Volume 4, Issue 5, May 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis of Bidgata using Apache Hadoop and Map Reduce

Mrigank Mridul, Akashdeep Khajuria, Snehasish Dutta, Kumar N
Dept of CSE,EWIT,VTU
India

Prasad.M.R
Asst.Prof, CSE Dept, EWIT
Indoia

Abstract— In this electronic age, increasing number of organizations are facing the problem of explosion of data and the size of the databases used in today's enterprises has been growing at exponential rates. Data is generated through many sources like business processes, transactions, social networking sites, web servers, etc. and remains in structured as well as unstructured form [2]. Today's business applications are having enterprise features like large scale, data-intensive, web-oriented and accessed from diverse devices including mobile devices. Processing or analyzing the huge amount of data or extracting meaningful information is a challenging task. The term "Big data" is used for large data sets whose size is beyond the ability of commonly used software tools to capture, manage, and process the data within a tolerable elapsed time. Big data sizes are a constantly moving target currently ranging from a few dozen terabytes to many peta bytes of data in a single data set. Difficulties include capture, storage, search, sharing, analytics and visualizing. Typical examples of big data found in current scenario includes web logs, RFID generated data, sensor networks, satellite and geo-spatial data, social data from social networks, Internet text and documents, Internet search indexing, call detail records, astronomy, atmospheric science, genomics, biogeochemical, biological, and other complex and/or interdisciplinary scientific research, military. Surveillance, medical records, photography archives, video archives, and large-scale ecommerce.

Keywords— Big Data Problem, Hadoop cluster, Hadoop Distributed File System, Parallel Processing, MapReduce

I. INTRODUCTION

Apache Hadoop[3]

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a thousands of computational independent computers and petabytes of data. Hadoop was derived from Google's Map Reduce and Google File System (GFS).

HDFS (Hadoop Distributed File System)[4]

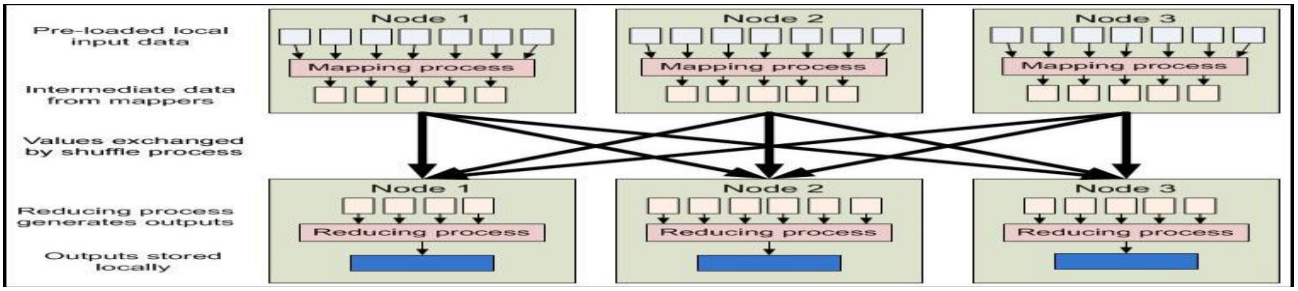
The Hadoop Distributed File System (HDFS) is a distributed file system providing fault tolerance and designed to run on commodity hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. Hadoop provides a distributed file system (HDFS) that can store data across thousands of servers, and a means of running work (Map/Reduce jobs) across those machines, running the work near the data. HDFS has master/slave architecture. Large data is automatically split into chunks which are managed by different nodes in the hadoop cluster.

HBASE[5]

HBase is a column-oriented database management system that runs on top of HDFS. It is well suited for sparse data sets, which are common in many big data use cases. Unlike relational database systems, HBase does not support SQL.In fact, HBase isn't a relational database at all. HBase applications are written In Java much like a typical MapReduce application.

Map Reduce[6]

Map educe is a software framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers. Map Reduce is a programming model for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with the same intermediate key.



