ISSN: 2455-2631 © April 2016 IJSDR | Volume 1, Issue 4

A Complete Introspection on Big Data and Apache Spark

[1]S. Praveen Kumar, [2]Dr. Y Srinivas, [3]Dr.D.Suba Rao, [4]Ashish Kumar
Gitam University

ABSTRACT: **Big Data is a term for massive data sets which have complex structure and cannot be handled by standard software's. This paper provides complete understanding of how big data is handled using Hadoop framework providing a diagrammatical hardware implementation. Hadoop framework allows distributed processing of large datasets across clusters of computers. We also discuss about Apache Spark which overcomes the latency problem in Hadoop MapReduce. Apache Spark is a lightning-fast cluster computing designed for fast computing.**

1. Introduction

Big Data has given a 'ray of hope' to companies and enabled them to make use of data of any size and volume. Earlier relational database management systems (RDBMS) were used to store data and access them. In today's era data dumped is so big that it cannot be handled by RDBMS anymore. Huge amount of information is created every day. In fact, the information that we could generate from the beginning of time until 2003, can now be created in 2 days. At this speed we may end up storing 3.2 zettabytes to 40 zettabytes of data by 2020. As per today's statistics Google processes 3.5 billion queries, 1.8 million Facebook likes, 200,000 photo uploads, 204 million emails, 278 thousand tweets and 307 hours of video is uploaded on YouTube in a single day. RDBMS works in accordance with the data ware house, and thus can store only 10% of the whole data at a time. For example, if we have 400 GB of data to be stored then only 40 GB of data can be stored at a time. So the same process has to be continued 10 times, which consumes more time. One of the other major disadvantages of RDBMS is that it can store data in only tabular format, but we can have data in any format like photo, audio, video, text, etc. So, in order to store this data into RDBMS we need to first convert it into different formats using ETL [1] (Extract, Transform and Load) tool. This shows that RDBMS is not scalable.

Big data provides solution to these problems by storing data efficiently of any size and format which are in an unstructured form. The second problem of storing all this data in a single shot is also done by applying some logics using Hadoop framework. IBM proposed that big data should have the following 3 characteristics. These characteristics are known as the 3 V's of big data[2]:

- Volume- It determines the actual problem and assumes the volume of data which is needed to solve the problem.
- Velocity- It keeps a track of the server logs. Which means how frequently the data changes or the rate at which the data changes.
- Variety- Big data provides a solution to different formats of data which is needed for the problem.

Cloud services, Support Vector Machines [3] and other machine learning techniques helped to reduce the storage problem till some extent but could not provide a scalable and a faster access to data which proved big data to be the optimal solution to storage problems [8].

"Big data makes us smarter, not wiser" – Tim Leberecht.

The bits of data collected via our mobile phones, GPS, sensors devices are no longer useless. Every bit of data collected gets stored and processed to derive useful insights about us (customers).Few things that big data is capable of doing are mentioned below[4][5]:

- Diagnostic analysis- By collecting related information about a particular event such as an global warming and what resulted in its happening, what impacts it has, what steps were taken to control it or avoid it, etc. can be used for future solutions to such problems.
- Predictive analysis-This is something that every human being does. We focus on the areas which leads us to our goal. This is also done by big data, for example, we have a hotel chain all over the globe. And we need to find which of these hotels will not meet their target sales. And if we know it, we can focus our efforts on these hotels. This becomes a classic problem of predictive analysis.
- Relative analysis- We can always find the things which are related to each other. For example, number of sales employees has literally no relation with sales. Then possibly you can reduce the number of sales employees if that does not do any other loss.
- Prescriptive analysis- This is the future of analytics. Let's say we are trying to predict a terrorist attack in a popular destination and possible strategy to safely move people. To make this prediction, you need to make a series of prediction, which might involve predicting number of tourist then number of tourist in that location, then predicting area which can be affected by a blast etc.
- Monitoring a situation as it develops-Real-time analytics involves gathering and interpreting data "as it happens" in order that insights gained from it will be as timely as possible. In business, the motivation is competition – the more closely your data resembles what is actually happening, right now, in your market, the better the chance you can react and respond to it before your competition.

2. Big data concepts:

Big data includes data sets with size beyond the ability of common software tools. It is a set of techniques and technologies to reveal insight from data sets that are diverse, complex and of massive scale. In this section we deal with data generation, distributed systems and how big data is stored and managed.

