International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 03 - Issue 02 || February 2018 || PP. 01-03

A Review on Spark and Map Reduce framework

Dr.E. Mary Shyla[1]
Assistant Professor
Department of Computer Science
Sri Ramakrishna College of Arts and ScienceFor women
Coimbatore

Jitha Janardhanan[2]
Research Scholar
Department of Computer Science
Sri Ramakrishna College of Arts and ScienceFor women
Coimbatore

Abstract: This paper deliberates two of the assessment of - Hadoop Map Reduce and the in recent times introduced Apache Spark – both of which provide a processing model for examining big data. Although both of these possibilities are based on the concept of Big Data, their performance varies significantly based on the use case under implementation. Apache Spark is setting the world of Big Data on fire. With aassurance of speeds up to 100 times faster than Hadoop MapReduce and comfortable APIs, some think this could be the end of Hadoop MapReduce. Sounds like Spark is bound to supplant Hadoop MapReduce

Keywords:HDFS,Spark,MapReduce,Hadoop

I. Introduction

Apache Hadoop is an open source scaffold that delivers solutions for handling big data along with widespread processing and exploration. It was shaped by Doug Cutting in 2005 when he was employed for Yahoo at the time for the Nutch search engine project. Hadoop has two foremost components titled HDFS (Hadoop Distributed File System) and the Map Reduce framework. Hadoop Distributed File System is said to be motivated by Google's The Google File System (GFS) andoffers a scalable, efficient, and replica based stowage of data at various nodes that form a part of a cluster.

As the progression of MapReduce in the Spark framework treats all the intermediate data as key/value tuples, a data cluster is the subset of all tuples with the same key . Because mapper and reducer are the caddies for map tasks and reduce tasks correspondingly, one way of implementations in Apache Spark is using hash algorithm to disseminate the clusters to each reducer, and all clusters which are processed by the same reducer comprise a partition. As the size of partitions hinge on on the number of relevant key/value tuples, data skew will give rise to the disproportion among different reducers because of the keys dispatching based on the hashing algorithm.

II. Hadoop Mapreduce

Hadoop MapReduce is a software scaffold for effortless script of applications which process enormousvolumes of data (multi-terabyte datasets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce *job* usually rips the input dataset into isolated chunks which are processed by the *map tasks* in anutterly parallel manner. The framework sort out the outputs of the maps, which are then input to the *reduce tasks*. Typically, both the input and the output of the job are stowed in a file-system. The scaffold takes care of scheduling tasks, scout out them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes[2].

III. Apache Spark

Apache Spark is plonking the sphere of Big Data on fire. With a promise of speeds up to 100 times wilder than Hadoop MapReduce and comfortable APIs, some ruminate this could be the end of Hadoop MapReduce. Sounds like Spark is obliged to switch Hadoop MapReduce.Apache Spark processes data in-memory while Hadoop MapReduce endures back to the disk after a map or reduce action, so Spark should outperform Hadoop MapReduce.Spark has comfortable APIs for Java, Scala and Python, and also includes Spark SQL (formerly known as Shark) for the SQL savvy.

Cost: considering Spark's yardsticks, it should be more fruitful since fewer hardware can accomplish the alike tasks much faster, specifically on the cloud where compute power is paid per use.
Failure Tolerance:Spark has rehashes per task and speculative execution—just like MapReduce. Nevertheless, because MapReduce bank on on hard drives, if a process clatters in the middle of implementation, it could linger where it left off, whereas Spark will have to start processing from the beginning. This can save time.

IV. Spark Architecture

Apache spark host RDD (Resilient Distributed Dataset)whichdelivers an application programming interface centered on a data structure. It is an immutable fault-tolerant, distributed collection of objects that can be operated on in parallel. It is the technology that parsimoniously make the most of in-memory LRU cache with possible on-disk eviction on memory full condition. And it puts all the dataset data on the local file systems during the "shuffle" process. The programmer can write a program in distributed manner with the Spark RDD which helps that data to endure in RAM thereby ensures effective processing of data. [3]

Spark SQL: It is a module in Spark that combine SQL database queries with algorithm based analytics. It works with structured data. It supports HiveQL query syntax with the Hive open source project Spark SQL supports the open source Hive project, and its SQL-like HiveQL query syntax. JDBC and ODBC connections, enabling a degree of integration with existing databases, data warehouses and business intelligence tools,it's also been supported by the Spark SQL. JDBC connectors with Apache drill provides access to broader range of data sources[4]

Spark Streaming: This module supports scalable and fault-tolerant processing of streaming data, and can integrate with established sources of data streams like Flume (optimized for data logs) and Kafka (optimized for distributed messaging). [4]

MLlib:This is Spark's scalable machine learning library, which implements a set of commonly used machine learning and statistical algorithms. These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.[4]

