Big Data Analysis with Apache Spark

Pallavi Singh
Software Engineer
J.C Penney

Saurabh Anand
Software Engineer
JC Penney

Sagar B. M.
Professor, RV College of Engineering, Bangalore

ABSTRACT
Manipulating big data distributed over a cluster is one of the big challenges which most of the current big data oriented companies face. This is evident by the popularity of MapReduce and Hadoop, and most recently Apache Spark, a fast, in-memory distributed collections framework which caters to provide solution for big data management. This paper, present a discussion on how technically Apache Spark help us in Big Data Analysis and Management. The paper aims to provide the conclusion stating apache Spark is more beneficial by almost 50 percent while working on big data. As when data size was increased to $5*10^6$ the time taken was drastically reduced by around 50 percent compared to when queried Cassandra without Spark. Cassandra is used as Data Source for conducting our experiment. For this, a experiment is conducted comparing spark with normal Cassandra DataSet or ResultSet. Gradually increased the number of records in Cassandra table and time taken to fetch the records from Cassandra using Spark and traditional Java ResultSet was compared. For the initial stages, when data size was less than 10 percent, Spark showed almost an average response time which was almost equal to the time taken without the use of Spark. As the data size exceeded beyond 10 percent of records Spark response time dropped by almost 50 percent as compared to querying Cassandra without Spark .Final record was analyzed at $5*10^6$ records. As the data size was increased, Spark was proved better than the traditional Cassandra ResultSet approach by almost reducing the time taken by 50 percent for really big dataset as our case of $5*10^6$ records.

General Terms
Bigdata

Keywords
Spark, RDD, MapReduce, Hadoop, Cassandra

1. INTRODUCTION
Data science has matured massively over the past few years and along with that the need for a different approach to data and its immensity also increased. An efficient framework is essential to design, implement and manage the required pipelines and algorithms to fulfill the computational requirements of immense data analysis [1]. To solve the afore mentioned problem Apache Spark [2] is a prominent system for large-scale data analysis across a variety of operations. Apache Spark is for distributed computing. A typical Spark program runs parallel to many nodes in a cluster.

The need for this fast and scalable system led to evolution of Spark. However, the developer needs to have requirement clear for the use of Spark. One should not prefer Spark for small data set as cluster environment needs a good understanding.

This paper, aims a demonstration to show how Spark is efficient for big data.

1.1 Apache Spark
It is a framework for performing general data analytics on distributed computing cluster [3] . Apache Spark is a all-around cluster computing engine with its support in Scala, Java and Python and libraries for streaming, graph processing and machine learning. Spark is an open source big data management framework which is built around easiness of use, speed and high degree analytics. Spark gives us a wide-ranging, collective framework for big data management and processing requirements with a variety of data sets that are diverse in nature as well as the source of data. It makes use of memory and can exploit disk also for processing data. The concept of MapReduce [4] is basically used by Spark and goes beyond it to efficiently use varied computations which includes Interactive Queries and Stream Processing.

2. BACKGROUND STUDY
Business is growing at a rapid pace and so is the data generated. Internet has evolved as a tool for public and data began to increase both in interconnectedness and volume. Analyzing this data has become need for the hour.

Data can be fetched without cluster computing for smaller data set but if data set is huge, processing is slow which leads to need for cloud computing. An environment which provides streaming capabilities making platform for speedy data analysis is needed.

In recent years, software developers have been investing on ways of data processing.

Spark has evolved as one the most efficient data processing [5] technique. It is evolutionary change in bid data processing environments as it provides batch as well as streaming capabilities.

3. WORKFLOW ARCHITECTURE
Our Workflow architecture for the demonstration of working with Spark and Cassandra consist of creating Spark context and then retrieving data from Cassandra. The input from Cassandra is taken and is serialized into POJO (Plain Old Java Object). Based on architecture above, Spark application is created, it in turn creates the Spark context. After creation of Spark context driver and worker is initialized in cluster mode. Then the executer will have task for call of Cassandra and retrieve all data present inside respective table. This data is stored in RDD (Resilient Distributed Data Set). RDD is data unit of Spark which stores all Cassandra records which was mapped to our POJO (Object created on the basis of Cassandra row). Each dataset in RDD is divided into multiple logical partitions, and each partition may be computed on different nodes of the cluster. This is what makes the Spark more fast as it is separately executed on different nodes. Now in demonstration a action is applied on top of the RDD to retrieve the count of records. Figure 1 shows how Spark application is built and at last Cassandra data fetch task is created in worker.

