# Design of an Optimal Data Placement Strategy in Hadoop Environment

Shah Dhairya Vipulkumar[1], Saket Swarndeep[2]

[1] *PG Scholar, Computer Engineering, L.J.I.E.T., Gujarat, India*
[2]*Assistant Professor, Dept. of Computer Engineering, L.J.I.E.T., Gujarat, India*

**ABSTRACT**

*The MapReduce framework has gained wide popularity as a scalable distributed system environment for efficient processing of large scale data of the order of Terabytes or more. Hadoop, an open source implementation of MapReduce coupled with Hadoop Distributed File System, is widely applied to support cluster computing jobs requiring low response time. The current Hadoop implementation assumes that nodes in the cluster are homogenous in nature. Data placement and locality has not been taken into account for launching speculative processing tasks. Furthermore, every node in the cluster is assumed to have same CPU and memory capacity despite some of the nodes being configured using vastly varying generation of hardware. Unfortunately, both the homogeneity and data placement assumptions in Hadoop are optimistic at best and unachievable at worst, potentially introducing performance problems in Hadoop clusters at data centres. This dissertation explores the Hadoop data placement policy in detail and proposes a modified data placement approach that increases the performance of the overall system. Also, the idea of placing data across the cluster according to the processing capacity utilization of the nodes is presented, which will improve the workload processing in Hadoop environment. This is expected to reduce the response times for the applications in large-scale Hadoop clusters.*

**Keyword:** - Hadoop, Data Placement, Data-Locality, Distributed Computing, Big Data, Cloud Computing.

## 1. INTRODUCTION

We live in a world that is driven by Data. Data intensive applications are on the rise. Since the communication paradigm of the Internet is sufficiently open and powerful, the World Wide Web, in the past decade or so, has been adopted as the ideal platform for developing data intensive applications. Data intensive applications like data mining and web indexing need to access ever expanding data sets ranging from a few gigabytes to several terabytes or even petabytes. Google, for example leverages the MapReduce model to process approximately twenty petabytes of data per day in a parallel scenario. The MapReduce programming framework can simplify the complexity by running parallel data processing functions across multiple computing nodes in a cluster as scalable MapReduce helps programmers to distribute programs across nodes as they as executed in parallel. MapReduce automatically gathers

results from the multiple nodes and returns a single result or set. More importantly, MapReduce platforms offer fault tolerance that is entirely transparent to programmers. This makes MapReduce a practical and attractive programming model for parallel data processing in high performance cluster computing environments.

Apache Hadoop is an open-source software framework used for distributed storage and processing of very large data sets. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are a common occurrence and should be automatically handled by the framework.

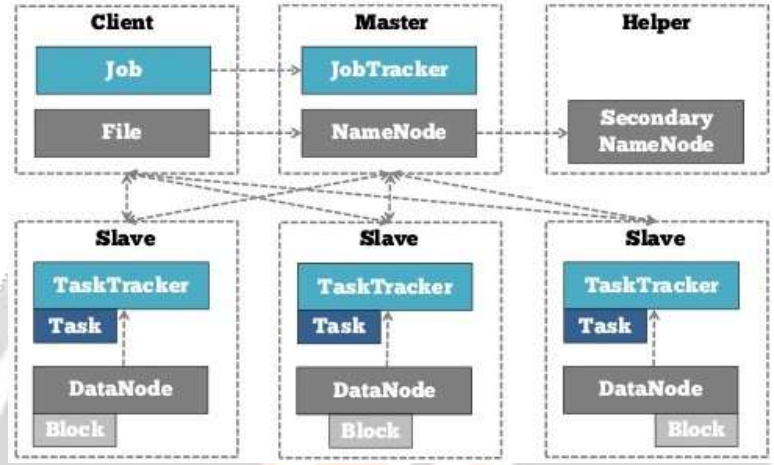


Figure 1. Hadoop Multi-Node Cluster Architecture

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data parallelly. This approach takes advantage of data locality – nodes manipulating the data they have access to – to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The default Hadoop implementation assumes that the computing nodes are heterogeneous in nature. Moreover, the input size data blocks are split into equal sized data blocks with a given value, and are allocated to nodes for processing. This strategy works well in homogeneous cluster. But it performs poorly in heterogeneous clusters because of the heterogeneity (in terms of processing, memory, throughput, I/O, etc.) of the nodes capabilities to process as well the considering the fact that Big-data is also heterogeneous and complex, there is considerable difference in the performance of nodes to process the data.

