Scientific Journal Impact Factor (SJIF) : 3.518
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

A STUDY ON MAPREDUCE FOR SIMPLIFIED PROCESSING OF BIG DATA

Ch. Shobha Rani
Research Scholar, Department of Computer Science, Kakatiya University, Warangal, Telangana

Abstract: Big Data refers to large and complex data sets made up of a variety of structured and unstructured data which are too big, too fast, or too hard to be managed by traditional techniques. Hadoop is an open source software platform for structuring Big Data on computer clusters built from commodity hardware and enabling it for data analysis. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance. Map Reduce is a programming model for processing large data sets with parallel distributed algorithm on cluster.[1] This paper presents the survey of Simplified big data processing with Hadoop architecture and mapreduce framework.

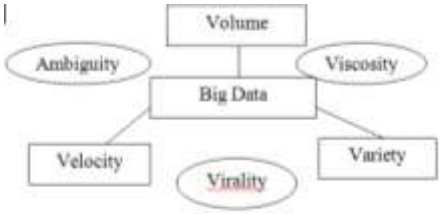
Keywords: Big data, Hadoop, Mapreduce, HDFS, BigData Analytics.

I. INTRODUCTION

Currently, enormous amounts of data are created every day. With the rapid expansion of data, we are moving from the Terabyte to the Petabyte Age. At the same time, new tech-nologies make it possible to organize and utilize the massive amounts of data currently being generated. However this trend creates a great demand for new data storage and analysis methods.[2] Especially, theoretical and practical aspects of extracting knowledge from massive data sets have become quite important.

The biggest phenomenon that has captured the attention of the modern computing industry today since the "Internet" is "Big Data". The fundamental reason why "Big Data" is popular today is because the technology platforms that have emerged along with it, provide the capability to process data of multiple formats and structures without worrying about the constraints associated with traditional systems and database platforms.

Big Data can be defined as volumes of data available in varying degrees of complexity, generated at different velocities and varying degrees of ambiguity, that cannot be processed using traditional technologies, processing methods, algorithms, or any commercial off-the-shelf solutions. Data defined as Big Data includes machine-generated data from sensor networks, nuclear plants, X-ray and scanning devices, and airplane engines, and consumer-driven data from social media. Big Data producers that exist within organizations include legal, sales, marketing, procurement, finance, and human resources departments.



Along with the three V's, there also exists ambiguity, viscosity, and virality.

- ✓ **Ambiguity**—a lack of metadata creates ambiguity in Big Data. For example, in a photograph or in a graph, M and F can depict gender or can depict Monday and Friday. This characteristic manifests in the volume-variety category most times.
- ✓ **Viscosity**—measures the resistance (slow down) to flow in the volume of data. Resistance can manifest in dataflows, business rules, and even be a limitation of technology. For example, social media monitoring falls into this category, where a number of enterprises just cannot

understand how it impacts their business and resist the usage of the data until it is too late in many cases.

✓ **Virality**—measures and describes how quickly data is shared in a people-to-people (peer) network. Rate of spread is measured in time. For example, re-tweets that are shared from an original tweet is a good way to follow a topic or a trend.

Big Data is non relational (traditional)

MapReduce is complementary to DBMS, not a competing technology. [3]

- Parallel DBMS are for efficient querying of large data sets. [3]
- MR-style systems are for complex analytics and ETL tasks.
- Parallel DBMS require data to fit into the relational paradigm of rows and columns.
- In contrast, the MR model does not require that data files adhere to a schema defined using the relational data model. That is, the MR programmer is free to structure their data in any manner or even to have no structure at all.

Big Data Pillars

- Big Table – Relational, Tabular format – rows & columns. [3]
- Big Text – All kinds of unstructured data, natural language, grammatical data, semantic data. [3]
- Big Metadata – Data about data, taxonomies, glossaries, facets, concepts, entity.
- Big Graphs – object connections, semantic discovery, degree of separation, linguistic analytic, subject predicate.