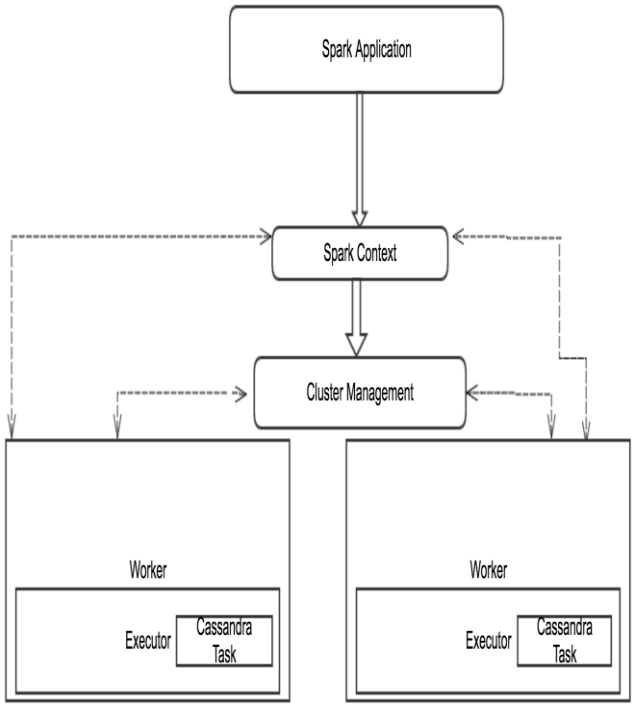

Figure 1. Spark Architecture for Cassandra Data Fetch

4. DRIVER AND EXECUTER MEMORY MANAGEMENT

The process that runs the code creates RDDs, for SparkContext in driver. The launch of Spark Shell signifies a driver creation in program. The application finishes on the termination of the driver.

The driver program splits the Spark application into the task. It schedules those task to run on the executor. The driver contains the task scheduler and distributes task among workers. The two main roles of drivers are to Convert user program into the task and Schedule task on an executor.

The machine on which the Spark Standalone cluster manager runs is the Master Node. The resources are allocated by Master "Workers" running throughout the cluster are used for creation of Executors for the "Driver". Driver process needs resources to run jobs/tasks. To fulfil this, the "Master" allocates the resources and "Workers" running throughout the cluster is used to create "Executors" for the "Driver". After that, Driver runs tasks on Executors [6].

5. STORAGE LAYER AND BIG DATA PROCESSING IN SPARK:

Spark need not use storage system provided by Hadoop [7]. It has support for storage systems which implement Hadoop APIs.

Spark holds the intermediary output in memory alternatively every time writing it to disk which makes it very time efficient mainly in scenarios where there is need to work around same dataset multiple times. Spark will try to store as much as data in memory and will only write left over or spilled data to disk. So it can store part of a data set in memory and the remaining which cannot be saved in memory data on the disk. Spark comes with performance advantage.

Big data queries is over all optimized with the lazy evaluation technique. Unless any processing is required for actions Spark will always postpone processing. This lazy evaluation method opted by Spark gives a lot of opportunities to introduce low-level optimizations. A transformation action of Spark that was count operation is used. All data from Cassandra in RDD is taken and then count action is done on that. But then also for big dataset, impact was minimal.

6. PERFORMANCE EVALUATION ON BASIS OF DEMONSTRATION

On analyzing the performance of both Spark job and Result Set of Cassandra to Fetch User Information Application, the performance of Spark job is more when compared to that of result set on Cassandra by almost Spark taking 50 percent less time when working with Big Data. In an ideal scenario, Cassandra result set is applicable only for small data set as in our case, for dataset size of 10^6 records Cassandra result set approach was time efficient as compared to Spark. As date records increased beyond 10^6 Spark was ultimately proved efficient.

For evaluating the performance of Spark and Cassandra one table in Cassandra is created with one column as primary key. Then through code increased the number of records in table in table gradually. And after every increase compared the time taken to do a fetch of total number of records in table.

Cassandra result set proved to be good initially for few records. But as records increased, the efficiency of Spark was far better as shown in table by average of almost 50 percent more efficient.

