IMPROVED MAPREDUCE ALGORITHM WITH EFFICIENCY IN STRUCTURE AND RELATIONS IN THE LARGE BODY OF THE DATA (BIGDATA)

Ph.D. Synopsis

For the Degree of Doctor of Philosophy
in Computer Engineering

Submitted by:
Maulik Vipinchandra Dhamecha
Enrolment No: 159997107017
Batch-2015
(Computer/IT Engineering)

Supervisor:
Dr. Tejas Patalia
Professor & Head,
Computer Engineering Department,
VVP Engineering College,
Rajkot, Gujarat, India.

DPC Members:

Dr. Amit Ganatra
Professor,
Principal of DEPSTAR,
Dean of FTE, Charusat
Changa, Gujarat, India

Dr. Atul Gosai
Professor
Computer Science Department,
Saurastra University,
Rajkot, Gujarat, India

Submitted to:

Gujarat Technological University, Ahmedabad

Index

1. Abstract

"Big Data" refers to enormous amounts of unstructured data produced by high-performance applications falling in a wide and heterogeneous family of application scenarios: from scientific computing applications to social networks, from e-government applications to medical information systems, and so forth. Recent research in big data has shown that the amount of data continues to increase at an exponential rate. To process a massive amount of data on computing clusters, you need a powerful computing model like Hadoop and MapReduce. MapReduce framework is a programming model that processes terabytes of data in a very less amount of time. To achieve excellent performance, big data requires proper scheduling. Scheduling Technique is used to reduce starvation, increase the usage of resources, and also to assign jobs for available resources [1].

For the Hadoop-MapReduce model, various scheduling algorithm has been developed, which differs widely in design, behavior, and handling various issues. First In First Out (FIFO) scheduling is the simplest and most efficient among the scheduling algorithm. A significant drawback of FIFO scheduling is the poor response time for short jobs in comparison to large jobs and low performance in handling multiple types of jobs. Existing resource allocation scheduling does not take the weight of each job into consideration which leads to unbalanced performance among nodes. The objective of this research is to provide a strategy that balances resource allocation between short jobs and long jobs, considering the weight of each job. This study provides a faster response time to smaller jobs to achieve better performance in scheduling [1].

In the current scenario, there is no consideration of system load during allocation to task trackers and if one task tracker is slow, it can delay the whole MapReduce job. So, for this kind of system, we have to calculate a load of all data nodes as well as the speed of every task tracker. Assign the task to that node whose speed of task tracker is better to compare to other data nodes and maintain the load balancing.

2. A brief description of the state of the art of the research topic

Konstantin Shvachko has proposed that Hadoop is a framework by the Apache project, which is used to make design and analysis on the set of a larger amount of data. Hadoop is used along with the mapreduce procedure [2,3]. With the use of map-reduce data can be distributed over many systems and processes in a parallel manner. HDFS is a Hadoop distributed file system that stores Metadata in node namely name node, and performed some process with data which are stored at data node. This node is not provided with security mechanisms so replication of data node is used. To provide a user facility to use data to its nearby place, this is very important. Also due to replication node failure can be easily handled [3].

In the architecture of HDFS, there are two main nodes available. First is NameNode and the second is Data node. All files and directories are stored in NameNode. HDFS client first contacts to Name node and then find appropriate DataNode to the user. Then all processing will be done in a pipeline manner. In every step of implementation, DataNode will handshake with the Name Node to check its software version and its respected id. If any new DataNode is generated then it will be allowed to any cluster and assigns a new id of NameNode [3].

Alfredo Cuzzocrea Yeol proposed the big data revolution. In that data stored in the underlying layer of all these application scenarios have some specific characteristics in common, among which we recall: (i) large-scale data, which refers to the size and the distribution of data repositories; (ii) scalability issues, which refers to the capabilities of applications running on large-scale, enormous data repositories (i.e., big data, for short) to scale over growing-in-size inputs rapidly; (iii) supporting advanced Extraction-Transformation-Loading (ETL) processes from low-level, raw data to somewhat structured information; (iv) designing and developing easy and interpretable analytics over big data repositories to derive intelligence and extract useful knowledge from them [1,4].

