Indonesian Journal of Electrical Engineering and Computer Science
Vol. 4, No. 1, October 2016, pp. 176 ~ 183
DOI: 10.11591/ijeecs.v4.i1.pp176-183

A Study on MapReduce: Challenges and Trends

Sachin Arun Thanekar*, K. Subrahmanyam, A.B. Bagwan
Department of Computer Science and Engineering, KL University,
Vaddeswaram, Guntr District, Andhra Pradesh, India
*Corresponding author, e-mail: sachin.thanekar@yahoo.co.in

Abstract

Nowadays we all are surrounded by Big data. The term 'Big Data' itself indicates huge volume, high velocity, variety and veracity i.e. uncertainty of data which gave rise to new difficulties and challenges. Big data generated may be structured data, Semi Structured data or unstructured data. For existing database and systems lot of difficulties are there to process, analyze, store and manage such a Big Data. The Big Data challenges are Protection, Curation, Capture, Analysis, Searching, Visualization, Storage, Transfer and sharing. Map Reduce is a framework using which we can write applications to process huge amount of data, in parallel, on large clusters of commodity hardware in a reliable manner. Lot of efforts have been put by different researchers to make it simple, easy, effective and efficient. In our survey paper we emphasized on the working of Map Reduce, challenges, opportunities and recent trends so that researchers can think on further improvement.

Keywords: *Big Data, Hadoop, MapReduce, HDFS, Cloud*

1. Introduction

Nowadays we all are surrounded by huge data. People upload/download videos, audios, images from variety of devices. Sending text messages, multimedia messages, updating their Facebook, WhatsApp, Twitter status, comments, online shopping, online advertising etc. generates huge data. As a result, machines have to generate and keep huge data too. Due to this exponential growth of data the analysis of that data become challenging and difficult. As shown in Figure 1 the term 'Big Data' means huge volume, high velocity, variety and veracity i.e. uncertainty of data. This big data is increasing tremendously day by day. The Big data generated may be structured data, Semi Structured data or unstructured data. Existing databases and tools are not good enough to process, analyze, store and manage such a Big Data effectively and efficiently [1-3].

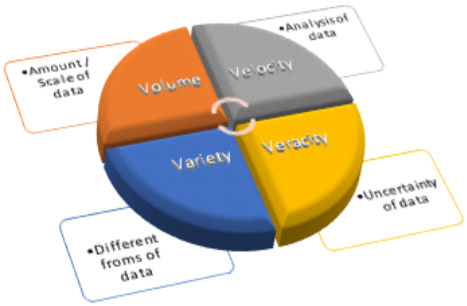

Figure 1. Four Vs of Big Data

2. Hadoop

Hadoop is an open-source, big data storage and high speed data processing software framework. As shown in Figure 2 it uses clusters of commodity hardware to store and process big data in a distributed fashion. Tremendous data storage, processing that data with high speed is making Hadoop more suitable for big data processing [4].

Received May 13, 2016; Revised August 28, 2016; Accepted September 16, 2016

Hadoop cluster is a set of commodity machines involving huge storage capabilities, networked together in one location i.e. cloud. These cloud machines are then used for Data storage and processing. From individual client's user can submit their jobs to cluster. These clients may be present at some remote locations from the Hadoop cluster. Distributed file system, faster processing, faster data transfer, good fault tolerance made Hadoop very efficient and reliable. Hadoop transfers code to data which is tiny and consumes less memory. Along with data required this tiny code get executed there itself. As data is locally available on that machine lot of time, computing resources are saved.

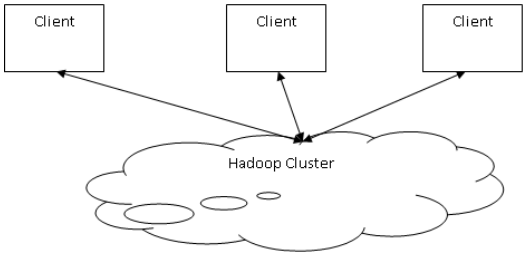

Figure 2. Hadoop Cluster

In order to provide better data availability and fault tolerance replication of data is done. User need not to worry about partitioning the data, data and task assignment to nodes, communication between nodes. As Hadoop handles it all, user can concentrate on data and operations on that data.