Below table shows analysis:

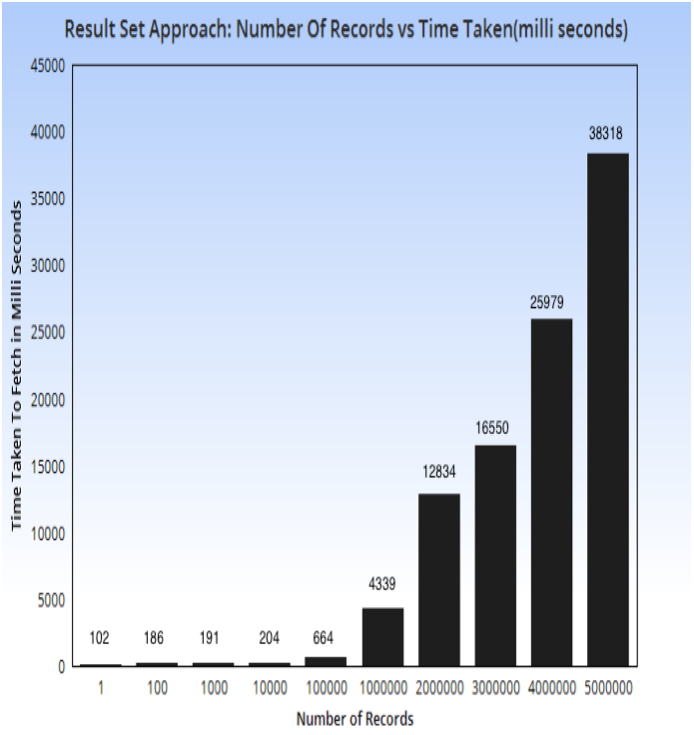

Figure 2. Result Set Approach for Cassandra Data Fetch Time Taken

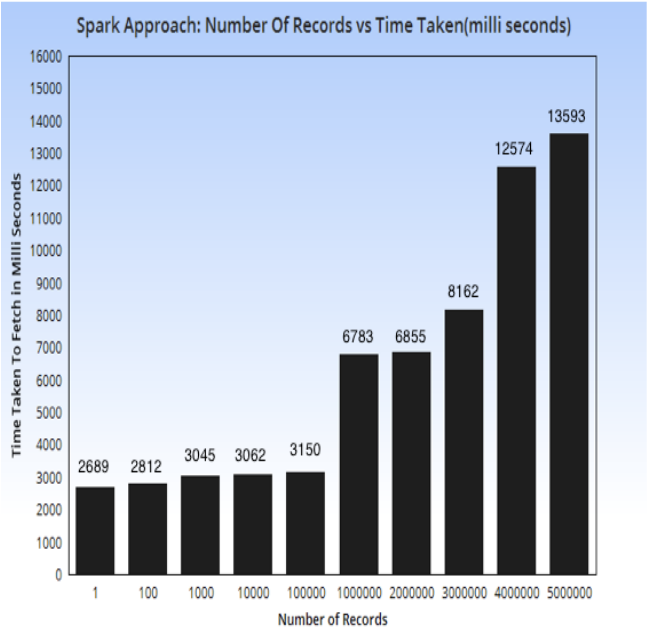

Figure 3. Spark Approach for Cassandra Data Fetch Time Taken

7. CONCLUSION

This paper, undergoes performance evaluation between Spark and Cassandra result set. For evaluation, Fetch user Information Application was created to get the retrieval time for records via Spark and Cassandra data set. A comparison was carried out by connecting to Cassandra and creating Cassandra data set against Spark result.

On comparing overall performance, Spark exceeds the Cassandra data set by taking almost half of the time lesser than the latter. In conclusion, the Spark is more efficient than the Cassandra call for the big data. As we can see from the figure 2 and 3, as data size increases from $4*10^6$ to $5*10^6$ the efficiency of Spark is even greater than 50 percent in terms of time taken for query and this trend continues as data increases.

Future work will include interactive analysis of exceptions, handling of streaming capabilities , the extension of safety and QoS policies, and the integration of other platforms, multidimensional data analysis, as another parallel execution framework. Apache Spark is predicted to be the next big change in data analytics and is regarded by many as a worthy research, to the MapReduce, the data processing which is powering Hadoop. Spark allows data processing unlike MapReduce. MapReduce causes considerable queuing delays not desirable in several real time data based applications.

8. REFERENCES

[1] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. SIGMOD Rec.,34(3):31–36, 2005.

[2] Spark: Cluster Computing with Working Sets. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. *HotCloud 2010*. June 2010.

[3] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Commun. ACM,51(1):107–113, 2008.

[4] J. Ekanayake, S. Pallickara, and G. Fox. MapReduce for data intensive scientific analyses. In ESCIENCE ’08, pages 277–284, Washington, DC, USA, 2008. IEEE Computer Society.

[5] R. Bose and J. Frew. Lineage retrieval for scientific data. processing: a survey. ACM Computing Surveys, 37:1–28,2005.

[6] Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, Ion Stoica. *HotCloud 2012*. June 2012.

[7] H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-reduce-merge: simplified relational data processing on large clusters. In SIGMOD ’07, pages 1029–1040. ACM, 2007.