Data Analysis Using Hadoop

Bhawna Makhija[1] Chhaya Amrute[2] Durga Nandan Mishra[3] Rashmi Tembhurne[4]
Vijay V. Chakole[5]
[1,2,3,4,5]Department of Electronics Engineering
[1,2,3,4,5]K.D.K.C.E ,Nagpur, Maharashtra, India

Abstract— The rapid growth of Internet and WWW has increases to vast amounts of information available online. Big data is an evolving term that describes an voluminous amount of structured, semistructured and understructed data that has the potential to be mined for information. For this nowadays various Data intensive technologies (Map Reduce) are used which uses computer applications, which requires large volumes of data and most of their processing time to I/O and manipulation of data .In order to store, manage, access, and process vast amount of data available online and the data that is created in structured and unstructured form, Data intensive computing is needed which satisfies the need to search, analyze, mine, and visualize the large amount of data and information. We will analysis already available Data intensive technologies with Hadoop Data intensive to provide high performance, should be fault resilient over hardware failures, communications errors, and software bugs and executing a variety of data intensive analysis benchmarks.
Key words: Big Data, HDFS, Map Reduced, Cluster

I. INTRODUCTION

We are living in an age when an explosive amount of data is being generated every day. Data from sensors, mobile devices, social networking websites, scientific data & enterprises – all are contributing to this huge explosion in data. This sudden bombardment can be grasped by the fact that we have created a vast volume of data in the last two years. Big Data- as these large chunks of data is generally called- has become one of the hottest research trends today.

Research suggests that tapping the potential of this data can benefit businesses, scientific disciplines and the public sector – contributing to their economic gains as well as development in every sphere. The need is to develop efficient systems that can exploit this potential to the maximum, keeping in mind the current challenges associated with its analysis, structure, scale, timeliness and privacy. There has been a shift in the architecture of data-processing systems today, from the centralized architecture to the distributed architecture. Enterprises face the challenge of processing these huge chunks of data, and have found that none of the existing centralized architectures can efficiently handle this huge volume of data. These are thus utilizing distributed architectures to harness this data. Several solutions to the Big

Data problem have emerged which includes the Map Reduce environment championed by Google which is now available open-source in Hadoop. Hadoop‟s distributed processing, Map Reduce algorithms and overall architecture are a major step towards achieving the promised benefits of Big Data.

Map Reduce & Hadoop are the most widely used models used today for Big Data processing. Hadoop is an open source large-scale data processing framework that supports distributed processing of large chunks of data using simple programming models. The Apache Hadoop project consists of the HDFS and Hadoop Map Reduce in addition to other modules. The software is modelled to harvest upon the processing power of clustered computing while managing failures at node level. The Map Reduce software framework which was originally introduced by Google in 2004 is a programming model, which now adopted by Apache Hadoop, consists of splitting the large chunks of data, and “Map” & “Reduce” phases. The Map Reduce framework handles task scheduling, monitoring and failures.

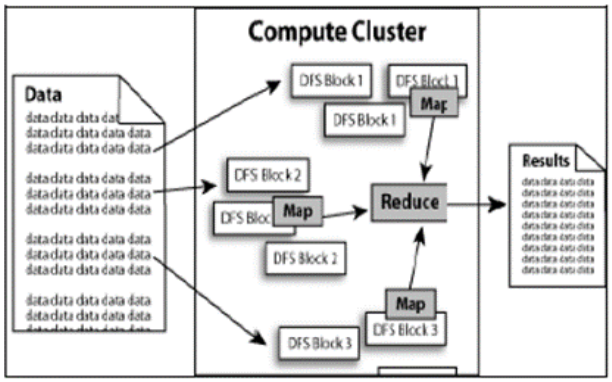

Fig. 1: Map Reduce in Hadoop

II. AIM AND OBJECTIVE

1) Implement different clustering method on Hadoop platform.
2) Hadoop is good at processing large amount of data in parallel. The idea is to breakdown the large input into smaller clusters and each can be processed separately on different machines. That way, we can alleviate the I/O bottleneck across many machines to achieve better overall performance.
3) The already available data intensive technologies can be analyzed with Hadoop data intensive to provide :
   1. high performance
   2. fault resilience over hardware failures
   3. reduce communications errors and software bugs and
   4. execute a variety of data intensive analysis.