"Map" step: The master node takes the input, partitions it up into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain:

Map (k1, v1) → list (K2, v2)

"Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain: Reduce (K2,

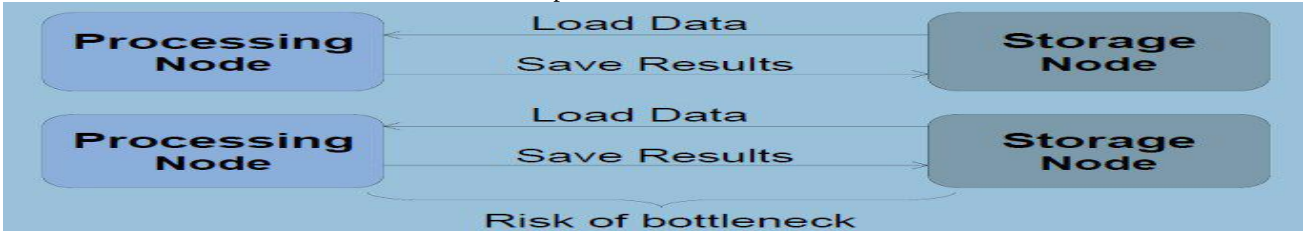
list (v2)) → list (v3)

II. LITERATURE SURVEY

Dr. Yan Mo won the 2012 Nobel Prize in Literature. This is probably the most controversial Nobel prize of this category, as Mo speaks Chinese, lives in a socialist country, and has the Chinese government's support. Searching on Google with "Yan Mo Nobel Prize", we get 1,050,000 web pointers on the Internet (as of January 3, 2013). "For all praises as well as criticisms," said Mo recently, "I am grateful." What types of praises and criticisms has Mo actually received over his 31-year writing career? As comments keep coming on the Internet and in various news media, can we summarize all types of opinions in different media in a real-time fashion, including updated, cross-referenced discussions by critics? This type of summarization program is an excellent example for Big Data processing, as the information comes from multiple, heterogeneous, autonomous sources with complex and evolving relationships, and keeps growing. Along with the above example, the era of Big Data has arrived (Nature Editorial 2008; Mervis J. 2012; Labrinidis and Jagadish 2012). Every day, 2.5 quintillion bytes of data are created and 90% of the data in the world today were produced within the past two years[9] *(IBM 2012).* Our capability for data generation has never been so powerful and enormous ever since the invention of the Information Technology in the early 19th century. As another example, on October 4, 2012, the first presidential debate between President Barack Obama and Governor Mitt Romney triggered more than 10 milliontweets within two hours (Twitter Blog 2012). Among all these tweets, the specific moments that generated the most discussions actually revealed the public interests, such as the discussions about Medicare and vouchers. Such online discussions provide a new means to sense the public interests and generate feedback in real-time, and are mostly appealing compared to generic media, such as radio or TV broadcasting. Another example is Flickr[10], a public picture sharing site, which received 1.8 million photos per day, on average, from February to March 2012 (Michel F. 2012). Assume the size of each photo is 2 megabytes (MB), this resulted in 3.6 terabytes (TB) storage every single day. As "a picture is worth a thousand words", the billions of pictures on Flicker are a treasure tank for us to explore the human society, social events, public affairs, disasters etc., only if we have the power to harness the enormous amount of data. The above examples demonstrate the rise of Big Data applications where data collection has grown tremendously and is beyond the ability of commonly used software tools to capture, manage, and process within a "tolerable elapsed time". The most fundamental challenge for the Big Data applications is to explore the largevolumes of data and extract useful information or knowledge for future actions (Rajaraman and Ullman, 2011). In many situations, the knowledge extraction process has to be very efficient and close to real-time because storing all observed data is nearly infeasible. For example, the Square Kilometer Array (SKA) (Dewdney *et al.* 2009) in Radio Astronomy consists of 1,000 to 1,500 15- meter dishes in a central 5km area. It provides 100 times more sensitive vision than any existing radio telescopes, answering fundamental questions about the Universe. However, with a 40 gigabytes (GB)/second data volume, the data generated from the SKA is exceptionally large. Although researchers have confirmed that interesting patterns, such as transient radio anomalies (Reed *et al.* 2011) can be discovered from the SKA data, existing methods are incapable of handling this Big Data. As a result, the unprecedented data volumes require an effective data analysis and prediction platform to achieve fast response and real-time classification for such Big Data.

