BIG DATA: HANDLING AND ANALYSIS USING MAPREDUCE

Samvit Chrungoo[1], Ravichandra H[2], Amutha S[3]
[1](Department of Computer Science & Engineering,
Dayananda Sagar College of Engineering, Bangalore. Email: samchrungoo.sc@gmail.com)
[2] (Assistant professor, Department of Computer Science & Engineering ,
Dayananda Sagar College of Engineering, Bangalore Email: ravihpvgd@gmail.com)
[3](Professor, Department of Computer Science & Engineering ,
Dayananda Sagar College of Engineering, , Bangalore Email: amuthanandhu@gmail.com)

Abstract: A tremendous quantity of data is generated in organizations today and it is growing continuously hence very difficult to analyze such a vast amount of data. Researchers are going on in data mining to address the problem of discovering knowledge or extracting patterns from these continuously growing large sets of data. There is much valuable information which resides in these large databases which can be used once analyzed. Data mining has grown into an thought-provoking area for extracting the precious surrounded information from large data sets. In recent years, many computational and data intensive and data analyses methods are time-honoured. For performing the large- scale data mining and analyses to meet scalability and performance of big data, many similar and parallel methods are applied. Some collective techniques used for parallelization are threads, MPI, MapReduce etc. The MapReduce paradigm is currently having a considerable enthusiasm for large-scale data analyse. It is inspired by functional programming which expresses distributed computations on a huge amount data. It is basically designed for large scale data processing as it allows running on clusters of commodity hardware. MapReduce is considered as a prominent parallel computing paradigm and is gaining sufficient momentum. In this paper we will go around MapReduce, its pros and cons and how it can be used in different technologies. We will also see about Hadoop which is a framework of tools that supports reliable distributed computing and processing of large data sets.

Keywords: *MapReduce, Hadoop, BigData, Scalable, Data mining, Parallel Processing.*

I. INTRODUCTION

One of the challenging problems nowadays is to analyse Big data. For handling such large amount of data and applications effectively, the MapReduce framework is widely used. Over the last few years, MapReduce has emerged as the most prevalent parallel computing model by using batch-style analyses of huge amount of data. Many areas having huge amount of data uses MapReduce for data analyse and prediction [1]. There are various other applications that handle Big data but still handling such a large amount of data remains an obstacle. Talking about Big data analytics, it is a process which allows data scientists and various other users to evaluate large amount of data of transactions or any other data sources generally to discover hidden patterns, market drifts, unknown relationships, customer preferences and other important information that enables an organisation to perform better. There are three V's associated with Bigdata namely Volume (the amount of data generated based on application), Velocity (Data generation rate), Variety (Data generated type) [10]. The MapReduce paradigm got recognised when it was successfully used by Google. MapReduce can be considered as a mountable and liability tolerant data processing tool which gives power to process huge amount of data in parallel with numerous low-end computing nodes [2]. MapReduce, because of its salient features like scalability, fault-tolerance and simplicity is becoming ubiquitous and is gaining tremendous drive. However, it has some limitations on the side of efficiency and performance. So, various studies have endeavoured for overcoming the limitations of this frame work [3][4]. The goal of this analysis is to brief about MapReduce and bring out the advantages and limitations of this framework.

II. RESEARCH BACKGROUND

The MapReduce framework was originally proposed by Google in the year 2014. It is used for processing a voluminous amount of data in a parallel manner. It allows the user to work freely and focus on given challenges rather than worrying about parallelization details like fault tolerance, load balancing, data distribution etc. Basically, MapReduce framework works on key-value pairs. The data which is given as input is divided into several parts and then is stored into the different nodes. It uses two functions, one is map function and another is reducing function. Map function proceeds with key value pairs from each node as input and generates key value pairs which indicate local frequent item set as output. The Reduce function takes the local frequent item set as the input and combines these key-value pair and generates output as key-value pair which indicates global frequent item set. The above process can be easily and effectively implement using Hadoop MapReduce frame shown in Figure 1.