Big Data has come up because we are living in society that uses the intensive use of increasing data technology. As there exist large amount of data, the various challenges are faced about the management of such extensive data .The challenges include the unstructured data, real time analytics, fault tolerance, processing and storage of the data and many more.

The size of the data is growing day by day with the exponential growth of the enterprises. For the purpose of decision making in an organizations, the need of processing and analyses of large volume of data is increases.

The various operations are used for the data processing that includes the culling, tagging, highlighting,

searching, indexing etc. Data is generated from the many sources in the form of structured as well as unstructured form [20].

Big data sizes vary from a few dozen terabytes to many petabytes of data. The processing and analysis of large amount of data or producing the valuable information is the challenging task.

As the Big data is the latest technology that can be beneficial for the business organizations, so it is necessary that various issues and challenges associated with this technology should bring out into light. The two main problems regarding big data are the storage capacity and the processing of the data.

III. PROPOSED WORK

A. Data Gathering

For performing the big data experiments, setup of Hadoop data cluster comprising of four nodes and Hadoop Distributed File System (HDFS) for storage was used. Before moving to multi-node cluster, single node cluster was first configured and tested. Hadoop has too many configuration parameters to describe here, but the most relevant for the purpose of this evaluation is the number of concurrent Map and Reduce tasks that are allowed to run on each node. We configured our cluster to run eight concurrent tasks per server. Each Map/Reduce program that is run is partitioned into M map tasks and R reduce tasks. Input and output data for the Map/Reduce programs is stored in HDFS, while input and output data for the data-parallel stack-based implementation is stored directly on the local disks. One node was configured as Master node and other nodes were designated as slave nodes. The master node runs the "master" daemons: Name Node for the HDFS storage layer and Job Tracker for the Map Reduce processing layer. The slave nodes run the "slave" daemons: Data Node for the HDFS layer and Task Tracker for Map Reduce processing layer. The master node was also used as slave node to increase the processing nodes.

B. Data Processing

MapReduce is a data processing model. Its greatest advantage is the easy scaling of data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers.

In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over [9].

Before developing the MapReduce framework, Google used hundreds of separate implementations to process and compute large datasets. Most of the computations were relatively simple, but the input data was often very large.

Hence the computations needed to be distributed across hundreds of computers in order to finish calculations in a reasonable time. MapReduce is highly efficient and scalable, and thus can be used to process huge datasets. When the MapReduce framework was introduced, Google completely rewrote its web search indexing system to use the new programming model. The indexing system produces the data structures used by Google web search. There is more than 20 Terabytes of input data for this operation. At first the indexing system ran as a sequence of eight MapReduce perations, but several new phases have been added since then. Overall, an average of hundred thousand MapReduce jobs is run daily on Google's clusters, processing more than twenty Petabytes of data every day.

The idea of MapReduce is to hide the complex details of parallelization, fault tolerance, data distribution and load balancing in a simple library. In addition to the computational problem, the programmer only needs to define parameters for controlling data distribution and parallelism.

Like Google's MapReduce, Hadoop uses many machines in a cluster to distribute data processing. The parallelization doesn't necessarily have to be performed over many machines in a network. There are different implementations of MapReduce for parallelizing computing in different environments. Hadoop is a distributed file system that can run on clusters ranging from a single computer up to many thousands of computers. Hadoop was inspired by two systems from Google, MapReduce and Google File System.

1) Instrument Calibration

When a MapReduce program is run by Hadoop, the job is sent to a master node, the jobtracker, which has multiple "slave" nodes, or tasktrackers that report to it and ask for new work whenever they are idle. Using this process, the jobtracker divides the map tasks (and quite often the reduce tasks as well) amongst the tasktrackers, so that they all work in parallel. Also, the jobtracker keeps track of which tasktrackers fail, so their tasks are redistributed to other task trackers, only causing a slight increase in execution time.

