Simple | Smart | Speed
CHOOLS

Analysis of Big Data using Apache Spark

Abstract- Data analysis is concerned with the automatic extraction of data related information from variety of sources. Although most analytical model addresses commercial tasks, such as product reviews, there is increasing interest in the affective dimension of the social media websites and various other sources. The current analytical models are not ideally suited for real time analysis of data. This model will collect data from sources of structured and un-structured data; it will filter the relevant data from the raw data in real time or stored data and make it useful for analysis and process it. Hence, such model will be successful in the sense of performing real time analysis than the current ones.

Keywords- Big Data; Large Dataset; Hadoop; HDFS; Spark; SparkSQL; RDD; Python; Java

.

1. INTRODUCTION

Now a days the large data sets are produced by the social media data, transport data, black box data, stock exchange data and many more and the traditional way is not that efficient to handle such a large data set[1]. The big data consist of the higher velocity, volume and diverse variety of data. Web utilization is being developed with the advancement in the technologies by which we can handle large data sets be it the information and views of the particular product on facebook or twitter or the performance information of the aircrafts [2]. The data can be present in three forms like the structured data which includes relational data in it, semi structured data like XML data and the unstructured data in which the word, pdf, text, media are included [3]. There are four organizations of big data volume, velocity, variety and values. To manage the big data instruments like Hadoop and Spark can be used. Hadoop can be used to handle the large and complex data sets by storing it and running the complex queries in very efficient way [4]. It is the distributed technologies that shares the work to various servers that is the large job is splited to small tasks and these tasks are run concurrently, thus Hadoop is the massively parallel processor. The main pattern in the Hadoop includes the Map Reduce which is written in Java [5]. But there are some limitations in Hadoop like it does not give the real time processing of the data, the SQL support is limited, insufficient execution and many more[6].

Thus to overcome such limitations of Hadoop, Spark can be used which is more efficient than Hadoop. Nowadays Spark is in race at it can process large amount of the data with very low latency processing that the map reduce of Hadoop is not able to do. Spark cab be preferred over Hadoop as its speed is 100 times more than that of map reduction used in the Hadoop [7]. It also reduces the time which is being consumed for reading and writing. Today Amazon, eBay, Yahoo and the Netflix has adopted spark. Thus the spark is the open source of the big data processing framework. The working of the spark is with the file system which distributes the data across the entire cluster and the data is processed simultaneously [8]. In the Hadoop the MapReduce was coded with the java whereas here it is coded not only by java but also the Scala and python[9]. It is the advancement of the mapreduce that was used in the Hadoop. Spark basically works with the combination of the SQL, complex analytics and streaming. It combines the libraries for SQL and data frames and stores the information rather than writing it[10].

2. BIG DATA

Big Data refers to large data that can be analysed computationally in order to reveal patterns, trends and association, relating to human behaviour and interaction.

These big data can't be handled with traditional computing techniques as they mostly use single machine to process data [1][11].

It portrays data having high volume, velocity and variety which requires new technologies and system to catch, store and analyse it. It can be described by "3V's".

- Volume - In order to analyse, data is collected from variety of sources business transitions, social media, and information from sensor or from machine data.
- Velocity- The flow of data from various sources is unpredictable, so it makes a challenge for the tradition approach to handle those kinds of data. In order

resolve those challenge we use big data approach to handle those datasets.

- Variety – The data to be analysed comes in all type of formats. It can be real time, predictive, streaming, mission critical. The data can also be structured, numeric in traditional databases or unstructured text documents email, video, stock ticker data and financial transactions [12].

Two other dimensions defining big data-

- Variability- The dataflow can be highly inconsistent with periodic trends because of increase in velocity and varieties of data. This makes data challenging to manage. The challenge is predominantly experienced while analysing unstructured data.
- Complexity – Data collection is done from multiple sources because of which it is difficult to link, cleanse, match and transform data across systems. It is important correlate and connect with the usefulness of the data[13].