2.1. Important Features of Hadoop

a) Low cost

As Hadoop is an open-source framework, it is free. It uses commodity hardware to store and process huge data. Hence not much costly.

b) High Computing power

Hadoop uses distributed computing model. Due to this task can be distributed amongst different nodes and can be processed quickly. Cluster has thousands of nodes which gives high computing capability to Hadoop.

c) Scalability

Nodes can be easily added and removed. Even failed nodes can be easily identified. For all these activities very little administration is required.

d) Huge and flexible storage

Massive data storage is available due to thousands of nodes in the cluster. It supports both structured and unstructured data. No preprocessing is required on data before storing it.

e) Fault tolerance and data protection

If any node fails the tasks in hand are automatically redirected to other nodes. Multiple copies of all data are automatically stored. Due to this even if any node fails that data is available on some other nodes also.

2.2. Comparison of Hadoop with traditional RDBMS

Table 1. Hadoop-RDBMS Comparision

Sr.No.	Hadoop	RDBMS
01	Hadoop stores both structured and unstructured data.	RDBMS stores data in a structural way.
02	SQL can be implemented on top of Hadoop as the execution engine	SQL (structured query language) is used.
03	Scaling out is not that much expensive as machines can be added or removed with ease and little administration.	Scaling up (upgradation) is very expensive.
04	Basic data unit is key/value pairs.	Basic data unit is relational tables.
05	With MapReduce we can use scripts and codes to tell actual steps in processing the data.	With SQL we can state expected result and database engine derives it.
06	Hadoop is designed for offline processing and analysis of large-scale data.	RDBMS is designed for online transactions.

Table 1 is showing the difference between traditional RDBMS and Hadoop which indicates that traditional databases are not that much supportive for big data.

2.3. Hadoop System Principles

a) Scaling Out

In Traditional RDBMS it is quite difficult to add more hardware, software resources i.e. scale up. In Hadoop this can be easily done i.e. scale down.

b) Transfer code to data

In RDBMS generally data is moved to code and results are stored back. As data is moving there is always a security threat. In Hadoop small code is moved to data and it is executed there itself. Thus data is local. Thus Hadoop correlates preprocessors and storage.

c) Fault Tolerance

Hadoop is designed to cope up with node failures. As large number of machines are there, a node failure is very common problem.

d) Abstraction of complexities

Hadoop provides proper interfaces between components for proper working.

e) Data protection and consistency

Hadoop handles system level challenges as it supports data consistency.

2.4. Building blocks of Hadoop

As shown in Figure 3 a set of resident programs i.e. daemons are running in Hadoop. These daemons may be running on the same server or on the different servers in the network. All these daemons have some specific functionality assigned to them. Let us see these daemons,

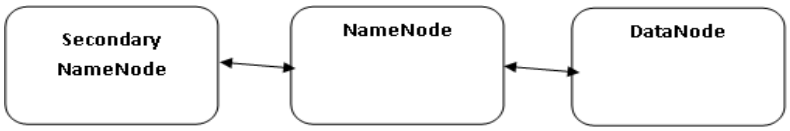

Figure 3. Hadoop cluster topology

a) Secondary NameNode

The Secondary NameNode (SNN) monitors the state of the cluster HDFS. Each cluster has one SNN which resides on its own machine also. On the same server any other DataNode or TaskTracker daemons cannot be run. NameNode also provides snapshots of the HDFS metadata at regular intervals to SNN.

b) NameNode

It is the Master node in HDFS. It provides instructions to slave (DataNode) for input output tasks of low level. The NameNode keeps track of files broken down into file blocks, nodes storing these blocks and the overall functionality of the distributed file system. NameNode is the only single point of failure component in HDFS.

c) DataNode

NameNode tells client the block addresses in DataNodes. Thus client can directly communicate to DataNodeto process the local files inside those blocks. For replication of data one DataNode may communicate with other DataNode directly. DataNodes continually provides information to NameNode regarding local changes. DataNode also receives instructions for creation or movement or deletion of blocks from the local disk.

d) JobTracker

The JobTracker determines the execution plan. It determines files to process, node assignments for different tasks, tasks monitoring etc. There is only one JobTracker daemon per Hadoop cluster. It runs on a server as a master node of the cluster.

e) TaskTracker

Individual tasks assigned by JobTracker are executed by TaskTracker. There is a single TaskTracker per slave node. TaskTracker may handle multiple tasks parallelly by using multiple JVMs. TaskTracker constantly communicates with the JobTracker. Within a specified

amount of time if the TaskTracker fails to respond to JobTracker then it is assumed that the TaskTracker has crashed. Corresponding tasks are resubmitted to other nodes in the cluster.