III. EXISTING SYSTEM

RDBMS databases arose in a world where query flexibility was more important than flexible schemas. An RDBMS is a database that follows Codd's 12 Rules. Typical RDBMSs are fixed-schema, roworiented databases with ACID properties and a sophisticated SQL query engine. The emphasis is on strong consistency, referential integrity, abstraction from the physical layer, and complex queries through the SQL language. For a majority of small- to medium volume applications, there is no substitute for the ease of use, flexibility, maturity, and powerful feature set of available open source RDBMS solutions like MySQL and PostgreSQL. However, if you need to scale up in terms of dataset size, read/write concurrency, or both, you'll soon find that the conveniences of an RDBMS come at an enormous performance penalty and make distribution inherently difficult. The scaling of an RDBMS usually involves breaking Codd's rules, loosening ACID restrictions, forgetting conventional DBA wisdom, and on the way losing most of the desirable properties that made relational databases so convenient in the first place.

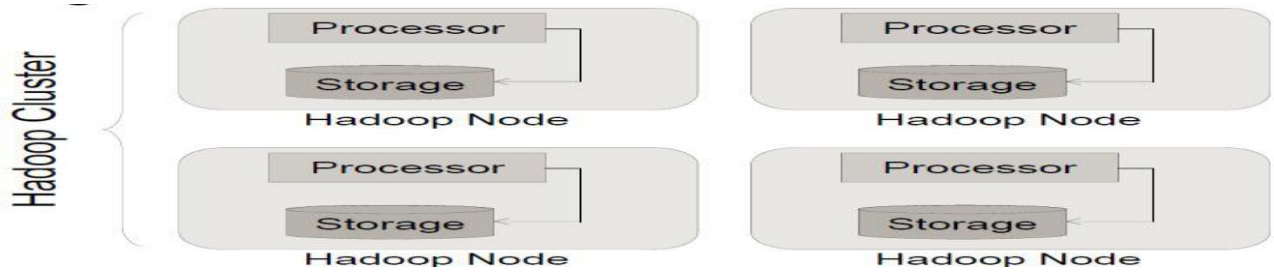


Disadvantages

1. RDMBS have ACID and supports transactions. Scaling "out" with RDBMS is harder to implement due to these concepts. 2. RDBMS tends to rely on expensive proprietary servers and storage systems to manage the exploding data and transaction volumes. 3. Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels.

IV.PROPOSED SYSTEM

HBase is a column-oriented database management system that runs on top of HDFS. It is well suited for sparse data sets, which are common in many big data use cases. Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical MapReduce application. HBase does support writing applications in Avro, REST and Thrift.



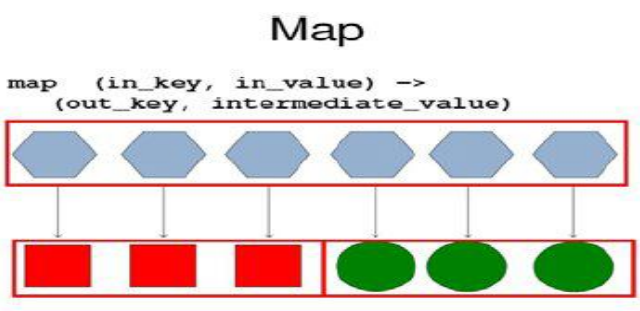
Advantages

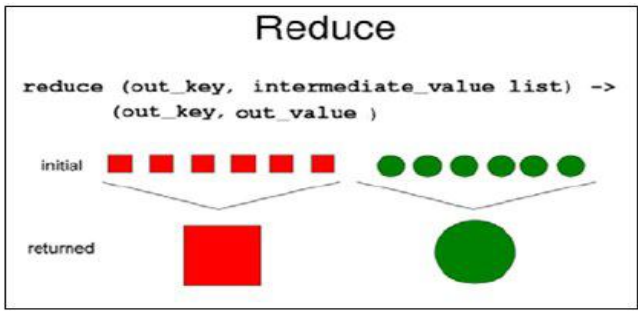
1. Scale-Out rather than Scale-Up. Instead of adding bigger servers to handle more data load, a NoSQL database allows a company to distribute the load across multiple hosts as the load increases. 2. Brings code to data rather than data to code.3. Deal with failures – they are common .4. Abstract complexity of distributed and concurrent applications. It frees developer from worrying about system level changes.