GraphX: This module began life as a separate UC Berkeley research project, which was eventually donated to the Apache Spark project. GraphX supports analysis of and computation over graphs of data, and supports a version of graph processing's Pregel API. GraphX includes a number of widely understood graph algorithms, including PageRank.[4]

Storage Options: Spark provides integration with many third –party open source or commercial data storage system which includes MapR (file system and database), Google Cloud, Amazon S3, Apache HBase,Berkeley's Tachyon project, Apache Hadoop (HDFS), Apache Cassandra, Apache Hive[4].But most of the developers chose the data storage system already available in workflow

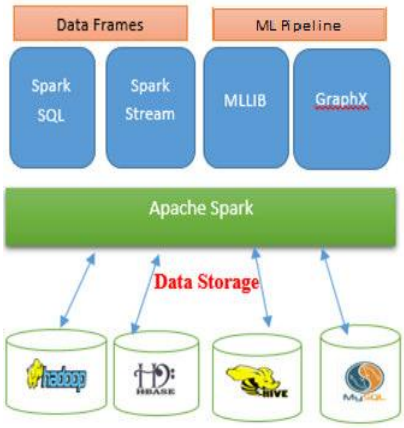


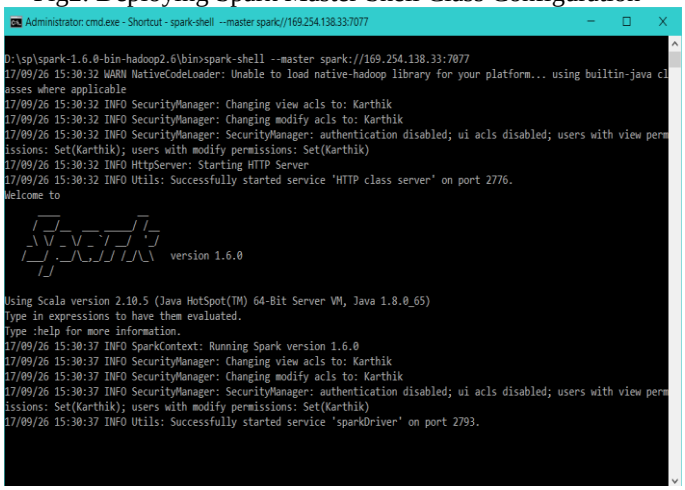
Fig1:Spark Architecture[3]

To sum up the main Spark features are listed below: [5]

- Spark supports three programming languages: Java, Scala and Python, providing necessary API-s
- Spark provides the ability to cache datasets in memory for interactive data analysis

- Reciprocal command line UI (in Scala and Python)
- Proven scalability to 2000 nodes in the research lab (EC2) and 1000 nodes in the production
- If data doesn`t fit into the memory, hard drive is used instead
- It is possible to replicate data in memory or on the hard drive
- Spark have additional operations for data processing like groupBy or filter

Fig2: Deploying Spark Master Shell Class Configuration



V. Comparision between Spark and Mapreduce

Spark	MapReduce
Apache Spark is simpler to Program doesnot require any abstraction	MapReduce is difficult to program with abstraction
It provides interactive mode	It has no inbuilt interactive mode
Upto 100 times faster than Hadoop as intermediate data/result is persist in memory	Slower as intermediate results stored in hard disk
It allows real streaming of data and processing	It can perform only batch processing on historical data
Higher memory requirement. Degradation of performance if data not fit in memory	Less memory requirement

VI. Conclusion

Spark and MapReduce framework both are popular distributed computing paradigm for providing an effective solution for handling this large amount of data called Big Data. The idea of the Spark framework is to enable iterative runs on the data sets. Apache Spark concept is slightly different from what Hadoop has. With Spark framework user loads the data and after that one can apply so much Map functions as it is needed, receiving intermediate data after each Map function.Many researchers and bloggers are claiming that Spark is 10 to 100 times faster than mapreduce framework. In this paper we are also comparing both the programming frameworks.

REFERENCES

[1]. ZhuoTanga, XiangshenZhanga, Kenli Li a, KeqinLi , "An intermediate data placement algorithm for load balancing in Spark", www.elsevier.com/locate/fgcs, June 2016
[2]. Dr. Urmila R. Pol," Big Data Analysis: Comparision of Hadoop MapReduce and Apache Spark", International Journal of Engineering Science and Computing, June 2016 Volume 6 Issue No. 6
[3]. Jai Prakash Verma1, Atul Patel, "Comparison of MapReduce and Spark Programming Frameworks for Big Data Analytics on HDFS"IJCS,Volume 7 • Number 2 March 2016 - Sept 2016 pp. 80-84
[4]. https://mapr.com/ebooks/spark/03-apache-spark-architecture-overview.html
[5]. Wikipedia, Apache Spark, http://en.wikipedia.org/wiki/Apache_Spark [04.11.2014]