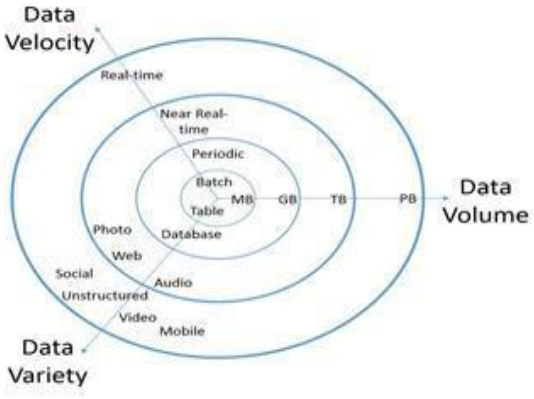

Figure 1: Area of Big Data

3. *HADOOP*

Hadoop is the analytics that is used for handling the large data sets. It is the platform of the big data that basically stores the large content of data in an efficient way in terms of reliability and flexibility also Hadoop runs the complex queries over the data very efficiently. It is the free open source analytical process of Apache. Scaling is its key feature like if your data increases then we need to expand our clusters by just adding the servers and the same query which is being designed can work over thousands of nodes of the cluster. The ecosystem of Hadoop includes the map reduction, SQLs on Hadoop, real time big data and also the new toolkit Apache Spark [14].

3.1 Architecture of Hadoop

Hadoop shares it work among many servers thus it is a distributed technology. The cluster of the Hadoop is the master architecture and it knows where the data among the worker that is the nodes of the cluster is distributed[15]. The cluster coordinates the queries, splitting it into various or multiple tasks among the nodes of the cluster. This node stores and executes the tasks distributed to them by the cluster. For example if we take into the consideration of the large data file which of 1GB then in Hadoop the file is splited in various pieces called as blocks and each block would replicate to have 3 copies of itself. Thus the 1GB file is now splited into 24 block of 128MB. Thus it is the ability of the Hadoop to split the data into multiple small tasks and these small tasks runs parellely to do the processing of the large data set. This is called massively paralle processing [16].

Hadoop's ability to split a very large job into many small tasks while those tasks run concurrently is what makes it so powerful—that's called Massively Parallel Processing (MPP), and it is the basis for querying huge amounts of data and getting the response in a reasonable time[17] [9] [4]. In order to see how much benefit you get from high concurrency, consider a more realistic example—a query of over 1 TB of data that is split into 256 MB blocks (the block size is one of the many options you can configure in Hadoop). If Hadoop splits that query into 4,096 tasks and the tasks take about 90 seconds each,the query would take more than 100 hours to complete on a single-node machine[18].

Fig. 2. Architecture of Hadoop

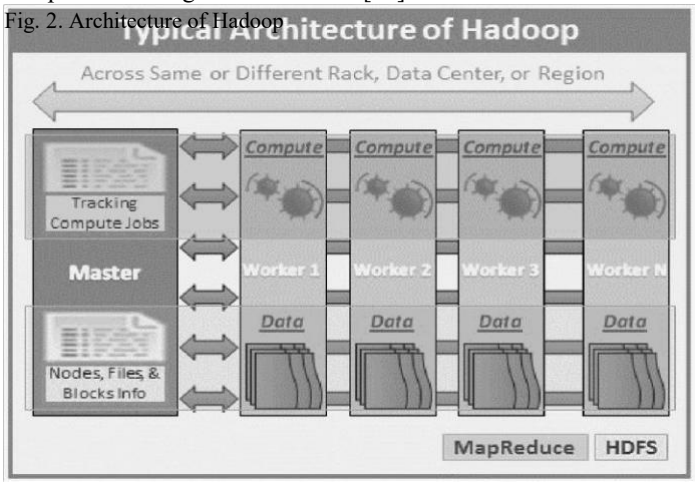

Figure 2. Architecture of Hadoop

On a powerful 12-node cluster, the same query would complete in 38 minutes. Table 1 shows how those tasks could be scheduled on different cluster sizes.