Furthermore, in case of slower workers slowing down the whole cluster, any tasks still running once there are no more new tasks left are given to machines that have finished their tasks already. Not every process nodes have a small piece of a larger file, so that when a file is accessed, the bandwidth of a large number of hard disks is able to be utilized in parallel.

In this way, the performance of Hadoop may be able to be improved by having the I/O of nodes work more concurrently, providing more throughput.

Map Reduce works in the following manner in below 7 tasks:

1) The Map-Reduce library in the user program first splits the input into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece. It then starts up many copies of the program on a cluster of machines.(Refer Figure 5)
2) One of the copies of the program is special- the master copy. The rest are workers that are assigned work by the master. There are M map task and R reduce tasks to assign; the master picks idle workers and assign each one a task
3) A worker who is assigned a map task reads the contents of the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.
4) Periodically, the buffered pairs are written to local disk partitioned into R regions by the partitioning function. The location of these buffered pairs on the local disk are

passed back to the master, who is responsible for forwarding these locations to the reduce workers

5) When a reduce worker is modified by the master about these locations.it uses remote procedure calls to read buffered data from the local disk of map workers. When a reduce worker has read all intermediate data, it sorts it by the many different key map to the same reduce task.
6) The reduce worker iterate over the sorted intermediate data and for each unique key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to the final output file for this reduce partition
7) When all map task and reduce task have been completed, the master wakes up the user program. At this point, the Map-Reduce call in the user program returns back to the user code.

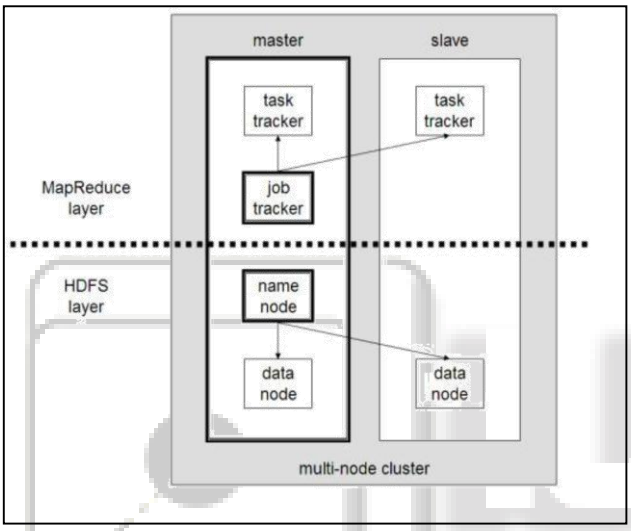

Fig. 2: Hadoop Architecture

IV. MAPREDUCE AND HADOOP

Data is conceptually record-oriented in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records. The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system.

Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of moving computation to the data, instead of moving thedata to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

A. K Means Clustering – Pseudo code

K-Means is a simple learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of n entities in k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized

The k-means algorithm is used for partitioning where each cluster's centre is represented by the mean value of the objects in the cluster

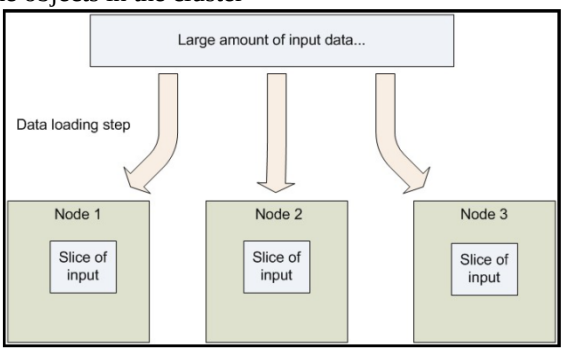

Fig. 3: K Means Clustering

V. Flowchart

The HDFS stores the input data as it can store huge chunks of data. HDFS also helps in data localizations for the map/reduce task.

