International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Survey Paper on Big Data Processing and Hadoop Components

Swati M. Gavali[1], Supriya Sarkar[2]

[1, 2]Department of Computer Engineering, SKN, Lonavala, Pune, India

Abstract: ***The term 'Big Data', refers to data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to capture, manage, process or analyzed. To analyze this enormous amount of data Hadoop can be used. However, processing is often time-consuming. One way to decrease response time is to executing the job partially, where an approximate, early result becomes available to the user, before completion of job. The implementation of the technique will be on top of Hadoop which will help to sample HDFS blocks uniformly. We will evaluate this technique using real-world datasets and applications and we will try to demonstrate the system's performance in terms of accuracy and time. The objective of the proposed technique is to significantly improve the performance of Hadoop MapReduce for efficient Big Data processing.***

Keywords: Big data, Hadoop, MapReduce, RDMS

1. Introduction

1.1 Big Data

Big data is a term that refers to data sets or combinations of data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to be captured, managed, processed or analyzed [4] by conventional technologies and tools, such as relational databases and desktop statistics or visualization packages, within the time necessary to make them useful. While the size used to determine whether a particular data set is considered big data is not _rmly de_ned and continues to change over time, most analysts and practitioners currently refer to data sets from 30-50 terabytes(10 12 or 1000 gigabytes per terabyte) to multiple petabytes (1015 or 1000 terabytes per petabyte) as big data[1]. The analysis of Big Data involves multiple distinct phases each of which introduces challenges. [7]. Map Reduce has emerged as a popular way to harness the power of large clusters of computers. Map Reduce allows programmers to think in a data-centric fashion: they focus on applying transformations to sets of data records, and allow the details of distributed execution, network communication and fault tolerance to be handled by the Map Reduce framework. Map Reduce is typically applied to large batch-oriented computations that are concerned primarily with time to job completion [13]. The Google Map Reduce framework and open-source Hadoop system reinforce this usage model through a batch-processing implementation strategy: the entire output of each map and reduce task is materialized to a local _le before it can be consumed by the next stage. Materialization allows for a simple and elegant checkpoint/restart fault tolerance mechanism that is critical in large deployments, which have a high probability of slowdowns or failures at worker nodes [19].

1.2 3 Volume's of Big Data

Volume of data:

Volume refers to amount of data. Volume of data stored in enterprises repositories have grown from megabytes and gigabytes to petabytes.

Variety of data:

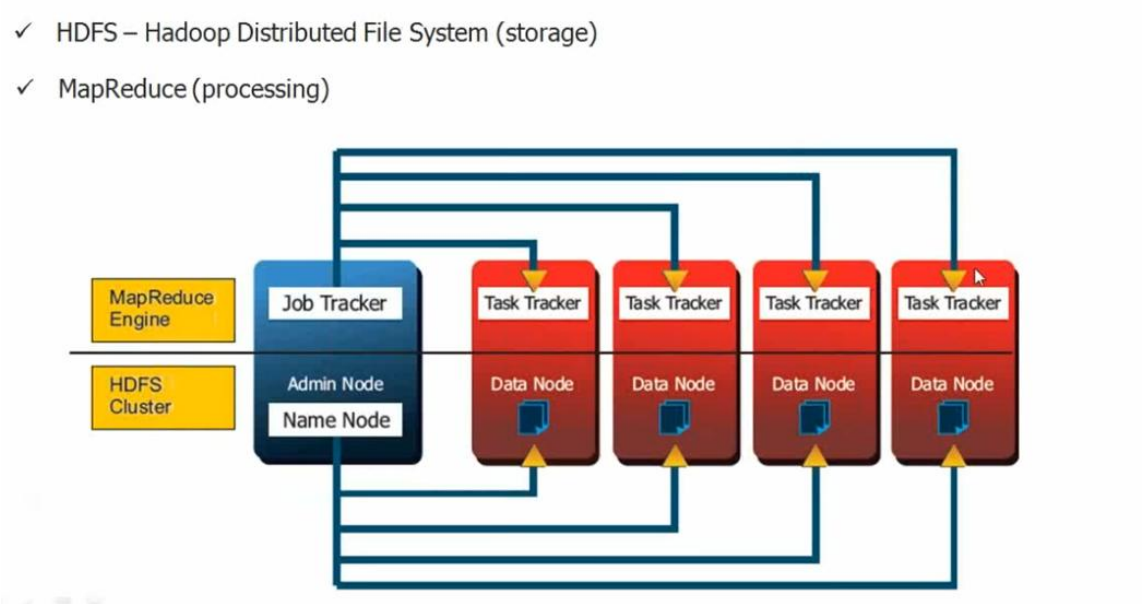
Different types of data and sources of data. Data variety exploded from structured and legacy data stored in enterprises repositories to unstructured, semi structured, audio, video, XML etc.