The interaction between JobTracker and TaskTracker is shown by Figure 4.

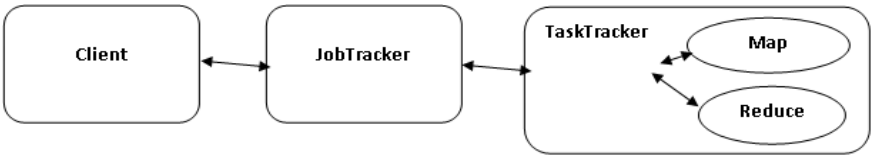

Figure 4. JobTracker and TaskTracker interaction

2.5. Hadoop Limitation

Hadoop can perform only batch processing and sequential access. Sequential access is time consuming. So a new technique is needed to get rid of this problem.

3. Hadoop Distributed File System (HDFS)

HDFS can store very large files. It supports streaming data access patterns. HDFS runs on clusters on commodity hardware. HDFS has following important characteristics,

1. Highly fault-tolerant
2. High throughput
3. Supports application with massive data sets
4. Streaming data access
5. Easily built on commodity hardware.

In HDFS a file is chopped into 64MB/128MB chunks and then stored known as blocks. As shown in Figure 5 HDFS cluster has two types of node – Master (NameNode) and Slave (DataNode). NameNode manages the namespace of the file system. It maintains the file system tree. The metadata contains the information about all the directories and files in the tree is also stored. This information is stored constantly on the local disk in the form of two files: the namespace image and the edit log.

Through the communication with the Namenode and DataNodes a client can get the access of the file system. The user code is unaware about which NameNode and DataNode are function. Only after instructions from NameNode, DataNodes store and retrieve blocks. At the same time, they are providing storage updates toNameNode.

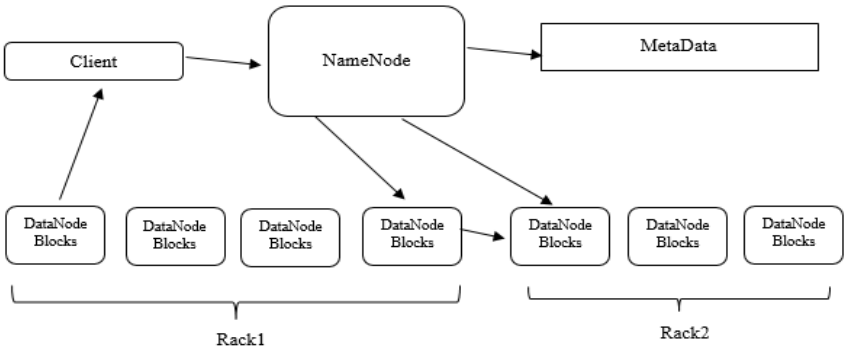

Figure 5. HDFS architecture

4. MapReduce

Huge amount of data can be easily, efficiently processed by Map Reduce with great parallelism. Moreover, these applications can run on clusters of commodity hardware which makes it suitable for scaling. Map Reduce is based on java. The Map Reduce algorithm

contains Map task and Reduce task. The general MapReduce dataflow is as shown in Figure 6. In Map task individual elements are broken down into tuples also known as key/value pairs. Reduce task further takes these intermediate tuples as an input. Then Reduce task combines it into a smaller set of tuples. Reduce task can be started only after the completion of Map task [13-16].