Subramaniyaswamy proposed a research work says that big data is getting complex due to its volume, velocity, and variety of data. Among all data, huge data is unstructured and needs to be managed properly for our required needs. For processing a huge amount of data, older systems are not sufficient so new technologies are used like map-reduce and Hadoop. Unstructured data cannot be processed directly. So it is required to convert unstructured data to structured data and then required operation will be applied [5].

Hadoop map-reduce is used to process terra bytes of data and convert unstructured data to structured data. Hadoop, the open-source framework uses the hash algorithm to separate data and perform analysis. Map-reduce is used for filtering the data [6]. The map function will filter the data in an appropriate way and the reduce function will aggregate the data. MRAP means Map Reduce Access Patterns are used in reformatting of data. In this paper collaborative filtering and sentiment analysis techniques are used. Sentiment analysis is used for the prediction of data. There are many collaborative filtering techniques available among that we use maven and mahout tools which are used for collaborative filtering. Mahout is highly scalable in a large amount of data sets and maven is used to build a project and manage it. So this technique helps to generate recommendations for any large amount of input source data [7].

3. Definition of the Problem

Scheduling in MapReduce is a very complex and time-critical job. It is very dependable for the fast execution of each and every process. Currently, MapReduce job can't calculate every data node's load with the use of available scheduling algorithm. So, need to develop a scheduling algorithm for MapReduce job for efficiency improvement of process in large amount data.

4. Objective and Scope of the work

- Calculate the load of all the data nodes.
- Find out the fast data node and also the weak data node.
- Assign a task to that node which has a minimum load compared to others to maintain the load balancing.

5. Original contribution by the thesis.

In the MapReduce, the current system load is not calculated during process allocation to data node and suppose any data node is slow in working, the whole MapReduce job can be delayed. So for this kind of system, initially we have to calculate the load of all data nodes and consider the speed of every data node. After that assign task to the only node whose speed of task tracker is better compared to other data nodes to maintain the load balancing. Here is the proposed algorithm for the solution.

Algorithm for computing Architecture Slice using

```
Input: String name, String document
Output: Total count of String Name
Process:
For each process,
        // Apply Job allocation
        [n] = Job tracker
        allocate first process to Name Node
Check Availability
Apply scheduling algorithm
for (n=0;n=[n];n++)
        if n==0;
        return No load
        else
        return current load

// name: document or file name
 // document: document or file contents
function map (String name [n], String document [n+1]):
        for each word w in document or file:
 emit (w, 1)
function shuffle (String name [n], String document [n+1]):
  w=0
         for same word w in document or file:
  w += w
  return w;

function reduce(String word[n], Iterator partial counts[n+1]):
sum = 0
  for each pc in partial Counts:
    sum += total count
  emit (word, sum)”
```

6. The methodology of Research, Results with Comparisons

The below figure shows the flow of the proposed system.