TABLE 1. COMPUTE TIME FOR HADOOP CLUSTER

Data Nodes	Total CPU Cores	Total Concurrent tasks	Best Compute Time
1	1	1	100 hrs
12	192	160	38 mins
50	1200	1000	6 mins

Hadoop aims for maximum utilization of the cluster when scheduling jobs. Each task is allocated to a single CPU core, which means that allowing for some processor overhead, a Analysis of Big Data using Apache Spark

cluster with 12 data nodes, each with 16 cores, can run 160 tasks concurrently [19].

A powerful 50-node cluster can run more than 1000 tasks concurrently, completing our 100-hour query in under six minutes. The more nodes you add, however, the more likely it is that a node will be allocated a task for which it does not locally store the data, which means the average task completion time will be longer, as nodes read data from other nodes across the network[20][29].

4. HDFS

HDFS is the storage part of the Hadoop platform, and with it you get reliability and scale with commodity hardware.

Commodity is important—it means you don't need specialist hardware in your Hadoop cluster, and nodes don't need to have the same—or even similar—specifications. Many Hadoop installations have started with a cluster built from beg-or-borrow servers and expanded with their own higher spec machines when the project took off. You can add new nodes to a running cluster while increasing your storage and compute power without any downtime [5][21].

The resilience built into HDFS means servers can go offline or disks can fail without loss of data, so that you don't even need RAID storage on your machines. And because Hadoop is much more infrastructure-aware than other compute platforms, you can configure Hadoop to know which rack each server node is on. It uses that knowledge to increase redundancy—by default, data stored in Hadoop is replicated three times in the cluster. On a large cluster with sufficient capacity, Hadoop will ensure one of the replicas is on a different rack from the others [19][22].

As far as storage is concerned, HDFS is closed system. When you read or write data in Hadoop, you must do it through the HDFS interface—because of its unique architecture, there's no support for connecting directly to a data node and reading files from its disk. The data is distributed among many data nodes, but the index specifying which file is on which node is stored centrally[4][23]. In this chapter, we'll get a better understanding of the architecture and see how to work with files in HDFS using the command line.

4.1 Architecture of HDFS

HDFS uses a master/slave architecture in which the master is called the “name node” and the slaves are called “data nodes.” Whenever you access data in HDFS, you do so via the name node, which owns the HDFS-equivalent of a file allocation table, called the file system namespace.

In order to write a file in HDFS, you make a PUT call to the name node and it will determine how and where the data will be stored. To read data, you make a GET call to the name node, and it will determine which data nodes get copies of the data and will direct you to read the data from those nodes [2][24].

The name node is a logical single point of failure for Hadoop. If the name node is unavailable, you can't access any of the data in HDFS. If the name node is irretrievably lost, your Hadoop journey could be at an end—you'll have a set of data nodes containing vast quantities of data but no name node capable of mapping where the data is, which means it might be impossible to get the cluster operational again and restore data access.

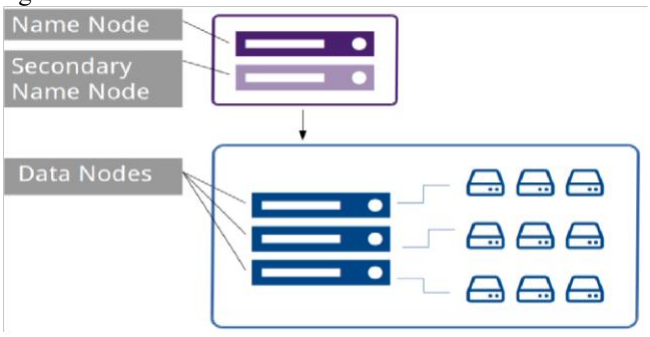

Figure 3. HDFS architecture

In order to prevent that, Hadoop clusters have a secondary name node that has a replicated file index from the primary name node. The secondary is a passive node—if the primary fails, you’ll need to manually switch to the secondary, which can take tens of minutes. For heavily used clusters in which that downtime is not acceptable, you can also configure the name nodes in a high-availability setup.