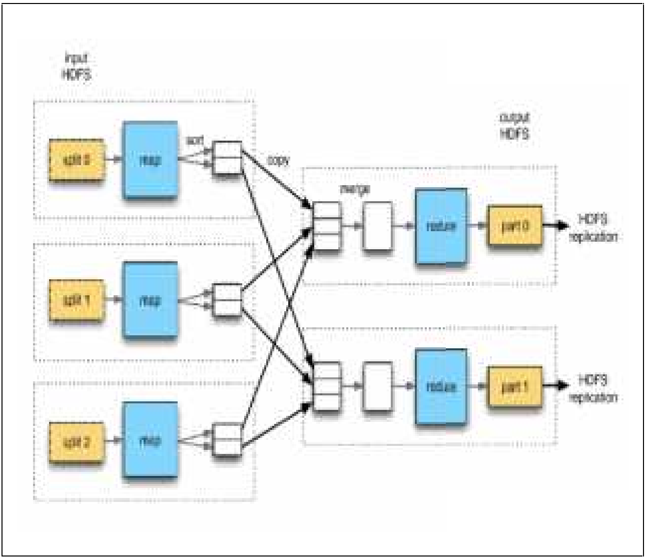

Figure1: Template of Map-Reduce

The Hadoop MapReduce is a software framework which makes it easy to write applications and programs that process huge amount of data sets paralleled on huge clusters of commodity hardware in a way which is reliable. Hadoop is open software that is built on Hadoop Distributed File System (HDFS). The MapReduce framework and HDFS are running on the very same node. In MapReduce, a large dataset is broken into multiple blocks. Each block is stored on distinct nodes to form cluster.

III. MAP-REDUCE ARCHITECTURE

To look at the architecture of MapReduce let's have a brief of Hadoop. Hadoop is a frame work which entails of two components (i) Hadoop Distributed File System- which is used as storage of Bigdata. (ii) Hadoop MapReduce- which is used to process the data present in HDFS. It stores files by dividing them into different blocks, the default size of each block is 128 MB. The data in the blocks are replicated in many different data nodes, which by default are three. There are two phases in which a MapReduce task can be divided (i) Map phase and (ii) Reduce phase. The Map phase consists of various map tasks which maps the values of data sets and gives provides the input for the reduce task. The minute every Map task is finished, the Reduce phase starts. This Reduce phases call the Reduce function written by user and then stores its output to HDFS. There are things like Knowledge compression which can be used to reduce the I/O time and the storage consumption for Hadoop framework. As it reduces the I/O time, still it increases the compute time. Henceforth a proper trade-off between the I/O and storage should be defined while dealing with such a large amount of data. This defines a brief description of Hadoop MapReduce process.

Talking about the architecture, MapReduce is used in clustering on the analyses of Big data. As the Big data is evolving, there are many different analytical approaches which are being rapidly developed. As discussed above, when the data is stored in the HDFS, it gets split in several files across machines and intermediate key-value pairs will be generated by the mapper on applying to every input key- value pairs. Then comes reducer which is applied to all values associated with the intermediate key-value pairs for generating output key-value pairs [5]. The simplified view of MapReduce is shown in Figure 2.can be described by certain steps [6][7]. Following are the series of steps for its working pattern.

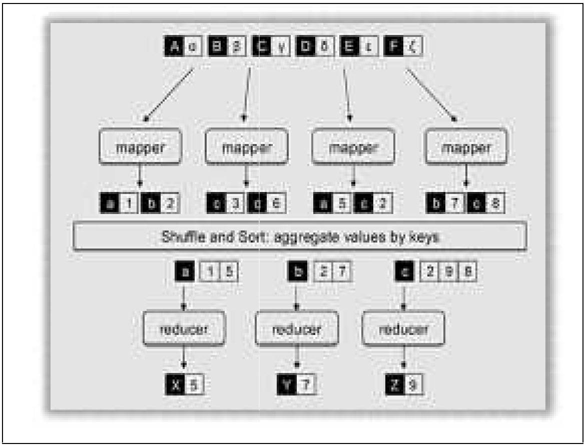

Figure2: Template of Map-Reduce work flow

The whole process also includes some other parts of MapReduce, they are mentioned below with functionalities:

Mappers: These are the tasks which are applied to every input key-value pairs to generate some intermediate key-value pair.

