International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-4 Issue-6, August 2015

Efficient Big Data Analysis with Apache Spark in HDFS

Amol Bansod

Abstract*— With the size of data increasing each day, the traditional methods of data processing have become inefficient and time consuming. Today, Facebook, Google, Twitter are generating Petabytes of data each day. This large amount of data is given the term 'Big Data'. To overcome this inefficiency, the processing of Data can be performed using Apache spark. Apache Spark is a fast, in-memory processing of large amount of data. In this research paper, the author discusses an efficient way of analyzing Big Data stored in Hadoop Distributed File System HDFS using Apache Spark framework, and its advantages over Hadoop MapReduce framework.

***Index Terms*— Big Data, Hadoop MapReduce, Spark**

I. INTRODUCTION

Data is growing faster than the computation speeds. Various sources of data are Public web, social media, business applications, data storage, machine log data, sensor data, archives, documents, and media [1], and the sources are growing. Big data analytics is the process of examining large amount of data to determine hidden patterns, unknown correlations and useful information that can be used for making better decisions. [2] The traditional analysis tools like UNIX shell commands, Pandas, and R get inefficient with huge data, as they work on a single machine, and a single machine can no longer process or even hold all of the data that we want to analyze. HDFS solves this problem by holding large amount of data up to petabytes of data and provide faster access to it. Data is stored by distributing it over clusters. Files are stored across multiple machines to make it more fault tolerant and ensure higher availability to parallel applications. [3] Apache Spark is a technique used to analyze the data stored in HDFS. It is a framework using which we can write applications to process the data in parallel. Spark has high scalability, very high processing speed and easy to use API's.

II. GOALS OF BIG DATA ANALYSIS

The main goal of big data analysis is to help the companies make better business decisions by enabling data scientists, Predictive modelers and analytics professionals to analyzeBig data [4]. It also helps in understanding the heterogeneity and commonality across subpopulations and explore the hidden structure of subpopulation of data and extract common features from it.

Manuscript published on 30 August 2015.
***** Correspondence Author (s)
Amol Bansod*, Department of Electronics and Telecommunication, V.E.S Institute of Technology, Mumbai (Maharashtra). India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

III. HADOOP DISTRIBUTED FILE SYSTEM

The Hadoop Distributed File System HDFS is designed to hold large amounts of data. It is a Java based file system and provides faster access to data.

It is highly scalable having scalability up to 200 PB of storage. HDFS is also highly reliable and fault tolerant system. A single cluster of HDFS containing 4500 servers can support billion files and blocks.

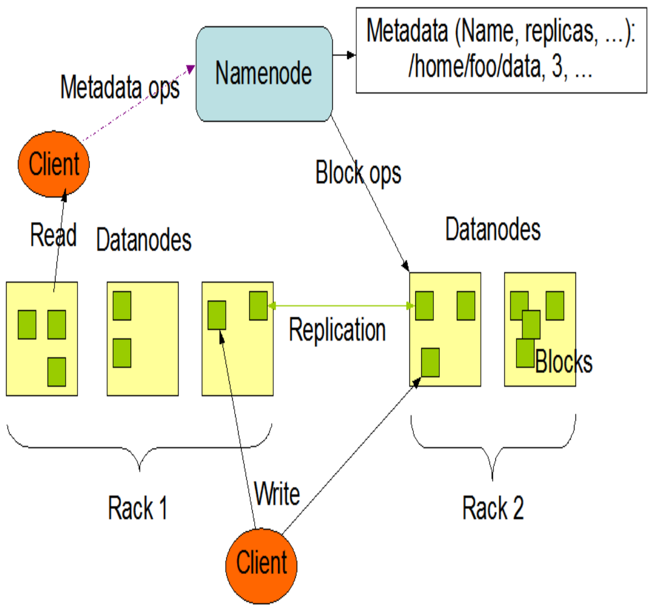

Fig 1: HDFS Architecture

Figure 1 shows the master slave architecture of HDFS. It consists of a NameNode and DataNode. NameNode is a centralized process maintains the directory tree of HDFS .It can performs common operations like rename, delete, open and close It does not store any data, but maintains a map of the blocks in the file, the file name, and the DataNode where the blocks are stored. [5] The data of the file is split into 3 parts, each of size 128 Megabytes, and each part is stored separately at multiple DataNodes. This is done so that the data is not lost even if one DataNode fails. Blocks are created or destroyed at the request of NameNode, which processes the requests from clients. Clients can communicate with the DataNodes directly to read or write data at the HDFS block level. [5] Apache Spark can be used with HDFS in following 3 ways: Standalone deployment, using YARN, and Spark in MapReduce.

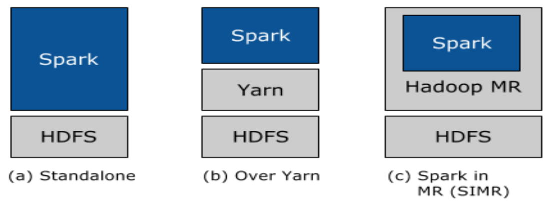

Fig 2: Deploying Spark over HDFS

A. Standalone Deployment

Spark can be deployed directly on HDFS and resources can be statically allocated on all or some machines in a Hadoop cluster. Spark can then be run with Hadoop MapReduce.