5. APACHE SPARK

Apache Spark designed for fast computation. It is an advanced computing technology based on Hadoop MapReduce and it also extends the Map Reduce model in oder to use it more efficiently for variety of computations and streaming process. It helps in memory based cluster computing which results in increase of the processing speed of an application[25][27].

The aim of developing Spark was to cover a wide range of data processing workloads such as batch processing, iterative algorithms, interactive queries and streaming of data. Other than supporting data processing in a respective system, it also reduces the heavy load of maintaining a separate tool for management [3][28].

The major features of spark are speed, support for multiple language and advanced analytics.

6. REVIEW

In the past few years there are various analysis process that have been introduced in order to process the large amount of data generated by the various sectors. Hadoop that uses the map-reduce model for processing such a large data but is not as efficient as spark. There are certain limitation in

the Hadoop that Spark has overcome. In this article we would be discussing how spark process such a large data and how it is more efficient than the Hadoop. Spark is the computing framework that is much faster than the Hadoop. Its speed for running the application is 40 times faster than the Hadoop. Although the Hadoop uses the map-reduce model but is unable to work efficiently as it cannot process the rich application in single pass. Users wants the iterative algorithms which the part of the complex multi pass algorithm for graph processing[30].

Now the problem is that the multi pass complex and the interactive applications needs to distribute the data and the only way to do it is to share the data between the parallel operations in the map reduce is to write the distributed file system which leads to overhead due to data replication. This problem is minimized by spark by providing primitive storage, the resilient distributed dataset RDDs.

The RDDs is more efficient than the distributed file system it stores the data in the memory over the entire queries ant it provides the fault tolerance without the replication. The RDDs runs 40 times faster than the distributed file system in the Hadoop.

For the better comparison of spark over hadoop we can consider an example, if we take 100GB data say on 50 node cluster hadoop takes around 110 second per iteration where as spark takes only 80 seconds for the iteration to load the data [20][26].

6.1 Comparison of Hadoop and Spark

It has been proved in many research papers or studies that Spark is more faster than Hadoop whether it be any operation. For analysing big data spark is better than Hadoop. Big data is basically a large dataset and these datasets can have any type of information, operations, algorithms and etc. It has been proved that spark is better than Hadoop for iterative operations.

A research paper was published on November 2013 by Lei Gu and Huan Li, In which they conducted a series of exhaustive experiments to evaluate Hadoop and Spark. They chose five real graph datasets and then generated five synthetic graph dataset to do the comparison. The following are the ten graph datasets given in the table below.

Name	File Size	Nodes	Edges	Description
wiki-Vote	1.0 MB	7,115	103,689	Wikipedia who-votes-on-whom network
soc-Slashdot0902	10.8 MB	82,168	948,464	Slashdot social network from February 2009
web-Google	71.9 MB	875,713	5,105,039	Web graph from Google
cit-Patents	267.5 MB	3,774,768	16,518,948	Citation network among US Patents
Twitter	1.3 GB	11,316,811	85,331,845	Twitter social network on who follows whom network
kronecker19	40 MB	416,962	3,206,497	Synthetic graph generated by Kronecker generator with 19 iterations
kronecker20	89MB	833,566	7,054,294	Synthetic graph generated by Kronecker generator with 20 iterations
kronecker21	209MB	1,665,554	15,519,448	Synthetic graph generated by Kronecker generator with 21 iterations
kronecker22	479MB	3,330,326	34,142,787	Synthetic graph generated by Kronecker generator with 22 iterations
kronecker23	1.1GB	6,654,956	75,114,153	Synthetic graph generated by Kronecker generator with 23 iterations

Figure 4. Graph dataset for experiments

According to the results, Spark has outperformed Hadoop in running time comparison according to the size of the dataset.

The following is the graph obtained after evaluating the result.