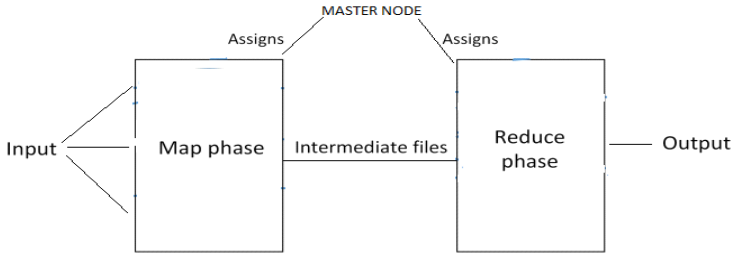

Figure 6. The general MapReduce dataflow

4.1. Map Reduce Core Functions

a) Input reader

Divides input into small parts / blocks. These blocks then get assigned to a Map function.

b) Map function

Individual elements are broken down into tuples also known as key/value pairs.

c) Shuffle and Sort

Partition function

With the given key and number of reducers it finds the correct reducer.

Compare function

Map intermediate outputs are sorted according to this compare function.

d) Reduce function

Combines intermediate tuples into a smaller set of tuples and gives it to ouput.

e) Output writer

Gives file output.

Let us understand MapReduce working with an example,
File1: "Hi Srushti Hi Shruti"
File2: "Bye Srushti Bye Shruti"
Number of occurrences of each word across different files are to be counted.
Three operations will be there as follows,

Map

Map1
< Hi, 1 >
<Srushti, 1 >
< Hi, 1 >
<Shruti, 1 >

Map2
<Bye, 1 >
<Srushti, 1 >
<Bye, 1 >
<Shruti, 1 >

Combine

Combine Map1
<Srushti, 1 >
<Shruti, 1 >
< Hi, 2 >

Combine Map2
<Srushti, 1 >
<Shruti, 1 >
<Bye, 2 >

Reduce
<Srushti, 2 >
<Shruti, 2 >
<Bye, 2 >
< Hi, 2 >

4.2. Number of Mappers and Reducers

Amount of data and the block size decides the number of Maps. Hadoop API with the setNumMapTasks(int) method provides the current number of mappers in the system.

A numbers of Reducers are directly related to the Mapper's input. As per specification it will be executed. Map Reduce command '-D mapred. reduce' can set the number of Reducers at runtime as well. 'conf. setNumReduceTasks(int)' is the method through which programmers can set it with coding.

4.3. Failure Handling in Map Reduce

Machine failure handling is very important aspect of Map Reduce as it uses hundreds or thousands of commodity machines. There are two types of basic failures as Master node failure or Worker node failure. If Master node fails, then all Map Reduce task is aborted. The whole task is to be assigned to a new Master node and again it has to be redone.

Master constantly checks the worker status in order to check failure. If worker does not respond to master in time, then it is marked as a failed. If map task worker fails, then with no consideration of any map tasks state i.e. whether it is in progress / completed etc. workers are reset to their initial idle state. The task then will be assigned to other idle worker. If reduce task fails an idle worker is chosen for reassignment of the task irrespective of any task state.

4.4. Data Storage and Replication in Map Reduce

In Map Reduce completed reduce tasks output is stored in global file system. Thus re-execution of completed reduce tasks is not required. Local disks are used to store the results of map tasks. In case of failure it can be re-executed from local disks.

4.5. MapReduce Challenges

Following are the limitations of MapReduce identified [5-8], [19],

a) No reduce can begin until all maps are complete

Map reduce reduce task starts only after finishing of the all map tasks.

b) Master must communicate locations of intermediate files.

After every map task lot of intermediate data is generated and it is to be stored and also to be informed to others.

a) Tasks scheduled based on location of data.

Lot of computation is required to provide data location and then to allocate resources on that location.

b) Before reduce finishes if map worker fails, task must be completely rerun

If master fails then the whole Map Reduce task get aborted, and it has to be redone after assigning new master node.

c) Intermediate data

Lots of intermediate data is generated. After use it is destroyed.

d) Heterogeneous data

Data is coming from different sources and different formats.

5. Results and Analysis

In this section we will discuss the work done by different researchers on different challenges.

Challenge I: No reduce can begin until all maps are complete

In Map Reduce, a reducer cannot start its processing till the completion of all the mapping tasks. The major drawback of this technique is that reducers have to wait unnecessarily. In other sense it is not an effective and efficient use of resources.

Abdel Rahman Elsayed et al., [7] done investigation on MapReduce research trends, and current research efforts. They suggested that new algorithms can be developed or framework can be modified in order to improve the performance of MapReduce.