Velocity of data:

Velocity refers to the speed of data processing. For time-sensitive processes such as catching fraud, big data must be used as it streams into your enterprise in order to maximize its value.

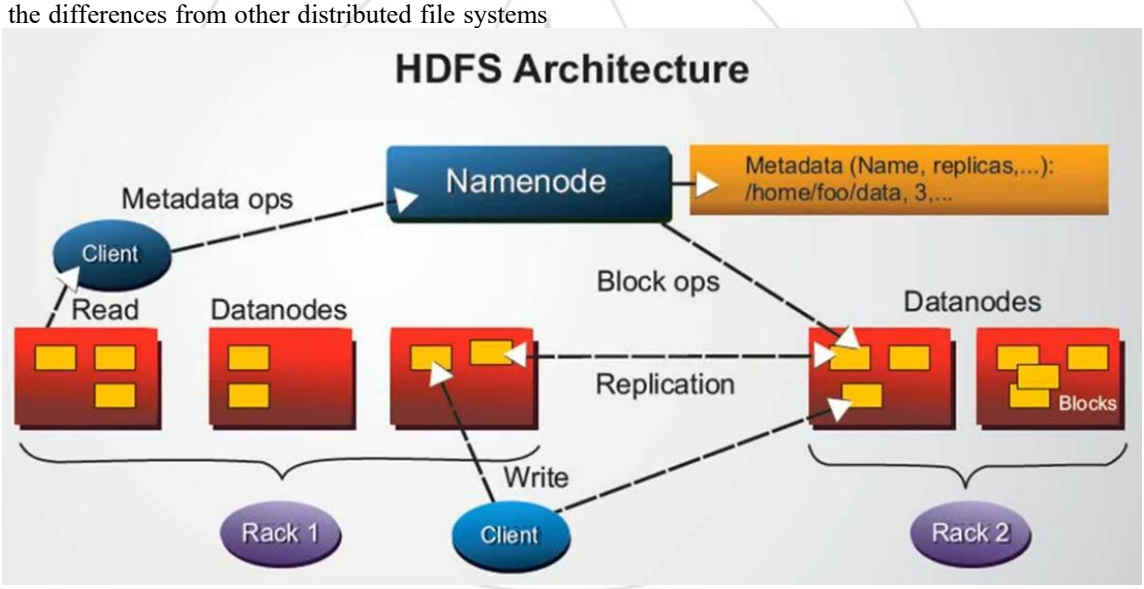
2. Hadoop

SQL standard and the relational data model provide a uniform, powerful language to express many query needs and, in principle, allows customers to choose between vendors, increasing competition. The challenge ahead of me is to combine these healthy features of prior systems [7]. Map Reduce has emerged as a popular way to harness the power of large clusters of computers. Map Reduce allows programmers to think in a data-centric fashion: they focus on applying transformations to sets of data records, and allow the details of distributed execution, network communication and fault tolerance to be handled by the Map Reduce framework. Map Reduce is typically applied to large batch-oriented computations that are concerned primarily with time to job completion [13]. The Google Map Reduce framework and open-source Hadoop system reinforce this usage model through a batch-processing implementation strategy: the entire output of each map and reduce task is materialized to a local _le before it can be consumed by the next stage. Materialization allows for a simple and elegant checkpoint/restart fault tolerance mechanism that is critical in large deployments, which have a high probability of slowdowns or failures at worker nodes [19].