In the Big Data community, MapReduce has been seen as one of the key enabling approaches for meeting continuously increasing demands on computing resources imposed by massive data sets. The reason for this is the high scalability of the MapReduce paradigm which allows for massively parallel and distributed execution over a large number of computing nodes.

Data processing has been a complex subject to deal with since the primitive days of computing. The underlying reason for this stems from the fact that complexity is induced from the instrumentation of data rather than the movement of data. As a reaction to this complexity, a new abstraction was designed that allows to express the simple computations being tried to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. That abstraction is inspired by the ***map*** and ***reduce*** primitives. Most of the computations involved applying a ***map*** operation to each logical "record" in the input in order to compute a set of intermediate key/value pairs, and then applying a ***reduce*** operation to all the values that shared the same key, in order to combine the derived data appropriately. The use this functional model with user-specified map and reduce operations allows to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance.[1]

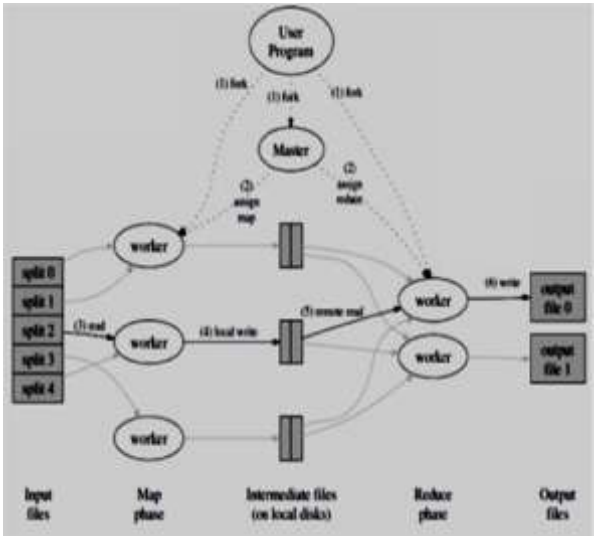
The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.[1]

II. MAPREDUCE PROGRAMMING MODEL

MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.[2]

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Processing can occur on data stored either in a files system (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing it on or near the storage assets in order to reduce the distance over which it must be transmitted.[2]

Map Reduce framework :



Map Phase: The master node takes the input, divides it into smaller sub-problems, and distribute 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.[2]

Reduce Phase : The master node then collects the answers to all the sub-problems and combine them in some way to form the output – the answer to the problem it was originally trying to solve.[2

Input reader: It divides the input into appropriate size (in practice typically 64 MB to 512 MB as per HDFS) and the framework assigns one split to one Map function. The input reader reads the data from stable storage (typically as in our case Hadoop distributed file system) and generates key/value pairs. [2]

Map function: Each Map function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs. The input and output types of the map can be and often are) different from each other. [2]

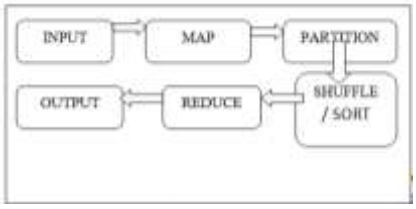
Partition function: Each Map function output is allocated to a particular reducer by the application' partition function for sharing purposes. The partition function is given the key and the number of reducers and returns the index of desired reduce. [2]

Comparison function: The input for every Reduce is fetched from the machine where the Map run and sorted using comparison function. [2]

Reduce function: The frame work calls the applications Reduce function for each unique key in the sorted order. It also itereates through the values that are associated with that key and produce zero or more outputs. [2]

Output writer: It writes the output of the Reduce function to stable storage, usually a Hadoop distributed file system. [2]

Their execution sequence can be seen as follows:



Performance :

MapReduce programs are not guaranteed to be fast. The main benefit of this programming model is to exploit the optimized shuffle operation of the platform, and only having to write the *Map* and *Reduce* parts of the program. In practice, the author of a MapReduce program however has to take the shuffle step into consideration; in particular the partition function and the amount of data written by the *Map* function can have a large impact on the performance. Additional modules