B. Hadoop Yarn Deployment

Spark can run on the HDFS with the help of YARN. It does not require any pre-installation or administrative access. This allows users to easily integrate spark and HDFS, and take full advantage of Spark.

C. Spark in MapReduce (SIMR)

In SIMR, the users who have access to Hadoop MapReduce v1 cluster can run spark directly on Hadoop MapReduce v1 without administrative rights. There is no need to install Spark or Scala on any node.

IV. BIG DATA ANALYSIS USING SPARK

Apache Spark is a high performance framework for analyzing large datasets. It was developed at UC Berkeley AMPLAB [7] as an alternative to Hadoop MapReduce framework. Figure 3 shows the architecture of Apache Spark. Apache spark consists of a driver program (SparkContext), workers also called executors, cluster manager, and the HDFS. Driver program is the main program of spark. SparkContext is the object that gets created during execution of spark program, and is responsible for entire execution of the job. The SparkContext object connects to cluster manager, which are used to manage the resources across cluster. Cluster managers provide Executors, which are used to run the logic and also storing the app data. Each application gets its own processes for duration of the whole application and run tasks in multiple threads and must be network addressable from worker nodes. [7]

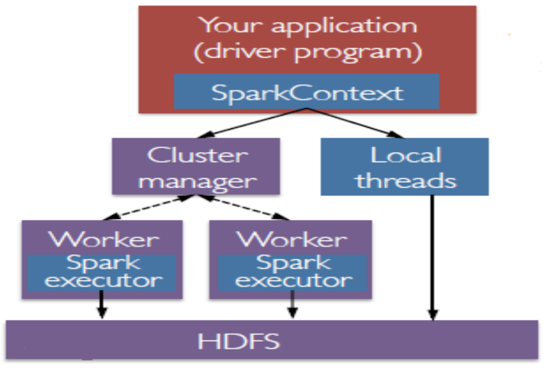

Fig 3: Spark Architecture

Spark is based on two concepts: Resilient Distributed Datasets (RDD) and execution engine Directed Acyclic Graph (DAG).

A. Resilient Distributed Datasets (RDD)

Resilient Distributed Datasets (RDD) are Spark's primary abstraction, which are a fault-tolerant collection of elements that can be operated on in parallel. They are immutable once you create an RDD. They can be transformed, or actions can be performed on them, but they cannot be changed. They help with rearranging the computations and optimizing the data processing. They are also fault tolerant because an RDD know how to recreate and compute the datasets [8]. RDD can be constructed by paralyzing existing collections such as lists, or by transforming existing RDD, or from existing files in HDFS. A spark programmer specifies the number of partitions for the RDD and if not specified, a default value is used.
There are two types of operations that can be performed on RDD's:

1) Transformation: When transformations are applied on RDD's, they return a new RDD and not a single value. Transformations are lazily evaluated, i.e. they are not computed immediately. They are executed only when an action runs on it. Some of the Transformation functions are map, filter, ReduceByKey, FlatMap and GroupByKey
2) Action: When Action operation are applied on RDD's, they evaluate and return a single value. All the queries that process the data are computed when an Action function is called, and return the result value. Some Action operations are first, take, reduce, collect, count, foreach and CountByKey.

B. Directed Acyclic Graph (DAG)

Spark consists of an advanced Directed Acyclic Graph (DAG) engine which supports cyclic data flow. Each Spark job creates a DAG of task stages to be performed on the cluster. DAGs created by Spark can contain any number of stages, as compared to MapReduce, which creates a DAG with two stages - Map and Reduce. This allows simple jobs to complete after just one stage, and more complex tasks to complete in a single run of many stages, rather than splitting it into multiple jobs. Thus, jobs complete faster than they would in MapReduce. [9]

V. SPARK COMPONENTS

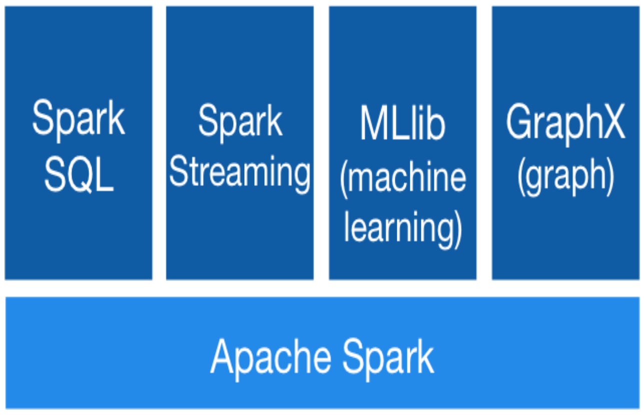

Fig 4. Spark Components

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Fig 4 shows the components of Spark framework. Along with core Apache spark component, there are 4 other components: Spark SQL, Spark Streaming, GraphX and MLlib Machine learning library.

A. Spark Core

Spark Core is the foundation of the framework. It provides basic I/O functionalities, distributed task dispatching, scheduling.