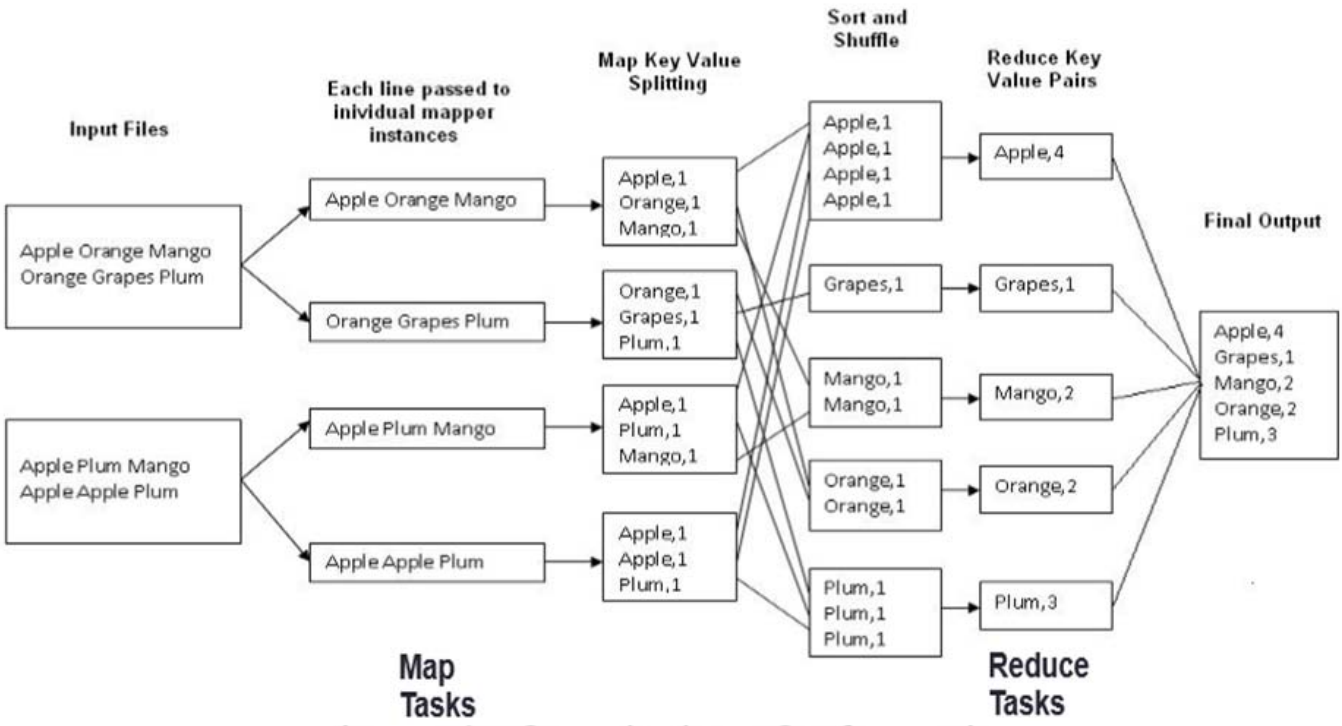
2.1 Hadoop Distributed File System(HDFS):

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.