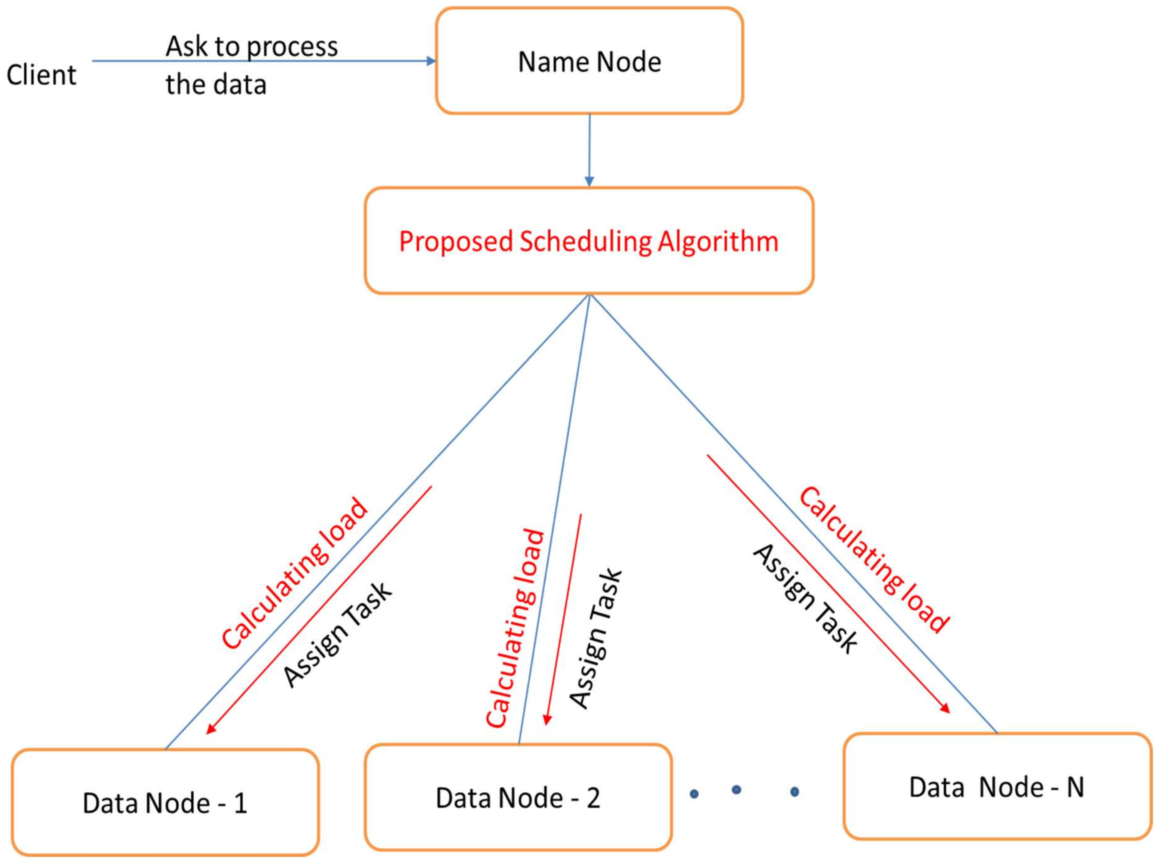

Figure (1) : Proposed flow of work

With reference to the above-proposed diagram, we have to consider certain parameters for scheduling. So, below-given table shows the comparative study of different parameters.

Sr. No.	**Parameters / Algorithm**	**Round Robin**	**Randomize**	**Central Manager**	**Lest connection**	**Local Queue**	**Central Queue**
1	Centralize / Decentralize	Decentralize	Decentralize	Decentralize	Decentralize	Centralize	Centralize
2	Nature	Static	Static	Static	Dynamic	Dynamic	Dynamic
3	Fault Tolerance	No	No	Yes	No	Yes	Yes
4	Response Time	Less	Less	Less	Less	More	More
5	Process Migration	No	No	No	No	Yes	No
6	Overhead	Low	Low	Low	High	High	High
7	Resource Utilization	Less	Less	Less	More	More	Less
8	Adaptability	Less	Less	Less	More	More	Less
9	Waiting Time	More	More	More	Less	Less	Less
10	Retrieval Time	Slow	Slow	Fast	Slow	Slow	Slow
11	Seek Time	Less	Less	Less	More	More	Less
12	Latency Time	High	High	Low	High	High	High

Table (1) : Comparative study of different parameters

With reference to the above parameter I have applied the proposed algorithm on data set named "Current enrollment across different education level in Higher Education" which is 10.80 GB in size and taken from the government of India's database. After applying this algorithm I got the following results.