V. MAP-REDUCE

The MapReduce algorithm consists of two steps-

MAP:-The master node takes the input, divides it into smaller sub problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

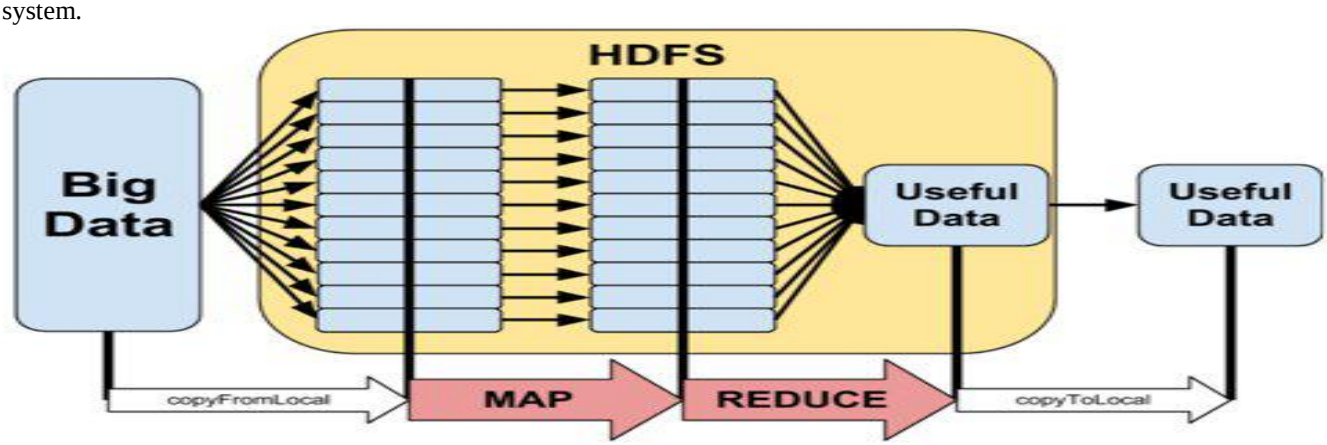




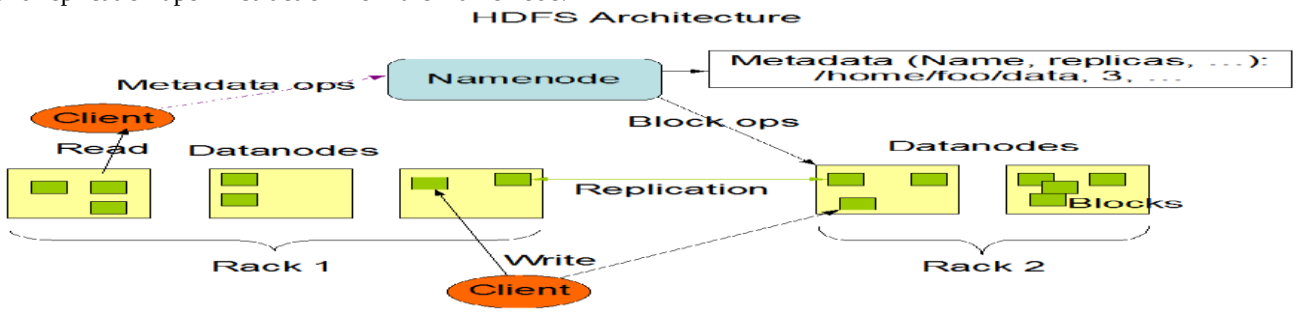
REDUCE:- The master node then collects the answers to all the subproblems and combines them in some way to form the output – the answer to the problem it was originally trying to solve

.