Dhole Poonam et al., [9] proposed a solution for this problem. In their work pipelined map reduce mapper can send its output directly to reducer as an input. Thus completion time, system utilization for batch jobs are improved.

Challenge II: Master must communicate locations of intermediate files.

Diana Moise et al., [13] proposed the use of BlobSeer data management service for storing intermediate results. It is a fault-tolerant, concurrency optimized data storage layer. Thus it is an alternative for local storage of the mappers. Thus the intermediate data can be maintained separately and later on it can be used again.

Challenge III: Tasks scheduled based on location of data.

Nilam Kadale et al., [12] stated that in MapReduce framework different task scheduling methods are used to schedule the task. Survey of various task scheduling methods of MapReduce framework is done.

Jun Liu et al., [17] introduced dynamic priority scheduling and real-time prediction model. They introduced the data locality algorithm which has minimum cost and also considers a weight. Real-time prediction model is used to better serve different size jobs. They also stated that resource utilization of unexecuted tasks can be predicted by calculating the running tasks.

Bo Zhang et al., [18] proposed a feedback control loop based approach. Based on the current state of the cluster they dynamically adjusted the Hadoop resource manager configuration. They improved the performance of the system by 30% as compared to default Hadoop setup.

Muhammad Idris et al., [20] provided good survey on Hadoop MapReduce scheduling and enhancements done so far. They also discussed open issues, challenges related to the scheduling done in MapReduce.

Challenge IV: Before reduce finishes if map worker fails, task must be completely rerun.

In order to solve this problem, the same task can be executed on different nodes. The node which finishes execution first gives output. Then simply we can abort all other executions [8].

Challenge V: Intermediate data

Yaxiong Zhao et al., [10] proposed a novel Dache (Data Aware Cache) technique. Cache manager gets intermediate results from different tasks. Before executing any task, a task queries the cache manager. If it is available in cache then same is used, if not then only new computing is done. They designed a new cache request and reply protocol, cache description scheme. Through their results they have shown the significant improvement in the completion time of MapReduce jobs.

R. Udendran et al., [11] done review on the data-aware cache (Dache) for big data applications. They also stated that better life-time management of cache is required if it requires huge amount of cache.

Diana Moise et al., [13] in their paper focused on intermediate data generated in map reduce process. They proposed a new storage mechanism for intermediate data on the BlobSeer data management service. Failure handling, minimum execution time, concurrency control etc. are managed properly. Thus reduced the local storage dependency. Thus reuse of intermediate data is possible.

Mrudula Varade et al., [22] given a good comparative study of a metadata management schemes. To maintain reliability, metadata is replicated in different NameNodes. Log replication technology is used for replication. To maintain replication consistency Paxos algorithm is used.

Challenge VI: Heterogeneous data

Jun Qu et al., [21] proposed a new framework called as Map-Reduce-Merge. Web heterogeneous data processing is efficiently done. They done their experiments on features of web data.

Nenavath Srinivas Naik et al., [23] proposed Map Reduce Reinforcement Learning scheduler. This scheduler suggests the re-execution of slower tasks to other available nodes by observing the system state and task execution. Thus faster execution of the task can be done. No prior knowledge of the system is required. Thus overall job completion time is significantly minimized.

6. Conclusion

Big data is increasing tremendously day by day which gave rise to new difficulties and challenges as we have to store, process, analyze, modify such a huge amount of data. Existing databases, tools are not good enough to handle this issue. In our paper we have provided

overview of the big data, its challenges with respect to Map reduce. Many efforts taken to reduce those challenges are also discussed. Thus better planning of Big Data projects can be done. For researchers' opportunities for future research can be identified.

References

[1] Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Khan S. The rise of big data on cloud computing: Review and open research issues. *Elsevier Information systems*. 2015; 47: 98-115.

[2] Wang L, Alexander CA. Big Data: Infrastructure, technology progress and challenges. *Journal of Data Management and Computer Science*. 2015; 2(1): 1-6.

[3] Wei Fan, Albert Bifet. Mining Big Data: Current Status, and Forecast to the Future. *SIGKDD Explorations*. 2012; 14(2).