Sr. No.	Algorithm	Response Time (Second)	Average Waiting Time (Second)	Retrieval Time (Second	seek time (Second)	Average Latency (Second)
1	Round Robin	1054.313	46.23	1149.643	2013.296	2108.626
2	Randomized	936.278	44.22	1031.608	1777.226	1872.556
3	Least Connection	1022.358	45.01	1117.688	1949.386	2044.716
4	Local Queue	1221.988	49.32	1317.318	2348.646	2443.976
5	Central Queue	2562.12	54.56	2657.450	5028.910	5124.240
6	Proposed Scheduler	962.289	47.62	1057.619	1829.248	1924.578

Table (2) : Result of the proposed algorithm with different parameters

Here is the comparative study of the above result compare to the threshold scheduler as a base.

Sr. No.	Algorithm	Response Time (%)	Average Waiting Time	Retrieval Time	seek time	Average Latency
1	Round Robin	9%	18%	8%	9%	9%
2	Randomized	11%	4%	10%	12%	11%
3	Least Connection	3%	3%	3%	3%	3%
4	Local Queue	-16%	-7%	-15%	-17%	-16%
5	Central Queue	-143%	-18%	-131%	-150%	-143%
6	Proposed Scheduler	9%	-3%	8%	9%	9%

Table (3) : Result of the proposed algorithm with different parameters

Here is the graphical analysis of the result.

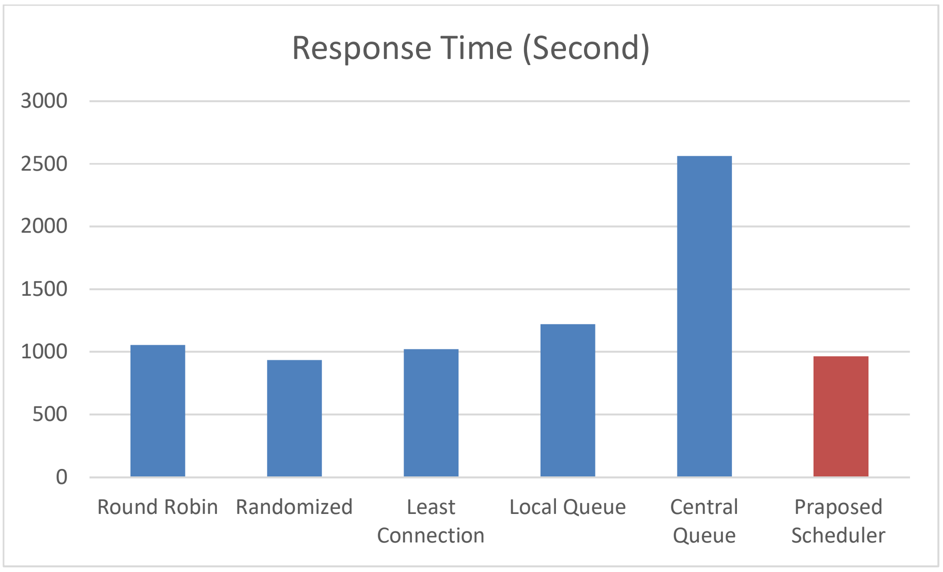

Figure (2) : Comparison of Response Time (Second)