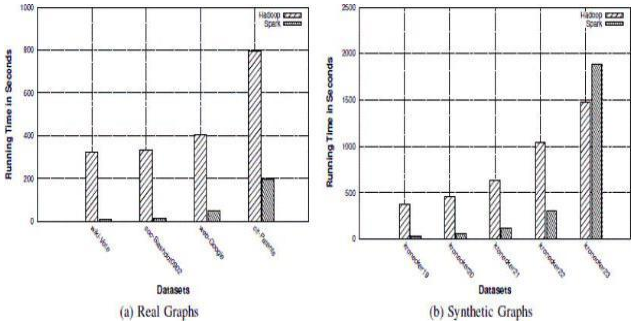

Figure 5. Running Time Comparison of Hadoop and Spark

On the other hand, after evaluating the results it came out the Spark is extremely memory consuming as compared to Hadoop.

The following graph is memory usage comparison by Spark and Hadoop using real graph datasets.

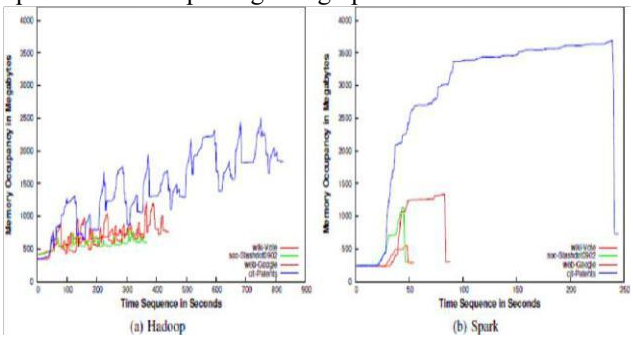

Figure 6. Memory usage using real graph datasets

And Fig. 8 shows the memory usage comparison by Spark and Hadoop using synthetic graph datasets. In conclusion, we get that Spark is less time consuming and is a framework to overcome the demerits in Hadoop but is memory consuming as compared to Hadoop framework.

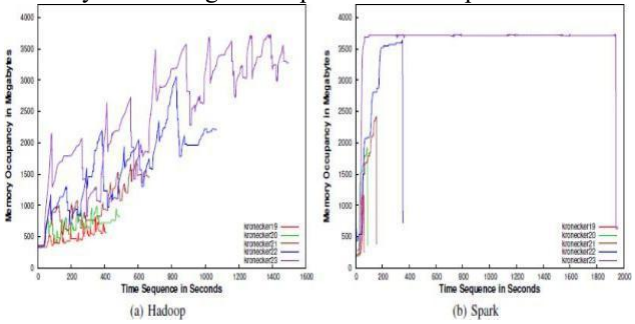

Figure 7. Memory usage using synthetic graph datasets

7. METHODOLOGY

7.1 Structure of Spark

- **Apache Spark Core**- It is general execution engine for spark platform on which each one of the alternative functionality is built upon. It helps In-Memory computing and referencing datasets in memory device systems.
- **Spark SQL**-Spark SQL is a part on top of Spark Core that introduces an alternative information abstraction known as SchemaRDD, that provides support for structured and semi-structured information.
- **Spark Streaming**-Spark Streaming leverages Spark Core's quick programing capability to perform streaming analytics. It ingests information in minibatches and also performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of information[8][2].

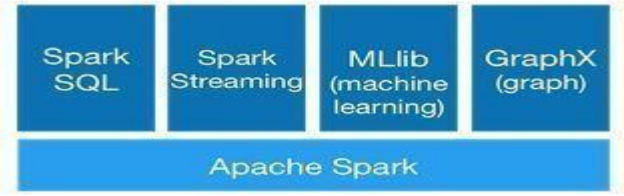

Figure 8. Components of Apache Spark

MLlib (Machine Learning Library)- A distributed machine learning framework on top of Spark as a result of the distributed memory-based Spark design. Spark MLlib is 9 times as faster in comparison Hadoop disk-based version of Apache mahout (before mahout gained a Spark interface).
GraphX- A distributed graph-processing framework on top of Spark which provides an inbuilt API for expressing graph computation which will model the user-defined graphs by using the Pregel abstraction API. It additionally provides an optimized runtime for this abstraction.