As the various tasks run in parallel, it manages all communications and data transfers between the various parts of the system.



HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.



The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode. HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. 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.

VI. REQUIREMENT SPECIFICATION

Input: - What exactly is getting inputted?
For instance: - Textual data, Multimedia.. Etc.
Output: - What exactly is the expected output from theanalyzed input?
Functional Requirements:-It defines the function of the System (BigData).Meaning :-Could be set of inputs, the behavior and the output. Simply, a functional requirement defines what a system is supposed to Do like PUSH, EXECUTE, VIEW, DELETE etc

In this case, we are analyzing large scale of Data Sets and their retrieval. Non-Functional Requirements:-
It specifies the criteria that can be used. For instance: - Hadoop and MapReduce, in this case. Simple, it defines how a system is supposed to perform.System should be up and running always External Interface Requirements:-LAN,Routers

VII. DESIGNING

Components of the system:-
• Client System:-System used by user for accessing, through a browser.

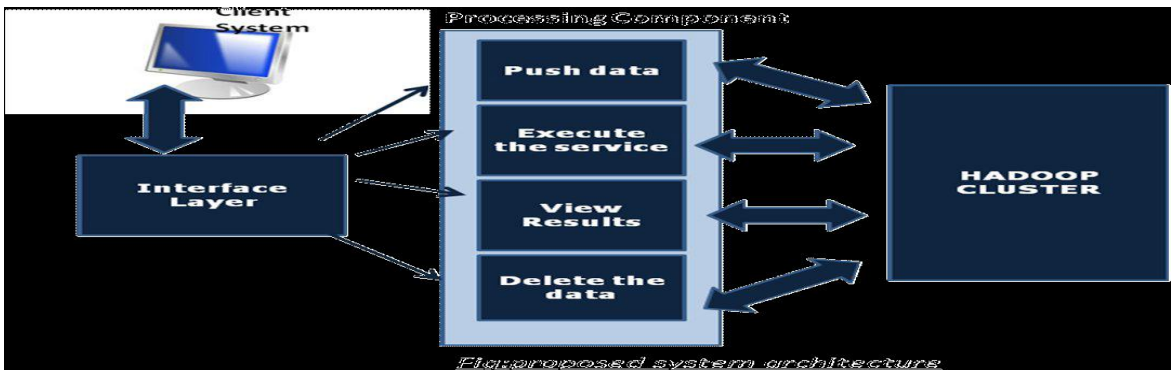
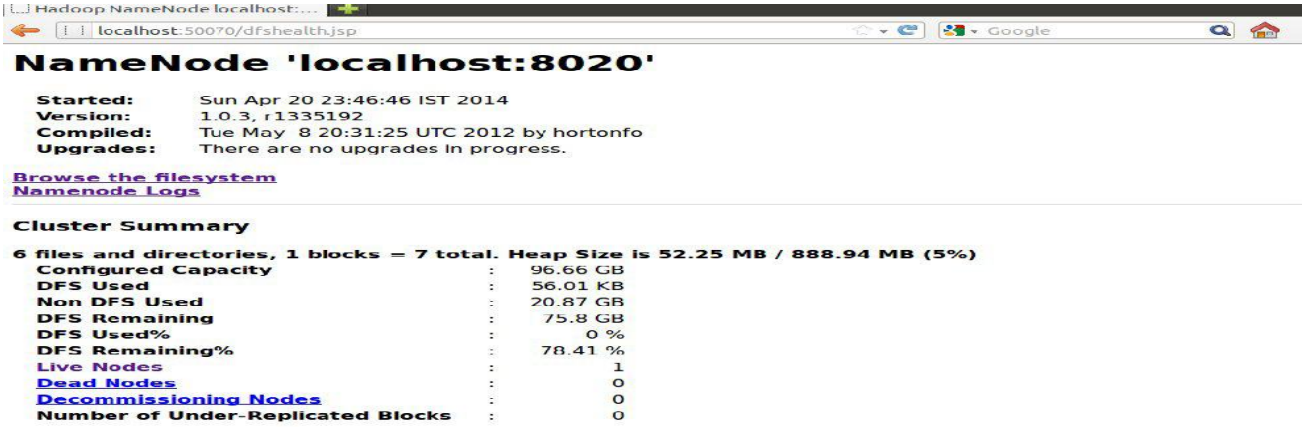