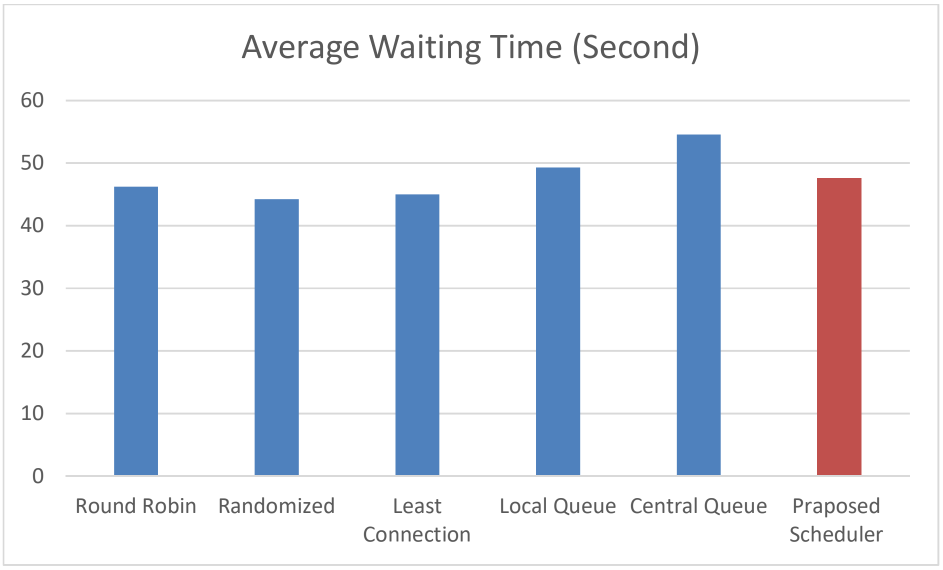

Figure (3) : Comparison of Waiting Time (Second)

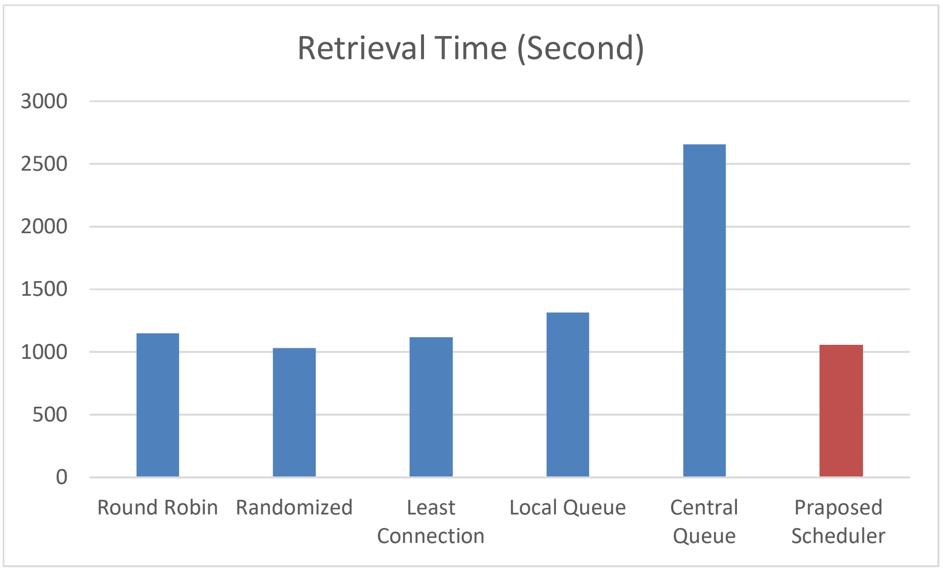

Figure (4) : Comparison of Retrieval Time (Second)

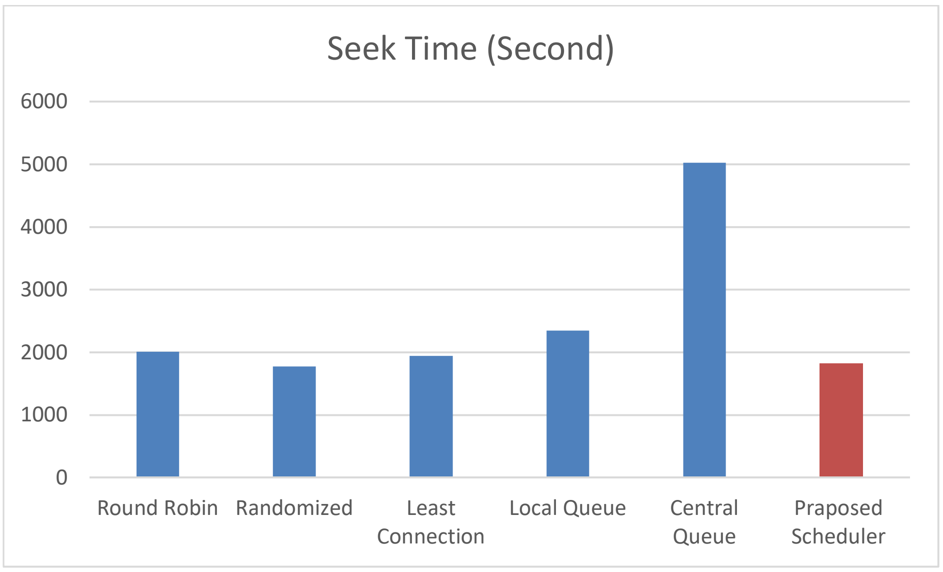

Figure (5) : Comparison of Seek Time (Second)

