IJCTA, 9(26) 2016, pp. 67-72

A Comparative Study on MapReduce and Spark

Blessy Trencia Lincy. S. S.[1] and Swarnalatha P.[2]

ABSTRACT

Big data in simple terms is the huge volume of data that is difficult to be processed, handled and managed. The term Big data is an emerging trend where a number of researches is carried out by researchers and data scientists. Various solutions is being provided for this big data problems. In this paper comparative study is made between the two important framework that is created as a solution for the big data problem. One is the Hadoop MapReduce framework and the second one is the Apache Spark. A study is made comparing the features, components, processing and techniques of both the framework. The evaluation gives an idea about the actual scenario in the today's world considering the researches done.

Keywords: Big data, Hadoop, MapReduce, Apache Spark

1. INTRODUCTION

In today's competitive world with the emerging trends and technology it is a very big challenge to process and handle this huge amount of data. Over the past decade a number of researchers and data scientists are striving hard to find different solutions for solving issues related to the data. Many techniques has been developed addressing the problems with data. The term Big Data just refers to the huge volume of data present today. Thus the big data is the main thing that is evolving around us when considering and analyzing today's trend in this leading and collaborative world. The level of data that is being generated now shows us that what has to be done in order to handle the data that will be generated in the mere future.

Big Data generates value in different domains that includes healthcare, retail, manufacturing, etc. The possibility of fusing data from various sources generates large datasets which requires Analytics tools in order to analyze it. The tool which is used can be said as effective when we see that with limited resources it can provide higher efficiency. Thus, data analytics plays a vital role involving the day to day activities of many private sectors, organizations and enterprises. The big data processing and storage is handled by the hadoop. The two major components of Hadoop is the MapReduce framework and the Hadoop distributed file system (HDFS). Both MapReduce framework and Spark is built upon hadoop. This paper describes about two important techniques used for processing the huge volume of data. One is the MapReduce which is a programming model that processes and generates large datasets. Next is the Apache Spark which is a cluster computing framework. A comparative study is made on both the framework and the issues related to performance and efficiency is studied and discussed here in this paper.

2. MAPREDUCE

The implementation of MapReduce provides a framework for processing and generating huge data sets. The two functions of MapReduce

[1] Research Scholar, School of Computing Science and Engineering, VIT University, Vellore, TN, India, *E-mail: blessylincy@gmail.com*

[2] Associate Professor, School of Computing Science and Engineering VIT University, Vellore, India, *E-mail: pswarnalatha@vit.ac.in*

1. Map function - Input to a map function is a key/value pair which then processes and produces intermediate key/value pairs as output to the map function.
2. Reduce function - The task of reduce function is to merge all the intermediate values that has the same key.

1.1. Features of MapReduce

a) Abstraction: The concept of hiding messy details about parallelization.

b) Fault-tolerance: The capability to withstand failure.

c) Data distribution: The distribution of data among different nodes and clusters.

d) Load balancing: Maintaining the balance in the nodes and clusters.

e) Locality optimization: Optimizing the locality of the data in terms of storage.

f) Inexpensive: Involves commodity hardware thus it is inexpensive.

The MapReduce programming model is successful due to the above mentioned reasons and a many problems are easily expressible as MapReduce computations. However MapReduce framework cannot support all the applications.

1.2. Processing of MapReduce

The input file is split into number of splits and it then copies the splits which is the program copies to a number of system in the cluster. Of it one of the copies become the master which assigns work for the slave nodes which does the task as instructed by the master node. The master picks up a node that is idle to allocate a map or reduce task. Then it is the task of the worker node to read the input and process it and produce the output as the intermediate key/value pair. This results from map function which will be buffered in the memory. The location of these buffers is sent to the master nodes. The master node then sends the location of these values to the reducer worker nodes so that they can be processed and the results can be obtained as key/value pairs.

The MapReduce programming model is very easy to use, even for the programmers without experience in the aspects of parallel and distributed systems. The concept of abstraction plays a vital role here by hiding complex, tedious details to the user of this system. The next impressive feature of MapReduce is a large variety of problems can be easily expressed in terms of the MapReduce computations. Many data mining, machine learning, web search engines make use of this efficient framework.