In this part we are working on the steps to get our system ready for collecting and analysing the data.

- **Data Collection-** Data collection is an important aspect of any type of analysis. Large data from various sources are collected. The next step in information handling flow is storing the data.
  Our collected data is uploaded to HDFS and is stored so that this data can be used to process. By storing the large datasets into HDFS, the solution provided is much more efficient, reliable, economical, and scalable.
- **Filtering Data-** Filtering data is one of the important aspects. Raw collected data from variety sources is filtered according to the needs of the analysis. The Hadoop administrator will put all those data in one file that will be imported in Hadoop HDFS for a filtration.
- **Processing Data-** Now the filtered data is used in order to achieve the desired results using various algorithms or procedures. We can use SparkSQL , Java, Python, Scala or R for processing the data.
- **Visualisation-** After getting the desired outcome from the processing of big data. We now use that data to plot graphs and charts so that it is easier for us to determine the outcome of the processing

8. RESULT AND DISCUSSION

In this paper we have found that after using Spark to process data we comparatively increased the processing speed with respect to Hadoop. Also, we have found that it consumes more amount of RAM is compared to Hadoop and less amount of disk usage in terms of read/write.The developed methodology can be used in real time processing of data using data streams and where there is huge amount of inflow of data so that it can perform real time analytics.Figure 9 shows the steps involved in this methodology.

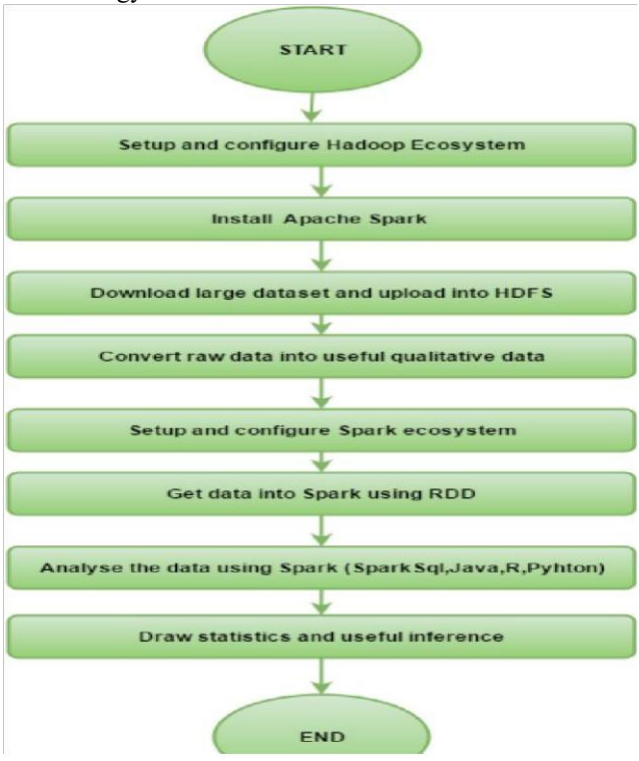

Figure 9. Steps in Analysing Data in Spark and Hadoop

9. CONCLUSION AND FUTURE SCOPE

In this research article, we described the ongoing resea rch to design and implement big data with Hadoop and Spark. This methodology can analyse data in real time and where there is an immense flow of data into the system.
In future we have to implement this methodology using Spark and Hadoop so that applications of real time analytics can be developed with ease.

ACKNOWLEDGEMENT

We are appreciative to my friends, family members and Mr. Ankur Saxena of Amity University at Noida. We thank our fellow students who provided their insight and expertise that really motivated and guided us during this research. We thank them for assisting us particularly under techniques and methodology section. We would also like to special thanks to our team for sharing their expertise and knowledge.