Reducers: These are applied to all intermediate values with a same associated key.

Partitioners: It divides the intermediate key space, after this it assigns the intermediate key-value pairs to the reducers. It is also accountable to stipulate intermediate key-value pair to which it should be clichéd.

Combiners: It is used before the shuffle and sort phase thereby providing us with local aggregation of data. Basically, the role of combiner is to save the bandwidth. Combiners are generally implemented by using local data structures. For example, intermediate computations and aggregations can be stored using an associative array.

MapReduce is a technique which is used to process huge amount and multi structured data across many data sets. It modularises the process by splitting the process in several small chunks. These chunks can be processed parallelly across many different nodes. This results in achieving high performance. MapReduce is a model for processing and producing huge data sets. The programs written in this functional approach are parallelised automatically and can be run on large clusters of commodity hardware/machines. Programmers are benefited by this as it makes it easy for them to use the various resources of a distributed and a parallel system. The working of MapReduce

Step 1: Single block is processed by one mapper at a time. The mapper is specified by the developer, so, it can specify own business or logic goals as per the requirements of the user.

Step 2: Output from Mapper also referred as intermediate output is written/copied to the local disk. Remember that the output of Mapper is not stored in HDFS as it is temporary, and we don't need any duplicates of it.

Step 3: Output of mapper is shuffled according to the key to the reducer node. The copying/hobbling is a corporeal movement of data which is done across the network [1][8].

Step 4: After every mapper is finished with its job and their outputs are shuffled on reducer nodes, then intermediate output is merged and sorted, then given as an input to the reduce phase.

Step 5: Reduce is a phase of processing where the user specifies his logic or business requirements. The participants to the reducer are the sorted output of all the mappers. The output of the reducer is the final output, which is written in HDFS.

The strategy for implementations in MapReduce is determined wholly at runtime. At runtime scheduling, fault tolerance is achieved by detecting failures and the reassigning tasks to other vigorous nodes in the cluster. Once a node completes the job assigned to it, it is immediately give next input block to process. In this way load balancing can be achieved. The faster nodes can process more number of blocks and this will leave slower nodes with few blocks to process thus, balancing the load. Moreover, MapReduce uses a redundant and speculative execution [12]. Tasks on sprawling nodes are excessively performed on other idle nodes that are done with the task assigned to them, though the tasks are not sure to end earlier on the newly assigned node than the sprawling node. Map and Reduce tasks are performed with no interaction between other tasks. Thus, there is no argument risen by synchronisation and no interaction cost between the tasks whilst a MapReduce execution. Although MapReduce is considered as the latest way of processing Big data, it has some disadvantages and is also criticized in comparison with Data Base Systems [5][6]. Still, many MapReduce supporters in industry say that MapReduce is not like a Data Base Systems and comparison among them is not fair.

IV. BENEFITS OF MAP-REDUCE

MapReduce is efficient yet simple for computing. Hence, it is often associated with DBMS query processing. For processing huge amount of data, the main leads of the MapReduce framework are as follows [5]: **Simple to use and easy**: The MapReduce framework is very simple at the same time very expressive. Every task is defined using Map and Reduce task only.

Very Flexible: MapReduce doesn't have any dependence any data model or schema.
Dealing with unstructured or irregular data is far simpler as compared to the DBMS.

Autonomous storage: The MapReduce is free from fundamental storage layer. Therefore, MapReduce can handle diverse storage layers
such as BigTable etc.

Fault Tolerance: MapReduce has been known for being very fault tolerant. It is also stated
that MapReduce can endure an average of 1.2
failures per analysis task at Google.

High Scalability: The benefit provided by MapReduce is very high scalability. As reported by Yahoo!, the Hadoop gear they have can scale out 4000 plus nodes in the year
2008.