2.1 Data generation:

Data generation is the first step of big data. Given Internet data as an example, huge amount of data in terms of searching entries, Internet forum posts, chatting records, and micro blog messages, are generated. Those data are closely related to people's daily life, and have similar features of high value and low density. Such Internet data may be valueless individually, but, through the exploitation of accumulated big data, useful information such as habits and hobbies of users can be identified, and it is even possible to forecast users' behaviours and emotional moods [6].

2.2 Distributed systems:

Distributed systems are used to store huge amount of data into different systems which are connected to each other. The memory available in a system is used to perform two major functions which are: *storing the data given by the user* and *performing different computational processes required by the system.* Only 30% of the total memory is used for storing data on a general machine. We cannot store data larger than this size on a single machine. Thus in order to reach our needs of storing big data we use distributed systems [6] [7].

If we connect many machines in order to store big data then each machine will provide 30% of its total memory which will result in sufficient space to perform our task. To use a distributed system to store massive data, the following factors should be taken into consideration:

- Consistency-Distributed systems work cooperativelyamong different servers. The data is divided into several pieces to be stored on different servers. Since the data is stored in different servers, the chances of its failure are very high. This may provide inconsistency among different copies of the same data. Consistency assures that multiple copy of the same data id identical.

- *Availability*- A distributed storage system operates in multiple sets of servers. As more servers are used, server failures are inevitable. It would be desirable if the entire system is not seriously affected to satisfy customer's requests in terms of reading and writing. This property is called availability.

- *Partition Tolerance*-Multiple servers in a distributed storage system are connected by a network. The network could have link/node failures or temporary congestion. The distributed system should have a certain level of tolerance to problems caused by network failures. It would be desirable that the distributed storage still works well when the network is partitioned.

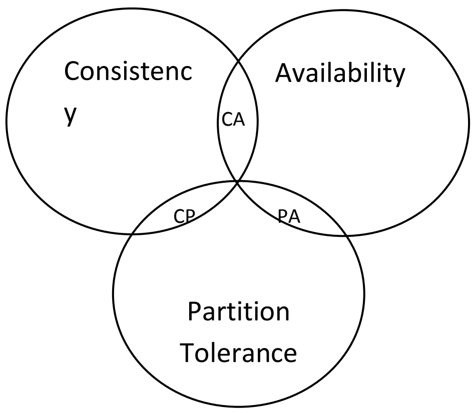

Fig 2.1: Different properties of a distributed system at the same time.

We cannot have all the three factors simultaneously in distributed systems. We can only have at most two factors simultaneously [6].

2.3 Hadoop framework:

Doug Cutting developed Hadoop with Mike Cafarella as the two worked on an open source Web crawler called Nutch, a project they started together in October 2002. In January 2006, Cutting started a sub-project by carving Hadoop code from Nutch. A few months later, in March 2006, Yahoo created its first Hadoop research cluster.

Hadoop framework is used for distributed processing of large data sets across clusters of commodity computers using a simple programming model. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

The two major componentsare:

*Hadoop distributed file system (HDFS)*is used to provide sufficient space to store data using distributed systems. It provides space required to store data and splits the data into small pieces and distributes it among different "nodes". It is natively redundant where the namenode tracks the location where the data is being stored.

MapReduce is common code which provides logic for putting data on different nodes. It splits a task across different processors and assembles the result. This is done with the help of Job Tracker and Task trackers. It has a high bandwidth of processing tasks.

2.3.1 Hadoop distributed file system(HDFS):

HDFS deals with the storage of big data using master/slave architecture [9]. This architecture consists of a NameNode which acts as the master who regulates data to different servers and stores their address and many DataNodes which act the slaves. DataNodes are used to store these data which are sent by the NameNode. There can be any number of DataNodes in a HDFS cluster depending upon the required amount of data input.

NameNode:

A NameNode stores the *Metadata*. Metadata contains name of the file, number of replicas, block and DataNode ids to which the data is stored. The NameNode acts like the brain to the whole HDFS cluster whose only job is to keep a track of where the data is stored. It also calculates the number of blocks required to store the data depending upon its size. It also makes the decision about the number of replicas of the block. It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode. We see its major components in detail below:

DataNode:

The DataNodes are responsible for serving read and write requests from the file system's clients. There are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode. It sends BlockReport continuously to NameNode stating its status.