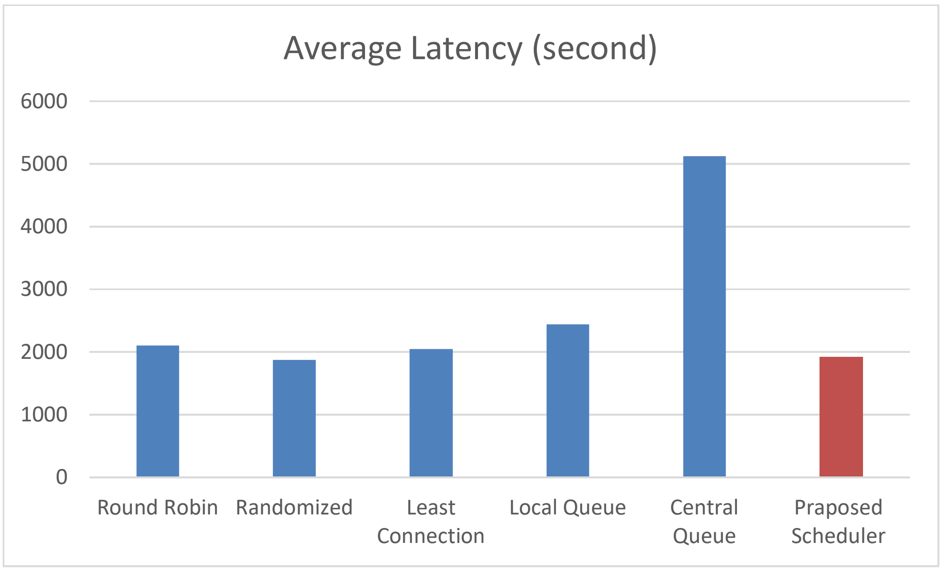

Figure (6) : Comparison of Average Latency (Second)

7. Achievements with respect to objectives

With the use of this new algorithm, we can find which node has a minimum load compared to others and thus Name node assigning the task to that node to maintain the load balancing. So, at the end load balancing occurs in cluster of nodes and process allocation becomes compatible.

8. Conclusion

In a distributed file system, whenever we deal with any number of nodes, and a large amount of data we are very much concerned about fast processing and getting in time results. In HDFS, whenever the flow of the process is large at name node, the process can be directly assigned to their data node in a serial manner and load distribution become unbalance. In this research, I try to overcome this problem. In this research, the proposed algorithm manages the load balancing at name node and distribute that node only to those data node which has a minimum load. This algorithm is applied in a homogenous environment so, the result is comparable with the same configured load and hence get the accurate balanced load distribution system.

9. Copies of papers published and a list of all publications arising from the thesis

The publication detail for the work is as below.

[1] Maulik Dhamecha, Dr. Tejas Patalia, "Fundamental Survey of Map Reduce in Bigdata with Hadoop Environment", Spinger – CCIS (2018).

[2] Maulik Dhamecha, Dr. Tejas Patalia, "Comparative study of Dynamic Load Balancing algorithm in large scale data (Big data)", International Journal of Advanced Science and Technology (2020)

[3] Maulik Dhamecha, Dr. Tejas Patalia, "Scheduling issue for Dynamic Load Balancing of mapreduce in large scale data (Big data)", Journal of Xidian University (2020)

10. References

Books:

[1] G. K. Gupta, "Introduction to Data Mining with Case Studies", PHI, 2006.