The MapReduce can scale to large cluster of commodity machines which may include thousands of systems. Thus, the applications that uses the MapReduce implementation can effectively make use of these resources and hence suitable for large computational problems. The concept of redundant execution is used to minimize the impact of the slow machines and to manage machine failures and lose of data.

3. APACHE SPARK

Apache Spark is an open source processing engine that is built for aspects including speed, easy usage and analytics. Provided a large volume of data which requires low latency processing, the Map Reduce program fails to do this because it takes more time to respond in processing input and output. Thus, Spark is an alternative providing low latency processing. Spark is 100 times faster than the Map Reduce framework while processing iterative algorithms and interactive mining of data. Apache Spark runs on and is built on top of Hadoop, Mesos, standalone, or in the cloud. It can access various different data sources such as HDFS, Cassandra, HBase, or S3.

3.1. Features of Spark

a) In memory cluster computing

b) Spark supports Java, Scala, and Python APIs thus ensures ease of development.

c) Spark not only supports 'Map' and 'reduce'. Spark combines SQL, streaming, Machine learning (ML), and Graph algorithms and complex analytics together seamlessly in the same application to handle a wide range of data processing scenarios.

d) Designed to cover a different range of workloads that includes,

- batch applications,
- iterative algorithms,
- interactive queries and
- streaming

e) Reduces the management burden of maintaining and handling separate tools.

There are three ways of Spark deployment as explained below.

i) Standalone " Spark Standalone deployment shows that the Spark occupies the top place of HDFS and the space is explicitly allocated for HDFS. Spark and MapReduce here in order to cover all spark jobs on cluster will run side by side. This shows that the applications can run Spark in a standalone environment.

ii) Hadoop Yarn " Hadoop Yarn deployment shows that, without any pre-installation or access to the root the spark runs on Yarn. This helps to integrate the Spark into the Hadoop stack or the Hadoop ecosystem. It also permits the other components to run on top of the stack.

iii) Spark in MapReduce (SIMR) " In addition to standalone deployment Spark in MapReduce is used initiate a spark job. The user can start Spark processing and can use shell with SIMR and without any administrative access.

3.2. Components of Spark:

1. **Apache Spark Core:** The Apache Spark Core is the general execution engine for spark platform which provides other functionalities, built upon them. In-Memory computing is provided by this execution engine and it references datasets from the external storage systems. The Spark core is responsible for scheduling, task dispatching, the basic input output operations and it also acts as an interface to users through the application programming provided by the RDD.
2. **Spark SQL**: Spark SQL is a component which is on top of the Spark Core where a new data abstraction called SchemaRDD is introduced. It is a domain specific language and provides support with command line interface. This provides support for both the structured and semi-structured data.
3. **Spark Streaming**: Spark Streaming leverages the fast scheduling capability of the Spark Core's in order to perform, streaming analytics. This enables to utilize the same set of code for batch analytics. It includes mini-batches of data and on this data it also performs RDD transformations.
4. **MLlib (Machine Learning Library):** Above the Spark framework is the MLlib which is a distributed machine learning framework and due to the distributed memory-based Spark architecture. According to the benchmark, which is done by the MLlib developers. It does it against the Alternating Least Squares (ALS) implementations. This library includes processing for statistics, correlations, hypothesis testing, sampling and classification and regression techniques that may include linear regression, logistic

regression, decision trees, etc. Spark MLlib is nine times as fast when compared to the Hadoop disk-based version of Apache Mahout.

5. **GraphX:** GraphX is a distributed graph-processing framework which is built on top of the Spark. By this processing framework the graphs becomes immutable thus making it unsuitable for graphs that needs to be updated. This component provides an API in order to express the graph computation, so that it can model the user-defined graphs by using the Pregel abstraction API. For this abstraction it also provides an optimized runtime.

4. EVALUATION

By analyzing and observing various researches done by different researchers and the scientists a question may rise that **"Whether Spark is going to replace Hadoop?"** or **"Spark or Hadoop MapReduce?".** The hadoop can be viewed as a general purpose framework that can support and help various models. On the other hand Spark can be an alternate to the MapReduce processing, rather than saying it replaces the hadoop framework.

The figure 1 shows the dependency of the MapReduce processing on the hadoop framework. The MapReduce jobs rely on the hadoop for both storage and computation. The storage issues are being addressed by the Hadoop Distributed Filesystem of the Hadoop.