Secondary NameNode (SNN):

Though it is quiet unusual for a NameNode to crash to crash, but if it does Secondary NameNode (SNN) comes into action. The Secondary NameNode gets the backup of all the information stored in the NameNode by visiting it in every one hour. If the NameNode crashes then SNN will have the latest information which would help in setting up the new NameNode with the required information about the Metadata and then the performance is carried out.

Working:

The HDFS cluster is communicates with the NameNode and informs it which file is to be saved. The NameNode then calculates the number of blocks required to store this file. The format in which data will be stored in NameNode is called *fsimage-byte image*[12]. The NameNode stores the Metadata (name, number of replications, block ids) of the file and send the data to the available DataNodes. The DataNodes in return send a feedback to the NameNode at constant intervals informing the NameNode about the BlockReport. The BlockReport consist of information whether the blocks in the DataNodes are free or busy. The BlockReport contains the list of all blocks present in the DataNode.The feedback signals are often called Heartbeats. This ensures the NameNode that the DataNodes are working properly.

The NameNode is constantly monitored and made sure that it doesn't overload or process data more than it can handle. If such a situation happens then the NameNode can suffer a crash. As we know that the NameNode acts as the brain of the HDFS cluster without it we cannot access or store data to DataNodes as the Metadata which stores the information of where our data is stored and BlockReport which gives us the block status is not available. A complete solution to this is still not available but Secondary NameNode helps us in retrieving the latest information stored in the NameNode. In Hadoop 1.0, the only disadvantage is that SNN stores a copy of the Metadata every hour but there is a chance of losing the data for the past one hour when the crash takes place. This drawback was over come in Hadoop 2.0.

If a DataNode goes down, we don't have a problem since another DataNode can do our task, but if NameNode goes down then our only option is to take data from Secondary NameNode and then proceed.

The Hadoop framework comes with some basic configurations as follows:

- One NameNode, One Secondary NameNode, any number of DataNodes
- Block size- 64mb
- Replications- 3
- Checkpoint (Secondary NameNode)- 1 hour
- Heartbeat (DataNodes)- 5 seconds

2.3.2 HadoopMapReduce:

Hadoop MapReduce is a framework for writing applications which can access huge amount of data. It provides the logical functioning of the Hadoop cluster. It tells how to store the data and how to operate on the data. With the help of JobTracker the TaskTrackers are managed[10]. The Hadoop MapReduce splits the task across processors. JobTracker are on the same level as the NameNode but are two different machines and the TaskTrackers are attached with the DataNodes. As the name suggests, the MapReduce algorithm contains two important tasks: Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). On the other hand, Reduce takes the output from a map as an input and combines the data tuples into smaller set of tuples. In MapReduce, the data is distributed over the cluster and processed. We see it major components in detail below:

JobTracker:

JobTracker is responsible for allocating different TaskTrackers with instructions about the operations that they need to perform. The JobTracker receives response from TaskTrackers in the same way as the DataNodes give to NameNode. This response tells the working status of the TaskTracker. Jobs are allocated to the DataNodes in which the required data is stored.

TakTracker:

The JobTracker instructs the TaskTrackers what operation is to be performed on the data. Every TaskTracker is configured with a set of *slots*; these indicate the number of tasks that it can accept. When the JobTracker tries to find somewhere to schedule a task within theMapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack.

Working:

A *jar file* is created with the set of logics which need to be performed on the data including the filename, read-write file specification and to do operations. This jar file is given to JobTracker which asks the NameNode about the location of the file. The NameNode provides the JobTracker with the actual file location with the DataNode and block id. It also specifies where the replicas of the original file are stored. The TaskTracker executes the given instruction and completes the job. Meanwhile the TaskTracker continuously sends *Heartbeats*to the JobTrackers stating its status and workload. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the TaskTracker as unreliable. This helps in improving the performance of the Hadoop framework with maximum efficiency.

2.3.3 Hardware implementation:

The below diagram shows the relationship between different parts of HDFS and MapReduce at different levels of Hadoop cluster architecture.

