matei Zaharia, Mosharaf Chowdhury, Michael J, Frank-
lin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with
Working Sets. University of California, Berkeley. 2012. P. 2.
7. Hongwei Chen, Lin Han, Zhou Hu, Qiao Hou, Zhiwei Ye,
Jun Zeng, Jiansen Yuan. A Feature Selection Method of Parallel
Grey Wolf Optimization Algorithm Based on Spark. Intelligent
Data Acquisition and Advanced Computing Systems: Technol-
ogy and Applications (IDAACS) 2019 10th IEEE International
Conference on. 2019. vol. 1. P. 81–85.
8. Zaharia M., Chowdhury M., Das T., Dave A., Ma J., Mc-
Cauley M., Franklin M.J., Shenker S., Stoica I. Resilient Dis-
tributed Datasets: A Fault-Tolerant Abstraction for In-Memory
Cluster Computing. University of California, Berkeley, April
2012. P. 3–4.
9. Harifi Sasan, Byagowi Ebrahim, Khalilian Madjid,
Tan Ying, Takagi Hideyuki, Shi, Yuhui. Comparative Study of
Apache Spark MLlib Clustering Algorithms. Data Mining and
Big Data. Lecture Notes in Computer Science. Springer Interna-
tional Publishing. 2017. P. 61–73.
10. Satish Gopalani, Rohan Arora. Comparing apache
spark and map reduce with performance analysis using k-
means. International Journal of Computer Applications. 2015.
Vol. 113 (1). P. 9–10.
11. Atish Das Sarma, Anisur Rahaman Molla, Gopal Pan-
durangan, Eli Upfal. Fast Distributed PageRank Computation.
Theoretical Computer Science. 2015. V. 561. P. 113–121.