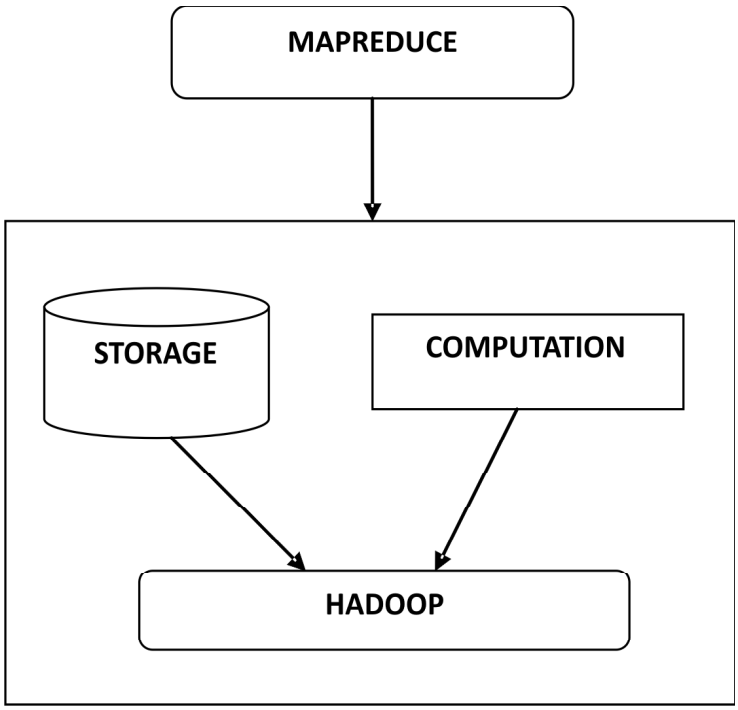

Figure 1: MapReduce framework upon Hadoop

The main distinction between the MapReduce and the Spark framework lies in the concept of iterative processing and message passing. Since, many graph algorithms involve iterative algorithms and some requires more number of iterations. In case of MapReduce it can perform only one iteration. In order to implement an iterative algorithm the programmer has to explicitly handle the iterations. This involves heavy I/O and additional job start up time resulting in low efficiency. The next feature is the message passing communication of information during the time of computation. The MapReduce framework does not provide this mechanism. Apache Spark framework provides both these features ensuring improved performance and efficiency.

Table 1
Comparison between MapReduce and Apache Spark

Description	*MapReduce*	*Spark*
Latency	High latency is provided.	Low latency is provided.
Computation	Multi-computation at multiple stages of application. Since, the intermediate results is stored in local disk the processing is slow.	In-memory computation is provided.
Data Sharing	Data sharing is slow due to a. Replication b. Serialization c. Disk I/O	Resilient Distributed Datasets is provided which stores the state of memory as object across the jobs and is sharable across the jobs.
Dependency	Depends on Hadoop for both a. Storage b. Computation	Depends on Hadoop only for Storage.
Intermediate Results	Intermediate results stored in disk.	Intermediate results stored in distributed memory. If not sufficient then stored in disk.
Dataset	Reference dataset from HDFS.	Reference dataset from shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

The figure 2 shows the dependency of the apache spark framework on hadoop. Unlike the MapReduce processing the Spark uses the hadoop framework only for the processing or the computation purpose. For storage purpose it has its own in memory clustering technique for handling data storage. Thus, for the computation part it may include Hadoop YARN or Apache Mesos for processing. By using the two framework the Spark can ensure effective scheduling, high availability, enhanced security and monitoring.