[4] Priyaneet Bhatia, Siddarth Gupta. Correlated Appraisal of Big Data, Hadoop and MapReduce. *Advances in Computer Science: An International Journal.* 2015; 4(4): 16.

[5] Du Zhang. *Inconsistencies in Big Data*. 12th IEEE Int. Conf. on Cognitive Informatics & Cognitive Computing. 2013.

[6] Noh Kyoo-sung, Lee Doo-sik. Big data Platform Design and Implementation Model. *Indian Journal of science and technology.* 2015; 8(18).

[7] Abdel rahman Elsayed, Osama Ismail, Mohamed E El-Sharkawi. Map Reduce: State-of-the-Art and Research Directions. *International Journal of Computer and Electrical Engineering*. 2014; 6(1).

[8] K Grolinger, M Hayes, W Higashino, A L'Heureux, DS Allison, MAM Capretz. *Challenges for MapReduce in Big Data*. IEEE 10th 2014 World Congress on Services (SERVICES 2014). Alaska USA. 2014.

[9] Dhole Poonam B, Gunjal Baisa L. Survey Paper on Traditional Hadoop and Pipelined Map Reduce. *International Journal of Computational Engineering Research*. 2013; 03(12).

[10] Yaxiong Zhao, Jie Wu, Cong Liu. Dache: A Data Aware Caching for Big-Data Applications Using the MapReduce Framework. *Tsinghua Science and Technology*. 2014; 19(1): 39-50.

[11] R Udendran, Govindaraj, P Kannan. Review Paper on Data-aware Caching for Big Data Applications. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2015; 5(3).

[12] Nilam Kadale, UA Mande. Survey of Task Scheduling Method for Map Reduce Framework in Hadoop. *International Journal of Applied Information Systems (IJAIS).* 2013.

[13] Diana Moise, Thi-Thu-Lan Trieu, Gabriel Antoniu, Luc Boug. *Optimizing Intermediate Data Management in Map Reduce Computations*. 1st International Workshop on Cloud Computing Platforms. 2011.

[14] Shafali Agarwal, Zeba Khanam. Map Reduce: A Survey Paper on Recent Expansion. *International Journal of Advanced Computer Science and Applications*. 2015; 6(8).

[15] Naga Malleswari TYJ, Vadivu G. MapReduce: A Technical Review. *Indian Journal of science and technology*. 9(1): 1-6.

[16] Ananthi Sheshasayee, JVN Lakshmi. Comparison of Machine Learning Algorithm on Map Reduction for Performance Improvement in Big Data. *Indian Journal of science and technology*. 2015; 8(29).

[17] Jun Liu, Tianshu Wu, Ming Wei Lin, Shuyu Chen. An Efficient Job Scheduling for Map Reduce Clusters. *International Journal of Future Generation Communication and Networking*. 2015; 8(2): 391-398.

[18] Bo Zhang, Filip Krikava, Romain Rouvoy, Lionel Seinturier. *Self- configuration of the Number of concurrently Running Map Reduce Jobs in a Hadoop Cluster*. ICAC 2015. 2015: 149-150.

[19] VA Ayma, RS Ferreira, P Happ, D Oliveira, R Feitosa, G Costa, A Plaza, P Gamba. *Classification Algorithms for big data analysis, A mapreduce approach.* The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-3/W2, Joint ISPRS conference Germany. 2015.

[20] Muhammad Idris, Shujaat Hussain, Maqbool Ali, Arsen Abdulali, Muhammad Hameed Siddiqi, Byeong Ho Kang, Sungyoung Lee. Context-aware scheduling in Map Reduce: a compact review. *Concurrency and Computation: Practice and Experience*. 5332-5349.

[21] Jun Qu, Chang-Qing Yin, Shangwei Song. The Optimization and Improvement of MapReduce in Web Data Mining. *International Journal of Future Generation Communication and Networking*. 2015; 8(2): 391-398.

[22] Mrudula Varade, Vimla Jethani. Distributed meta data management scheme in HDFS. *International Journal of Advanced Computer Science and Applications*. 2015; 6(8).

[23] Nenavath Srinivas Naik, Atul Negi, VN Sastry. *Performance Improvement of MapReduce Framework in Heterogeneous Context using Reinforcement Learning*. Elsevier ISBCC'15. 2015.