2.2 Map-Reduce

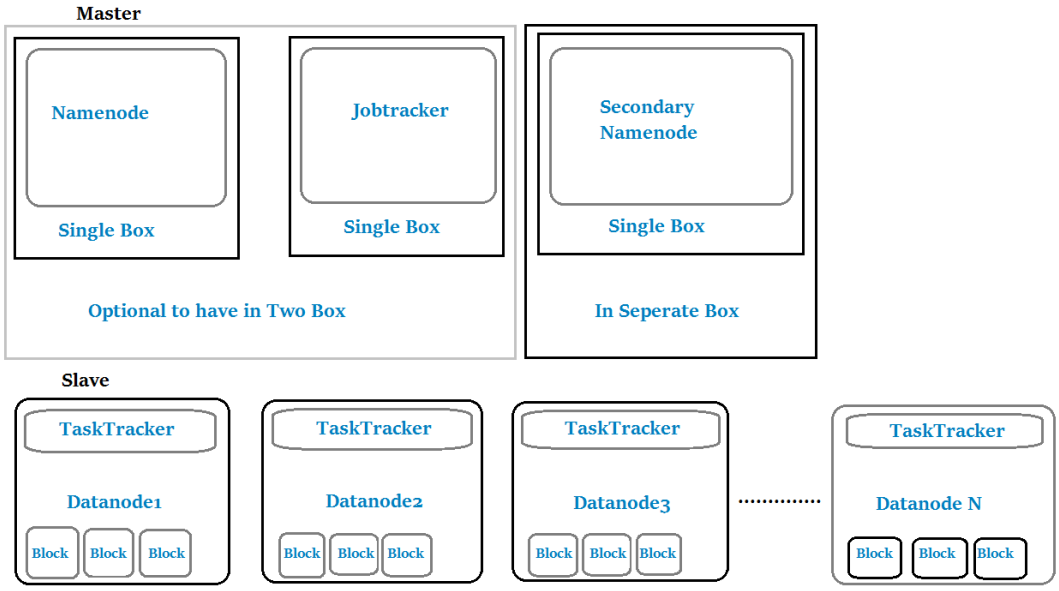
MapReduce is one of the parallel data processing paradigm designed for large scale data processing on cluster-based computing architectures. It was originally proposed by Google to handle large-scale web search applications. This approach has been proved to be an effective programming approach for developing machine learning, data mining, and search applications in data centers. Its advantage is that it allows programmers to abstract from the issues of scheduling, parallelization, partitioning, replication and focus on developing their applications. Hadoop MapReduce programming model consists of data processing functions: Map and Reduce. Parallel Map tasks are run on input data which is partitioned into fixed sized blocks and produce intermediate output as a collection of <key, value> pairs. These pairs are shuffled across different reduce tasks based on <key, value> pairs. Each Reduce task accepts only one key at a time and process data for that key and outputs the results as <key, value> pairs. The Hadoop MapReduce architecture consists of one JobTracker (Master) and many TaskTrackers (Workers). The JobTracker receives job submitted from user, breaks it down into map and reduce tasks, assigns the tasks to Task Trackers, monitors the progress of the Task Trackers, and finally when all the tasks are complete, reports the user about the job completion. Each Task Tracker has a fixed number of map and reduce task slots that determine how many map and reduce tasks it can run at a time. HDFS supports reliability and fault tolerance of MapReduce computation by storing and replicating the inputs and outputs of a Hadoop job.



3. MapReduce Framework

3.1 Master-Slave Architecture

Master: Namenode, JobTracker
Slave: {DataNode, TaskTraker}, ….. {DataNode, TaskTraker}



HDFS is one primary components of Hadoop cluster and HDFS is designed to have Master-slave architecture.
Master: NameNode
Slave: {Datanode}…..{Datanode}

The Master (NameNode) manages the file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes along with regulating access to files by clients. Slaves (DataNodes) are responsible for serving read and write requests from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master (NameNode).

Map/Reduce is also primary component of Hadoop and it also have Master-slave architecture
Master: JobTracker
Slaves: {tasktraker}……{Tasktraker}
Master {Jobtracker} is the point of interaction between users and the map/reduce framework. When a map/reduce job is submitted, Jobtracker puts it in a queue of pending jobs and executes them on a first-come/first-served basis and then manages the assignment of map and reduce tasks to the tasktrackers. Slaves {tasktracker} execute tasks upon instruction from the Master {Jobtracker} and also handle data motion between the map and reduce phases.

3.2 Programming Model of MapReduce

The computation of MapReduce takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: map and reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the reduce function. The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values together to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory.

4. Historical Model

4.1 Disadvantages of Traditional File System

Conventionally, the data were stored and processed using traditional file pro- cessing systems. In these traditional file systems, each file is independent of other file, and data in different files can be integrated only by writing individual program for each application. The data and the application programs that uses the data are so arranged that any change to the data requires modifying all the programs that uses the data. This is because each file is hard-coded with specific information like data type, data size etc. Some time it is even not possible to identify all the programs using that data and is identified on a trial-and-error basis. A file processing system of an organization is shown in figure below. All functional areas in the organization creates, processes and disseminates its own files. The files such as inventory and payroll generate separate files and do not communicate with each other. Following are the disadvantages of Traditional File System:

1) Data Redundancy: Since each application has its own data file, the same data may have to be recorded and stored in many files. For example, personal file and payroll file, both contain data on employee name, designation etc. The result is unnecessary duplicate or redundant data items. This redundancy requires additional or higher storage space, costs extra time and money, and requires additional efforts to keep all files upto-date.
2) Data Inconsistency: Data redundancy leads to data inconsistency especially when data is to be updated. Data inconsistency occurs due to the same data items that appear in more than one file do not get updated simultaneously in each and every file. For example, an employee is promoted from Clerk to Superintendent and the same is immediately updated in the payroll file may not necessarily be updated in provident fund file. This results in two different designations of an employee at the same time. Over the period of time, such discrepencis degrade the quality of information contain in the data file that affects the accuracy of reports.
3) Lack of Data Integration: Since independent data file exists, users face difficulty in getting information on any ad hoc query that requires accessing the data stored in many files. In such a case complicated programs have to be developed to retrieve data from every file or the users have to manually collect the required information.
4) Program Dependence: The reports produced by the file processing system are program dependent, which means if any change in the format or structure of data and records in the file is to be made, the programs have to modified correspondingly. Also, a new program will have to be developed to produce a new report.
5) Data Dependence: The Applications/programs in file processing system are data dependent i.e., the file organization, its physical location and retrieval from the storage media are dictated by the requirements of the particular application. For example, in payroll application, the file may be organised on employee records sorted on their last name, which implies that accessing of any employee's record has to be through the last name only.
6) Limited Data Sharing: There is limited data sharing possibilities with the traditional file system. Each application has its own private files and users have little choice to share the data outside their own applications. Complex programs required to be written to obtain data from several incompatible files.

5. Conclusion and Future Scope

5.1 Conclusion

The proposed system is based on implementation of Online Aggregation of MapReduce in Hadoop for e_cient big data processing. Traditional Map Reduce implementations materialize the intermediate results of mappers and do not allow pipelining between the map and the reduce phases. However, reducers cannot start executing tasks before all mappers have _nished. The limitation of traditional MapReduce lowers resource utilization and leads to inefficient execution for many applications. The main motivation of Map Reduce Online is to overcome these problems, by allowing pipelining between operators.

5.2 Future Scope

Future enhancement of the project work will be to enhance the map reduce functionality to achieve pipelining between jobs. Another direction would be to investigate alternative Aggregation techniques and explore the possibility for integration with large-scale processing frameworks. Another chance for future scope would be to improve the storing of data into HDFS so that processing can be benefited. There can be a limitation on how much intermediate results can be stored in Redis while processing the data. The storing capacity of the Redis can be increased to store more intermediate results.

References

[1] S.Vikram Phaneendra & E.Madhusudhan Reddy Big Data- solutions for RDBMS problems- A survey In 12th IEEE/IFIP Network Operations & Management Symposium (NOMS 2010) (Osaka, Japan, Apr 1923 2013)

[2] Kiran kumara Reddi & Dnvsl Indira Di_erent Technique to Transfer Big Data : survey IEEE Transactions on 52(8) (Aug.2013) 2348 2355
[3] Jimmy Lin MapReduce Is Good Enough? The control project. IEEE Computer 32 (2013).
[4] Hongfei Li, Usage analysis for smart meter management in Proc of 2011 IEEE Conference.
[5] Daswin De Silva, XinghuoYu,DammindaAlahakoon, and Grahame Holmes, A Data Mining Framework for Electricity Consumption Analysis From Meter Data IEEE Trans.on Ind. Informatics, vol. 7, no. 3.
[6] Yang Wang,, Qing Xia, Chongqing Kang, Secondary Forecasting Based on Deviation Analysis for Short-Term Load Forecasting IEEE Trans..on Power Systems, vol. 26, no.2.
[7] XindongWu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda , Geo_rey J.McLachlan, Angus Ng, Bing Liu, Philip S. Yu, ZhiHua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg, "Top 10 algorithms in data mining", KnowlInfSyst, 2008 14, pp. 1-37.
[8] Jiawei Han and MichelineKamber, Classi_cation and PredictioninData Mining: Concepts and Techniques 2nd ed., San Francisco, CA The Morgan Kaufmann, 2006.
[9] http://www.nyiso.com/public/markets operations/market data/load data/index.jsp
[10] Report from Pike research, http://www.pikeresearch.com/research/smartgriddata-analytics.
[11] National Climate Data Center [Online]. Available:http://www.ncdc.noaa.gov/oa/ncdc.html