[2] Sima Acharya, Subhashini Chhellappan, "BIG Data and Analytics", Willey

[3] Chris Eaton,Dirk derooset al., "Understanding Big data ", McGraw Hill, 2012.

[4] "Big Data Now", OReilly, 20012.

Research Paper:

[1] Acher M, "Separation of Concerns in Feature Modeling : Support and Applications", Proceedings of the 11th annual international conference on Aspect oriented Software Development (2017)

[2] Agrawal H, "Incremental Regression Testing", Proceedings of the Conference on Software Maintenance, IEEE (2015)

[3] Agrawal H., Horgan J.R, "Dynamic Program Slicing", In Proceedings of the ACM SIGPLAN'90 Conference on Programming Language Design and Implementation

[4] Acher, M. et al., 2011. Slicing feature models. In 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). IEEE

[5] Monika Kushwaha, Saurabh Gupta, “Various Schemes of Load Balancing in Distributed Systems- A Review”, ijsret-2015

[6] CHEN Jinming, YUAN Yubo, GUO Yajuan, SUN Jian, LI Bin, “Research and Application of Load Balance Approach for Distribution Based on Big Data”, CICED- 2016

[7] Dhole Poonam B, Gunjal Baisa L, “Survey Paper on Traditional Hadoop and Pipelined Map Reduce”, IEEE-2017

[8] Feilong Tang, Laurence T. Yang, Can Tang, Jie Li and Minyi Guo, “A Dynamical and Load-Balanced Flow Scheduling Approach for Big Data Centers in Clouds”, IEEE-2016

[9] Arnab K. Paul, Arpit Goyal, Feiyi Wang, “I/O Load Balancing for Big Data HPC Applications”, IEEE-2017

[10] Barry Wilkinson and Michael Allen “Techniques and Applications using Networked Workstations and Parallel Computers”, Prentice Hall

[11] C.Jayashri, P.Abitha, S.Subburaj, S.Yamuna Devi, “Big Data Transfers through Dynamic and Load Balanced Flow on Cloud Networks”, AEEICB-17

[12] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, Yahoo!, Sunnyvale, California USA, “The Hadoop Distributed File System”, IEEE-2010

[13] Alfredo Cuzzocrea, Yeol Song, Karen C. Davis, “Analytics over Large-Scale Multidimensional Data: The Big Data Revolution”, ACM-2011

[14] Subramaniyaswamy, Vijayakumar, Logesh, Indragandhi, “Unstructured Data Analysis on Big Data using Map Reduce”, ScienceDirect-2015

[15] “Oracle: Big Data for the enterprise” An Oracle whitepaper, Jan 2012

[16] Dayton L. Jones, “Big Data Challenges for Large Radio Arrays”, IEEE-2012 Jeffrey Dean and Sanjay Ghemawat, “MapReduce Simplified Data Processing on Large Clusters”, OSDI-2004

[17] Ge Song, “A Hadoop MapReduce Performance Prediction Method”, IEEE-2013

[18] Anucha Tungkasthan, Wichian Premchaiswadi, “A parallel processing framework using MapReduce for content-based image retrieval”, IEEE-2013

[19] Weijia xu, Wei luo, “Anasisis and optimization of data import with hadoop”, IEEE-2016

[20] Maitrey S, Jha c.k., "Handling big data efficiently by using map reduce technique", IEEE- 2015

[21] Puneet Agarwal, Gautam Shroff, Pankaj Malhotra, "Approximate Incremental Big-Data Harmonization", IEEE-2017

[22] Guozhang Wang, Marcos Vaz Salles, "Behavioral Simulations in MapReduce", IEEE-2010

[23] J. Dean, S. Ghemawat, "Simplified Data Processing on Large Clusters", 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, (2014)

Web Reference:

[1] Structured, semi structured and unstructured data – Jeremy Ronk

[2] Extract, transform, load - Wikipedia, the free encyclopedia

[3] www.ibm.com/big-data/

[4] www.ieeexplore.ieee.org/

[5] https://hadoop.apache.org/