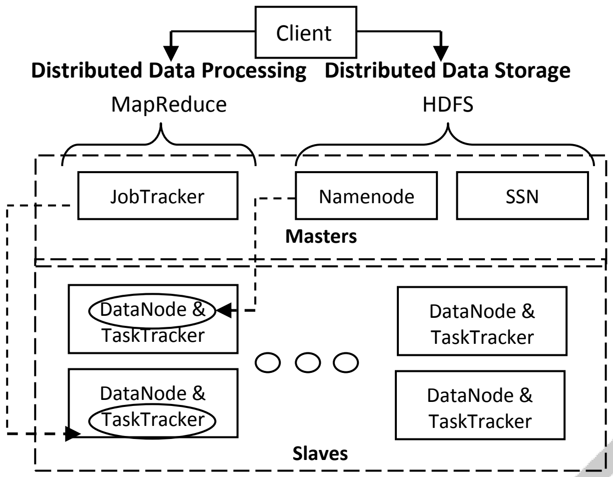

Fig 2.2: Hardware implementation of master/slave architecture

3. *Batch processing vs. Real-time processing:*

Batch data processing is an efficient way of processing when a group of transactions containing high volumes of data is collected over a period of time. In this processing technique data is collected, entered, processed and then the batch results are produced (Hadoop is focused on batch data processing). Batch processing requires separate programs for input, process and output. Whereas,real time data processing involves a continual input, process and output of data. In real time data processing data must be processed in a small time period (or near real time).In the current scenario many organizations use batch systems, but real time systems can also be used in some situations where immediate action needs to be taken within seconds or minutes[13].

Complex event processing (CEP)[14] combines data from multiple sources to detect patterns and attempt to identify either opportunities or threats. The goal is to identify significant events and respond fast.Operational Intelligence (OI)[15]uses real time data processing and CEP to gain insight into operations by running query analysis against live feeds and event data. OI is near real time analytics over operational data and provides visibility over many data sources. Here,the goal is to obtain near real time insight using continuous analytics to allow the organization to take immediate action.

In a Hadoop environment, the trick to providing near real time analysis is a scalable in-memory layer between Hadoop and CEP. Storm is an open source distributed real time computation system that processes streams of data. Storm can help with real time analytics, online machine learning, continuous computation, distributed RPC [16]and ETL[1]. Hadoop MapReduce processes "jobs" in batch while Storm processes streams in near real time. The idea is to reconcile real time and batch processing when dealing with large data sets. An example is detecting transaction fraud in near real time while incorporating data from the data warehouse or hadoop clusters.

4. Apache Spark:

Apache Spark is a framework for performing general data analytics on distributed computing cluster like Hadoop. It provides in memory computations for increase speed and data process over Hadoop MapReduce[17]. It runs on top of existing Hadoop cluster and access Hadoop data store (HDFS), can also process structured data in Hive and Streaming data from HDFS, Flume, Kafka, and Twitter. Apache Spark was written in Scala, whereas Hadoop MapReduce was written in Java. MapReduce processes data in batch mode which makes it impossible to process data in real time. Apache Spark executes the jobs in micro batches that are very short say approximately 5 seconds or less than that. Apache Spark has over the time been successful in providing more stability when compared to the real time stream oriented Hadoop Frameworks.

4.1 Why was Apache Spark introduced?

Hadoop MapReducewas envisioned at Google and successfully implemented. MapReduce satisfied the users in terms of processing the data. MapReduce has powerful logical implementation methods for performing all types of operations required by the client. However, there has always been a complain regarding the high latency problem with Hadoop MapReduce stating that the batch mode response for all the real time applications is highly painful when it comes to processing and analyzing data[17][18].

This drawback paved ways for the introduction of Apache Spark. Apache Spark is more powerful and flexible than Hadoop MapReduce. Despite the fact that it might not be possible to completely abandon Hadoop MapReduce from future allocations or existing applications, but there is scope for future applications to make use of this advance technology in future that comes with many more innovative features, to accomplish much more than that is possible with MapReduce Hadoop. Spark provides 100 times faster processing than Hadoop MapReduce as it stores data in memory and not on disk. On the other hand, Spark does RAM based computing.