B. Spark SQL

Spark SQL allows running SQL like queries on Spark data using traditional BI and visualization tools. It provides support for structured and semi structured data by introducing a new data abstraction SchemaRDD. It also provides SQL language support, with command-line interfaces and ODBC/JDBC server.

C. Spark Streaming

Spark streaming allows processing the real-time data. It uses DStream, which is a series of RDD's, to process real-time data.

D. Spark GraphX

GraphX introduces the Resilient Distributed Property Graph, which is directed multi-graph having properties attached to each edge and vertex. GraphX includes a set of operators like aggregateMessages, subgraph and joinVertices, and optimized variant of Pregel API. It also includes builders and graph algorithms to simplify graph analytics tasks.

E. MLlib

MLlib is Spark's scalable machine learning which consists of utilities and common learning algorithms like regression, classification, collaborative filtering and dimensionality reduction.

VI. ADVANTAGES OF USING APACHE SPARK IN HDFS OVER HADOOP MAPREDUCE

A. Faster Processing

Spark uses in memory processing rather than storing the results in disc like MapReduce which makes it up to 100 times faster than MapReduce. The in-memory is possible due to use of Resilient Distributed Dataset (RDD) which allows it to store the all the data on memory and write it to disc only when needed, thus eliminating the need of disc read and write which are main time consuming factors of data processing.

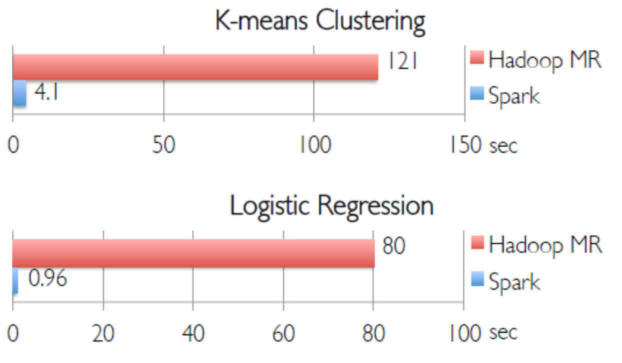

Fig 5. Two Iterative Machine Learning Algorithms

Figure 5 shows the comparison of how much time it takes for two iterative machine learning algorithms K-means clustering and logistic regression. For K-means clustering, MapReduce takes 121 seconds whereas Spark finishes it in 4.1 seconds. For logistic regression, Mapreduce requires 80 seconds whereas Spark completes it in less than a second. This is the performance benefit in Spark due to in-memory processing.

B. Ease of Use

Spark applications can be developed with Python, Java, Scala or R. Thus the developers can work on developing applications with the language they are comfortable.

C. Support for sophisticated Analytics

Spark is much more powerful than Hadoop MapReduce. Like MapReduce, Spark also has Map and Reduce functions, but along with it, it provides many more functions like join, Filter, Group-by etc. The instructions in Spark are higher level abstractions of the instructions in MapReduce.

D. Libraries

Spark has integrated libraries, like MLlib Machine learning, GraphX, Streaming and Spark SQL, which make things much easier for the developers. The developers otherwise have to use many unrelated packages, making things very complex.

E. Data Sources

Spark can run independently, or it can run on YARN cluster manager and read any existing hadoop data. For example HBase, HDFS.

VII. CONCLUSION

As we have entered the era of Big Data, new analytics tools like Hadoop MapReduce and Apache Spark have been developed to analyze and process this Big Data.

Apache Spark has gained significant momentum and is considered to be a promising alternative to support iterative processing logic and ad-hoc queries by replacing MapReduce. Apache Spark has received a lot of appreciation in many fields like pattern recognition, machine learning, data mining, information retrieval, and image retrieval. However, as the amount of data to be processed grows, many data processing methods have become less efficient.

This paper exploits the Apache Spark framework for efficient analysis of big data in HDFS, and compares it with other data analysis framework- Hadoop MapReduce. It is seen that Apache Spark increases the speed of computation of iterative algorithms and completes them in much less time as compared to Hadoop MapReduce. Apache Spark also provides a high performance, highly scalable and fault tolerant framework for big data analysis.

REFERENCES

1. Understanding The Various Sources of Big Data, https://datafloq.com/read/understanding-sources-big-data-infographic/338
2. Big data Analytics, http://www.sas.com/en_us/insights/analytics/big-data-analytics.html.

3. Hadoop Tutorial,YahooInc., https://developer.yahoo.com/hadoop/tutorial/index.html
4. Big Data Analytics, http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics
5. J. Shafer, S. Rixner, A.L. Cox, "The Hadoop Distributed Filesystem: Performance versus Portability", IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2010), White Plains, NY (March 2010).
6. https://databricks.com/blog/2014/01/21/spark-and-hadoop.html.
7. Apache Spark, http://spark.apache.org/
8. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley, 2011
9. Apachespark,https://www.mapr.com/products/product-overview/apache-spark.

Amol Bansod: Ex-student, Department of Electronics and Telecommunication, V.E.S. Institute of Technology, Mumbai, India.