Hadoop assumes that the computing capacity of each node in a cluster is the same. In such a homogeneous environment, each node is assigned to the same load, and thus it can fully use the resources in the cluster. There would not be many nodes that are idle or overloaded. However, in real-world applications, clusters are often worked in a heterogeneous environment [research papers]. In such a cluster, there is likely to be various specifications of PCs or servers, which causes the abilities of the nodes to differ. If such a heterogeneous environment still uses the original Hadoop strategy that distributes data blocks into each node equally and the load is also evenly distributed to each node, then the overall performance of Hadoop may be reduced.

## 2. RELATED WORK

[1] Suhas V. Ambade, Prof. Priya R. Deshpande have proposed "Heterogeneity-based files placement in Hadoop cluster"

This paper proposes a data placement method based on complexity of the input file. But this is not optimal for intensive large-scale processing in clusters. Authors have proposed format specific data distribution amongst the nodes by using computational power of each node. They have Proposed Customized Block Placement policy. At the time of writing data, the DataNodes are selected based on the complexity of data and computation power of each node. Complexity will be dependent on the file type. According to the type or complexity of input file, the DataNode is selected to place the file. The Max CPU MHz is considered as the computational power of node for placing the file.

[2] Chia-Wei Lee, Kuang-Yu Hsieh, Sun-YuanK Hsieh, Hung-Chang Hsiao have proposed "A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments"

Here, the authors have proposed Dynamic Data Placement strategy. Data placement algorithm to resolve the unbalanced node workload problem. Dynamically adapt and balance data stored in each node based on the computing capacity of each node in a heterogeneous Hadoop cluster.

[3] Jia-xuan WU, Chang-sheng ZHANG, Bin ZHANG, Peng WANG have proposed "A New Data-Grouping Aware Dynamic Data Placement Method that Take into Account Jobs Execute Frequency for Hadoop"

In this paper, the authors have proposed a new Data-Grouping-Aware Dynamic (DGAD) data placement method based on the job execution frequency. Firstly, a job access correlation relation model is built among the data blocks according to the relationships provided by the records about historical data block access. Then BEA clustering algorithm is used to divide data blocks into clusters according to the job access correlation relation model among the data blocks. These correlated data blocks within a cluster are put on different nodes.

[4] B Ganesh Babu, Shabeera T P, Madhu Kumar S D have proposed "Dynamic Colocation Algorithm for Hadoop

In this paper, the authors have proposed Dynamic Colocation Algorithm. Algorithm based on combinations of all datasets, for efficient data placement. According to the authors, in real time, one task may require many datasets for execution and one dataset may also be accessed by many tasks. In their strategy, they try to keep these datasets in one DataNode or nearby DataNodes, so that when tasks were scheduled to this DataNode, most if not all of the data they need to execute the task are stored locally.

[5] Vrushali Ubarhande, Alina-Madalina Popescu, Horacio Gonzalez-Velez have proposed "Novel Data-Distribution Technique for Hadoop in Heterogeneous Cloud Environments"

In this paper, the authors have proposed the distribution of data blocks within the Hadoop cluster based on the computing capability of each slave node. First, a *Speed Analyzer* component *is created on NameNode,* and installed an executed on each slave node. The master node reads the response time taken by each slave node from the respective log file, and then creates a file with the computing ratio. This file is fed to the *Data Distribution* component through the Hadoop Distributed File System. Speed Analyzer agent is created to measure the processing capacity of each slave node.