Fig:proposed system architecture

• Interface Layer:-Provides the user interface for the client system. From here the user can access all the functional operation of this system.
• Processing Component:-It will provide different functional capabilities to perform different operations on data such as reading, modifying etc.
• Hadoop Cluster:-Stores and manages the huge amount of data. In the server side where the hadoop cluster is present it can be monitored there also by the developer as shown in the figure by accessing the live nodes:-

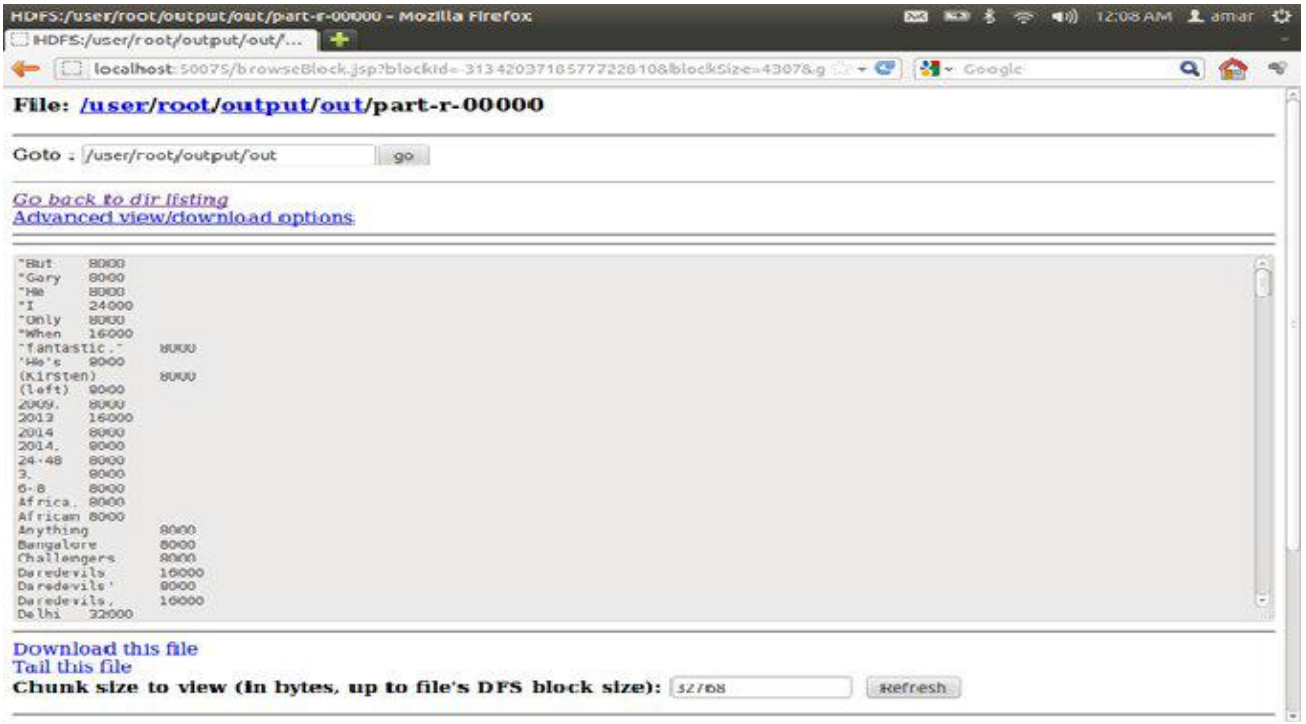


VIII. EXPERIMENTAL RESULTS

Considering the wordcount example for a text document, experiments were performed on both java and hadoop mapreduce environment for counting the number of words. The size of data is not huge as the experiments were performed in low configuration computers. The results are mentioned below:-

Table No.1

Size of File(approx)	Time Taken in Java(approx)	Time in Hadoop Environment(approx)
2 Mb	300 sec	100 sec
10 Mb	1400 sec	420 sec
180 Mb	20 minutes	6 minutes
400 Mb	45 minutes	14 minutes
720 Mb	1 hr 40 minutes	40 minutes



The results above state that proposed system's technique to access and analyze data is minimum 35-45% faster than the techniques used.