4.2 Real-time big data analysisusing Apache Spark:

In real time, data comes at the rate of millions of events per second. Real-time big data analysis deals with processing this huge amount of data. The strength of Spark lies in its abilities to support streaming of data along with distributed processing. This is a useful combination that delivers near real-time processing of data. MapReduce is handicapped of such an advantage as it was designed to perform batch cum distributed processing on large amounts of data. Real-time data can still be processed on MapReduce but its speed is nowhere close to that of Spark[18][19].

Hadoop MapReduce	***Apache Spark***
Fast	*100x Fast than MapReduce*
Batch Processing	*Real-time Processing*
Stores data on disk	*Stores data on RAM*
Written in Java	*Written in Scala*

5. Conclusion:

Big data has transformed the data gathering and storing of large amount of information. It overcomes all the disadvantages faced by the relational database systems. We have discussed few of the analysis and things which big data is capable of doing. We have discussed the big data concepts regarding data generation and distributed systems which play and important role in Hadoop framework for handling big data.Hadoop framework is used to operate on big data using its 2 main parts: Hadoop Distributed File Systems (HDFS) and Hadoop MapReduce. HDFS provides storage space whereas MapReduce takes care of the processing part.We also see how batch processing and real-time processing systems perform their tasks. Hadoop framework uses batch processing which provides high latency rate.Thus Apache Spark wasdesigned based on real-time processing. This provided a better processing rate and efficiency than Hadoop MapReduce. Apache Spark consists of various machine learning techniques and executes the jobs in micro batches during real time. This provides a more stable orientation compared to real time stream oriented Hadoop Frameworks. Spark runs on top of Hadoop cluster which gives it an advantage as many companies or organizations already have hadoop installed, so replacing the whole framework will increase the cost.

6. References:

[1.] Overview of Extraction, Transformation, and Loading https://docs.oracle.com/cd/B19306_01/server.102/b14223/ettover.htm

[2.] Big Data – What is Big Data – 3 V's of Big Data – Volume, Velocity and Variety http://blog.sqlauthority.com/2013/10/02/big-data-what-is-big-data-3-vs-of-big-data-volume-velocity-and-variety-day-2-of-21/

[3.] Support Vector Machines (SVM) Introductory Overviewhttp://www.statsoft.com/textbook/support-vector-machines

[4.] 4 Things Big Data Can Do, and 3 Things It Can't Doby Bernard Marhttp://data-informed.com/4-things-big-data-can-do-and-3-things-it-cant-do/

[5.]Nobody Tells You – 5 things Big Data 'CAN' and 'Cannot' Dohttp://www.analyticsvidhya.com/blog/2015/11/5-big-data-can-cannot/

[6.] Big Data: A Survey http://link.springer.com/article/10.1007/s11036-013-0489-0/fulltext.html

[7.]Distributed systems http://book.mixu.net/distsys/single-page.html

[8.] ACM SIGMOD Blog http://wp.sigmod.org/?cat=11

[9.] Hadoop architecture: https://hadooptutorial.wikispaces.com/Hadoop+architecture

[10.]MapReduce : https://hadooptutorial.wikispaces.com/MapReduce

[11.]Exascale computing and big data :http://cacm.acm.org/magazines/2015/7/188732-exascale-computing-and-big-data/fulltext

[12.] Understanding HDFS cluster fsimage image and looking its content https://cloudcelebrity.wordpress.com/2013/03/20/understanding-hdfs-cluster-fsimage-image-and-looking-its-contents/

[13.]Data science central http://www.datasciencecentral.com/profiles/blogs/batch-vs-real-time-data-processing

[14.]How to Use Complex Event Processing for Big Data By Judith Hurwitz, Alan Nugent, Fern Halper, and Marcia Kaufmanhttp://www.dummies.com/how-to/content/how-to-use-complex-event-processing-for-big-data.html

[15.]Operational Intelligence: Real-Time Business Analytics for Big Data https://tdwi.org/webcasts/2012/08/operational-intelligence-realtime-business-analytics-for-big-data.aspx?tc=page0

[16.] Remote Procedure Calls(RPC) https://www.cs.cf.ac.uk/Dave/C/node33.html

[17.] Big data processing with Apache Spark http://www.infoq.com/articles/apache-spark-introduction

[18.] Is Apache Spark going to replace Hadoop? http://aptuz.com/blog/is-apache-spark-going-to-replace-hadoop/

[19.] Introduction to Apache Spark https://www.toptal.com/spark/introduction-to-apache-spark