The input data is divided into parts and allotted to the mapper. The output from the mapper is key-value pair. The key is itemset and value is support count then output is passed the combiner. It combines all the count value related to a particular item which is known as key. The result from this is taken in by the reducer which combines and sum up of the values corresponding to an item.

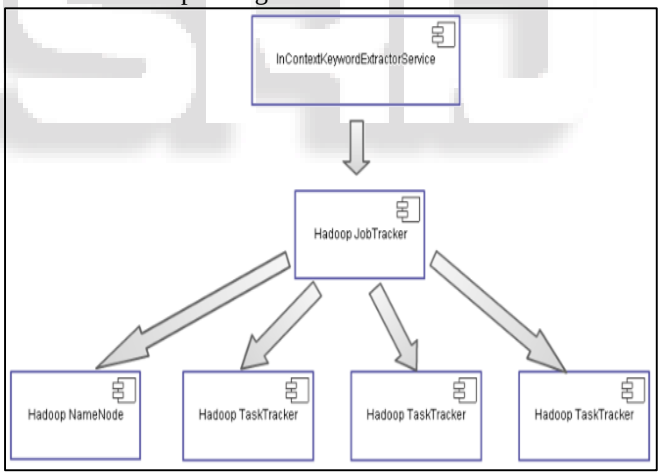

Fig. 4:

VI. CONCLUSION

The paper comes up with a DBSCAN algorithm based on the MapReduce, which faces in scalabily problems when dealing with large data sets. Large data sets are cut into small set of data , and it processes the small set of data parallelly. Experimental results show that the DBSCAN algorithm based on the MapReduce alleviated the problem of time delay caused by large data sets and have good timeliness. The next task is to study how to solve the problem of mining efficiency of dynamic massive database furtherly. And applications of DBSCAN algorithm based on MapReduce in other fields.

As data clustering has attracted a significant amount of research attention, many clustering algorithms have been

proposed in the past decades. However, the enlarging data in applications makes clustering of very large scale of data a challenging task. In this paper, we propose a fast parallel k-means clustering algorithm based on MapReduce, which has been widely embraced by both academia and industry. We use speedup, scaleup and sizeup to evaluate the performances of our proposed algorithm. The results show that the proposed algorithm can process large datasets on commodity hardware effectively.

Future work will focus on performance evaluation and modeling of hadoop data-intensive applications on cloud platforms like Amazon Elastic Compute Cloud (EC2).

REFERENCES

[1] Grosso, P. ; de Laat, C. ; Membrey, P.,(20-24 May 2013)," Addressing big data issues in Scientific Data Infrastructure"
[2] Kogge, P.M.,(20-24 May,2013), "Big data, deep data, and the effect of system architectures on performance"
[3] Sagiroglu, S.; Sinanc, D. ,(20-24 May 2013),"Big Data: A Review"
[4] Garlasu, D.; Sandulescu, V. ; Halcu, I. ; Neculoiu, G. ;,(17-19 Jan. 2013),"A Big Data implementation based on Grid Computing", Grid Zhang, Du,(16-18 July,2013)," Inconsistencies in big data"
[5] Computing. Szczuka, Marcin,(24-28 June,2013)," How deep data becomes big data.
[6] Szczuka, Marcin,(24-28 June,2013)," How deep data becomes big data"
[7] Bakshi, K.,(2012)," Considerations for big data: Architecture and approach"
[8] Mukherjee, A.; Datta, J.; Jorapur, R.; Singhvi, R.; Haloi, S.; Akram, W., (18-22 Dec.,2012) , "Shared disk big data analytics with Apache Hadoop"
[9] Aditya B. Patel, Manashvi Birla, Ushma Nair ,(6-8 Dec. 2012),"Addressing Big Data Problem Using Hadoop and Map Reduce"
[10] Yu Li ; Wenming Qiu ; Awada, U. ; Keqiu Li,,(Dec 2012)," Big Data Processing in Cloud Computing Environments"
[11] Tien, J.M.(17-19 July,2013)," Big Data: Unleashing information"