IX. RELATED WORK

MAPREDUCE has emerged as a popular and easy-to-use programming model for cloud computing . It has been used by numerous organizations to process explosive amounts of data, perform massive computation, and extract critical knowledge for business intelligence. Hadoop is an open source implementation of MapReduce, currently maintained by the Apache Foundation, and supported by leading IT companies such as Facebook and Yahoo!. Hadoop implements MapReduce framework with two categories of components: a JobTracker and many Task- Trackers. The JobTracker commands TaskTrackers (a.k.a.slaves) to process data in parallel through two main functions: map and reduce. In this process, the JobTracker is in charge of scheduling map tasks (MapTasks) and reduce tasks (ReduceTasks) to TaskTrackers. It also monitors their progress, collects runtime execution statistics, and handles possible faults and errors through task reexecution. Between the two phases, a ReduceTask needs to fetch a part of the intermediate output from all finished MapTasks. Globally, this leads to the shuffling of intermediate data (in segments) from all MapTasks to all ReduceTasks. For many data-intensive MapReduce programs, data shuffling can lead to a significant number of disk operations, contending for the limited I/O bandwidth. This presents a severe problem of disk I/O contention in MapReduce programs, which entails further research on efficient data shuffling and merging algorithms.

X. CONCLUSION

We have examined the design and architecture of Hadoop's MapReduce framework in great detail. Particularly, our analysis has focused on data processing. We would conclude by saying that bigdata is the new buzz word and Hadoop Mapreduce is the best tool available for processing data and its distributed, column-oriented database, HBase which uses HDFS for its underlying storage, and support provides more efficiency to the system.

ACKNOWLEDGEMENT

This work is a part of final year project work, East West Institute of Technology, Bangalore and completed under the guidance **Mr.Prasad M.R**, Asst. Prof, Dept of CSE, East West Institute of Technology, Bangalore.

REFERENCES

[1] Data Mining with BigData by Xindong Wu, Xingquan Zhu, Gong-Qing Wu, Wei Ding , 1041-4347/13/$31.00 © 2013 IEEE

[2] Ahmed and Karypis 2012, Rezwan Ahmed, George Karypis, Algorithms for mining the evolution of conserved relational states in dynamic networks, *Knowledge and Information Systems*, December 2012, Volume 33, Issue 3, pp 603-630

[3][4][5] Apache Hadoop Project, http://hadoop.apache.org/, 2013.

[6] Jiang, B.C. Ooi, L. Shi, and S. Wu, "The Performance of MapReduce: An In-Depth Study," Proc. VLDB Endowment, vol. 3, no. 1, pp. 472-483, 2010.

[7] Bughin et al. 2010, J Bughin, M Chui, J Manyika, Clouds, big data, and smart assets: Ten tech enabled business trends to watch, McKinSey Quarterly, 2010.

[8] Gillick et al., 2006, Gillick D., Faria A., DeNero J., MapReduce: Distributed Computing for Machine Learning, Berkley, December 18, 2006.

[9] IBM 2012, What is big data: Bring big data to the enterprise, http://www-01.ibm.com/software/data/bigdata/, IBM.

[10] Michel F. 2012, How many photos are uploaded to Flickr? http://www.flickr.com/photos/franckmichel/6855169886/.

[11] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Symp. Operating System Design and Implementation (OSDI '04), pp. 137-150, Dec. 2004.

[12] D. Jiang, B.C. Ooi, L. Shi, and S. Wu, "The Performance of MapReduce: An In-Depth Study," Proc. VLDB Endowment, vol. 3, no. 1, pp. 472-483, 2010.

[13] T. Condie, N. Conway, P. Alvaro, J.M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce Online," Proc. Seventh USENIX Symp. Networked Systems Design and Implementation (NSDI), pp. 312-328,Apr