Therefore, because of these key advantages of this framework, it has been integrated with many advance techniques for processing huge databases [9]. The growing curiosity and admiration of MapReduce has made many DBMS vendors to incorporate MapReduce features. This ability not only gives the benefits mentioned above for positioning user defined purpose, but as well add the benefits of MapReduce to the relational Data Base Systems environment.

V. CHALLENGES IN MAP-REDUCE

1. MapReduce has become omnipresent, even though having its performance and efficiency under scrutiny. However, MapReduce displays that many difficulties could be solved in this framework. [11] [12].
2. Due to regular barriers and runtime planning with abstract execution, MapReduce exposes low efficiency. However, these procedures are very essential for achieving fault tolerance and high scalability in Big data analyses.
3. Increasing efficiency maintaining the same status of fault tolerance and scalability in a very challenging task.
4. Utilizing the features of recent hardware is still unanswered in many areas. However, recent devices like chip level multiprocessors and Solid State Disk (SSD) can significantly help in reducing the I/Os and computations.

5. The continuously growing size of MapReduce clusters is also a challenge. A cluster containing 4000 nodes is not surprising nowadays. Managing resources efficiently in such a huge cluster in an environment which supports multiple users and then utilizing the resources in surely difficult.
Research is going on to solve these challenges in MapReduce so that it can be optimized and
can be used in a better way [10].

VI. CONCLUSION

The numerous benefits as well as challenges of MapReduce have been briefed. MapReduce framework is very simple but offers decent fault tolerance and scalability for huge data processing. The traditional mining methods become ineffective for mining such a big amount of data because of large resource criteria and excess communication cost. This programming framework as a parallel programming model has emerged for mining such data. Google is successfully using the MapReduce model for many different purposes. The success of MapReduce can be credited to several reasons. Firstly, using this framework is extremely easy, even if you don't have any prior experiences with distributed and parallel systems. Secondly, MapReduce can easily express a large variety of problems. Thirdly, it is a framework which can scale up to hundreds and thousands of nodes. Several repairs in the existing system are looked upon in the hope of improving the model which will lead in reducing the amount of data which is being sent across the network. However, MapReduce is dubious to replace DBMS even for the process of storing data (Data Warehousing). Instead, MapReduce is likely to compliment DBMS with its parallel processing, providing features like fault tolerance, load balancing and high scalability for many data analytical processes.

REFERENCES

[1] Maitrey S, Jha. An Integrated Approach for CURE Clustering using Map-Reduce Technique. Elsevier, ISBN 978-81- 910691-6-3,2nd August 2013.
[2] D. DeWitt and M. Stonebraker. MapReduce: A major step backwards. The Database Column, 1, 2008.
[3] A. Pavlov et al. A comparison of approaches to large-scale data analysis. ACM SIGMOD, pages 165–178, 2009.
[4] M. Stonebraker et al. MapReduce and parallel DBMSs: friends or foes? Communications of the ACM, 53(1):64–71, 2010.
[5] Eugene J. Shekita, Yuanyuan Tian. A comparison of join algorithms for log processing in MapReduce. Proceedings of the 2010 ACM SIGMOID International Conference on Management of data, Pages 975-986.
[6] Jeffrey Dean and et al. MapReduce: Simplified data processing on large clusters. 6th USENIX OSDI, pages 137–150, 2004.
[7] J. Dean and et al. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1):107– 113, 2008.
[8.] Kyong and et al. Parallel Data Processing with MapReduce: A Survey in SIGMOD Record, December 2011 (Vol. 40, No. 4).
[9] D. Jiang and et al. The performance of MapReduce: An in-depth study. Proceedings of the VLDB Endowment, 3(1-2):472– 483, 2010
[10] N. Li, and et al (2012). Parallel implementation of Apriori algorithm based on Map-Reduce. SNPD, pages 236–241.
[11] J. Ekanayake, and et al(2010). A runtime for iterative MapReduce. HPDC, pages 810–818. ACM.
[12] Zhiyong Ma and et al,(2016). An Improved Eclat Algorithm for Mining Association Rules Based on Increased Search Strategy "International Journal of Database Theory and Application Vol.9, No.5, pp.251-266.