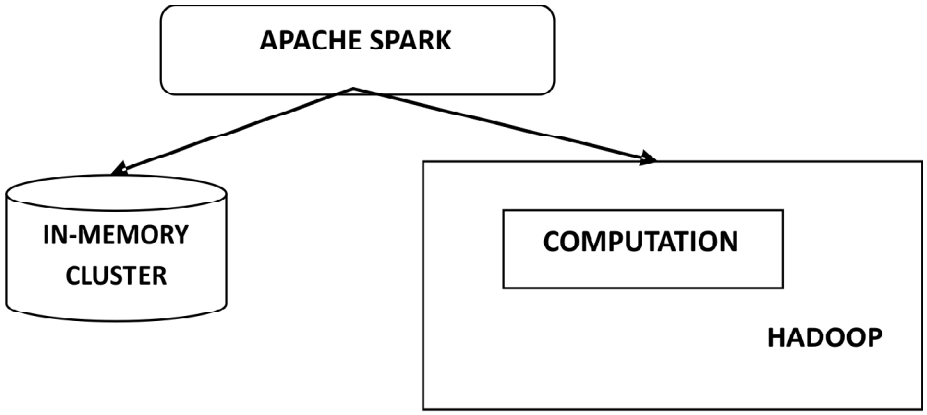

Figure 2: Apache Spark upon Hadoop

5. CONCLUSION

The paper gives an idea about the comparative features of the MapReduce processing and the spark framework. The various components of both the frameworks are discussed and analyzed. The Spark provides an environment which is more efficient and prominent compared to the MapReduce processing jobs in terms of speed and parallel processing. Since, the MapReduce framework cannot be suitable for all the applications the Apache Spark framework is more compatible when compared to the MapReduce framework built upon hadoop. Further research can be processed on trying to integrate some components of the MapReduce and the spark techniques.

REFERENCES

[1] Mapreduce: simpliûed data processing on large clusters, *The Proceedings of the 6th Symposium on Operating Systems Design and Implementation, J. Dean and S. Ghemawat.. Commun. ACM,* ***51(1),*** 107–113, 2008.

[2] Prominence of MapReduce in BIG DATA Processing, Shweta Pandey, Vrinda Tokekar, *The Proceedings of the Fourth International Conference on Communication Systems and Network Technologies CSNT,* 555 – 560, 2014.

[3] *http://spark.apache.org/*

[4] A Performance Analysis of MapReduce Task with Large Number of Files Dataset in Big Data Using Hadoop, , Amrit Pal, Kunal Jain, Pinki Agrawal, Sanjay Agrawal, *Proceedings, 2014 Fourth International Conference on Communication Systems and Network Technologies : CSNT,* 587 – 591,2014

[5] Breaking the Boundary for Whole System Performance Optimization of Big Data, Yan Li Kun Wang Qi Guo Xin Li Xiaochen Zhang xGuancheng Chen Tao Liu Jian Li & IBM Research, *2013 IEEE International Symposium on Low Power Electronics and Design (ISLPED),*126 – 131,2013.

[6] Addressing Big Data Problem Using Hadoop and Map Reduce, Aditya B. Patel, Manashvi Birla, Ushma Nair, *Proceedings, Nirma University International Conference on Engineering (NUiCONE),* 1 – 5, 2012.

[7] Experimental Framework for Searching Large RDFon GPUs based on Key-Value Storage, Chidchanok Choksuchat, Chantana Chantrapornchai, *Proceedings of the 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE),* 171 – 176, 2013.

[8] Think Big with Big Data: Identifying Suitable Big Data Strategies in Corporate Environments, Katharina Ebner, Thilo Bühnen, *Proceedings, 47th Hawaii International Conference on System Sciences (HICSS),* 3748 – 3757, 2014.

[9] An Ensemble MIC-based Approach for Performance Diagnosis in Big Data Platform Pengfei Chen, Yong Qi, Xinyi Li, *2013 IEEE International Congress on Big Data* (*BigData 2013*), 78 – 85, 2013.

[10] Enhancing MapReduce via Asynchronous Data Processing Marwa Elteir, Heshan Lin and Wu-chun Feng, *Proceedings, IEEE 16th International Conference on Parallel and Distributed Systems* (*ICPADS*), 397 – 405, 2010.

[11] Dynamic Data Partitioning and Virtual Machine Mapping: Efficient Data Intensive Computation, Kenn Slagter, Ching-Hsien Hsu, Yeh-Ching Chung, *Proceedings, IEEE 5th International Conference on Cloud Computing Technology and Science*, 220 – 223, 2013.

[12] An Approach for Log Analysis Based Failure Monitoring in Hadoop Cluster, Madhury Mohandas, Dhanya P M, Proceedings, *International Conference on Green Computing, Communication and Conservation of Energy* (*ICGCE*), 861 – 867, 2013.