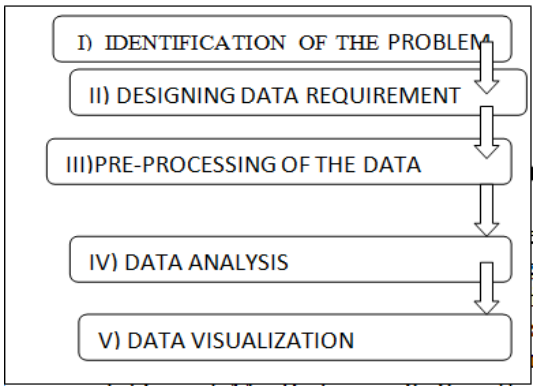
such as the *Combiner* function can help to reduce the amount of data written to disk, and transmitted over the network.

When designing a MapReduce algorithm, the author needs to choose a good tradeoff between the computation and the communication costs. Communication cost often dominates the computation cost, and many MapReduce implementations are designed to write all communication to distributed storage for crash recovery.

III. BIG DATA ANALYTICS

The infrastructure required for analyzing big data must be able to support deeper analytics such as statistical analysis and data mining, on a wider variety of data types stored in diverse systems; scale to extreme data volumes; deliver faster response times driven by changes in behavior; and automate decisions based on analytical models. Most importantly, the infrastructure must be able to integrate analysis on the combination of big data and traditional enterprise data. New insight comes not just from analyzing new data, but from analyzing it within the context of the old to provide new perspectives on old problems. For example, analyzing inventory data from a smart vending machine in combination with the events calendar for the venue in which the vending machine is located, will dictate the optimal product mix and replenishment schedule for the vending machine.

The data analytics project life cycle stages are seen in the following diagram:



IV. CONCLUSION

There are many new technologies emerging at a rapid rate, each with technological advancements and with the potential of making ease in use of technology. However one must be very careful to understand the limitations and security risks posed in utilizing these technologies. Neither MapReduce-like software, nor parallel databases are ideal solutions for data analysis in the cloud. Hybrid solution that combines the fault tolerance, heterogeneous cluster, and ease of use out-of-the-box capabilities of MapReduce with the efficiency, performance, and tool plug ability of shared-nothing parallel systems could have a significant impact on the cloud market. This paper analyzes the concept of Big data and how it differs from traditional database. It also clearly specifies the Hadoop environment, its architecture and how it can be implemented using MapReduce along with various functions. So, it is sure that this paper helps the researches to understand the basic concepts of Big data, Hadoop and MapReduce to move further.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in In Proceedings of OSDI'04: Sixth Symposium on Operating System Design and Implementation, December 2004.
[2] V. Patil, V.B. Nikam, "Study of Mining Alogorithm in cloud computing using MapReduce Framework", Journal of Engineering, Computers & Applied Sciences (JEC&AS) Vol.2, No.7, July 2013.
[3] D. Usha, A.P.S. Aslin Jenil, " A Survey of Big Data Processing in Perspective of Hadoop and Mapreduce", International Journal of Current Engineering and Technology, Vol.4, No.2,April 2014.
[4] M. Deodhar, C. Jones, and J. Ghosh. "Parallel simultaneous co-clustering and learning with Map-Reduce", In GrC, 2000.

[5] Apache. Apache Hadoop. http://hadoop.apache.org,2010
[6] T. Sun, C. Shuy, F. Liy, H. Yuy, L. Ma, and Y. Fang. "An efficient hierarchical clustering method for large datasets with Map-Reduce", In PDCAT, 2009.
[7] S. Ghemawat et al . "The google file system." ACM SIGOPS Operating Systems Review, 37(5):29–43, 2003.
[8] Saptarshi Guha. "RHIPE- R and Hadoop Integrated Processing Environment." http://www.stat.purdue.edu/ sguha/rhipe/, 2010.
[9] R. Taylor. "An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics." ,BMC bioinformatics, 11(Suppl 12):S1, 2010.
[10] T. White. Hadoop: "The Definitive Guide.",Yahoo Press,2010.