### A. Comparative Analysis

| Sr. No. | Paper Title | Methods/ Techniques | Advantages | Limitations |
|---|---|---|---|---|
| 1. | Heterogeneity-based files placement in Hadoop cluster | Customized format specific Placement Policy/ Modified Hadoop | File format specific intelligent Data Placement | Simple file format specific data placement is not optimal approach for intensive processing in large-scale clusters |
| 2. | A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments | Dynamic Data Placement (DDP), Computing Capacity based Placement | Can dynamically adapt and balance data stored in each node based on the computing capacity of each node, reduce execution time, improve Hadoop performance | Dynamic placement might induce run-time processing skew. |
| 3. | A New Data-Grouping-Aware Dynamic Data Placement Method that Take into Account Jobs Execute Frequency for Hadoop | Data-Grouping Aware Dynamic Placement (DGAD), Clustering based Data Placement, Bond Energy Algorithm (BEA), Clustered Matrix | Improves Hadoop performance considerably | Inefficient utilization of cluster resources |
| 4. | Dynamic Colocation Algorithm for Hadoop | Dynamic Colocation Algorithm | Better than BEA as it doesn't need extra step for creating combinations of datasets | Inefficient use of cluster resources |
| 5. | Novel Data-Distribution Technique for Hadoop in Heterogeneous Cloud Environments | Speed Analyzer, Processing Capability based Distribution | Understands processing capability and assigns data block as per processing speed | Straggler node issue |

Table 1. Literature Comparison

## 3. PROPOSED WORK

### A. Proposed System Architecture

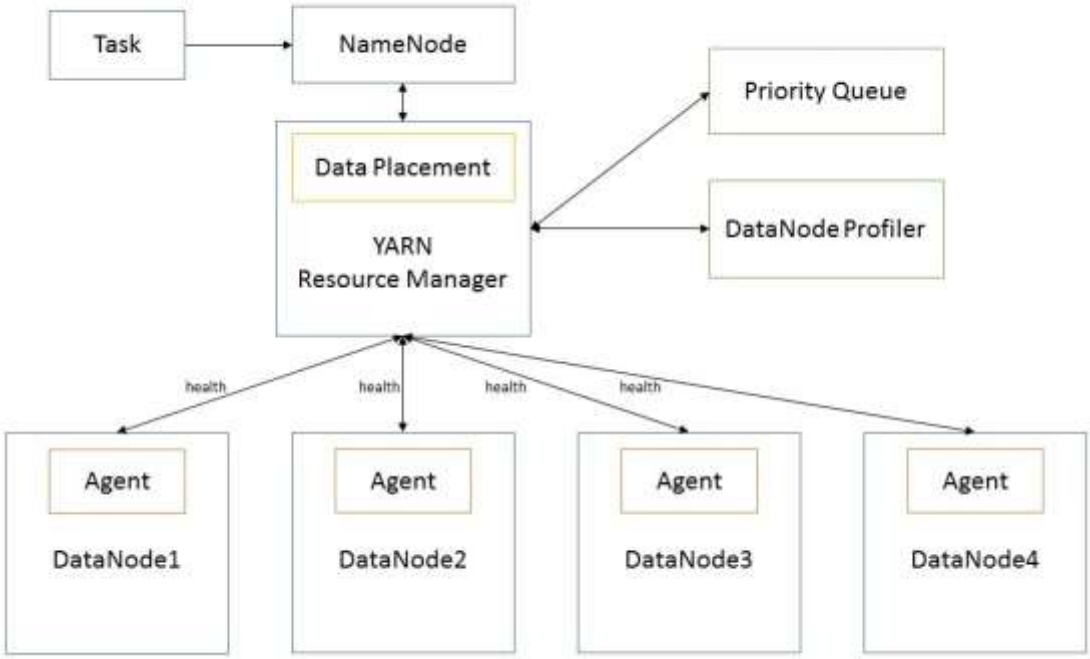


Figure 2. Proposed System Architecture

*Resource Manager*: Resource Manager will be responsible for managing the resources like CPU and memory. The Data Placement module will be integrated with YARN Resource Manager. The Resource Manager will keep health check of the DataNodes by pinging onto the Agent module of DataNodes after a fixed time interval.

*Agent*: The agent will respond to the ping from the Resource Manager with the health check information of the DataNodes. Also, it will forward the CPU and memory details to the Resource Manager.

*DataNode Profiler*: This will maintain and keep the DataNode information from the Resource Manager updated. It will do DataNode profiling based on the CPU usage of the nodes.

*Priority Queue*: This Queue will keep the profiled DataNode information in a sorted manner. The information from this queue will be fed to the Data Placement module in the Resource Manager which will be responsible for optimal data placement.

**B. Proposed Flow for Choosing the Target DataNode**

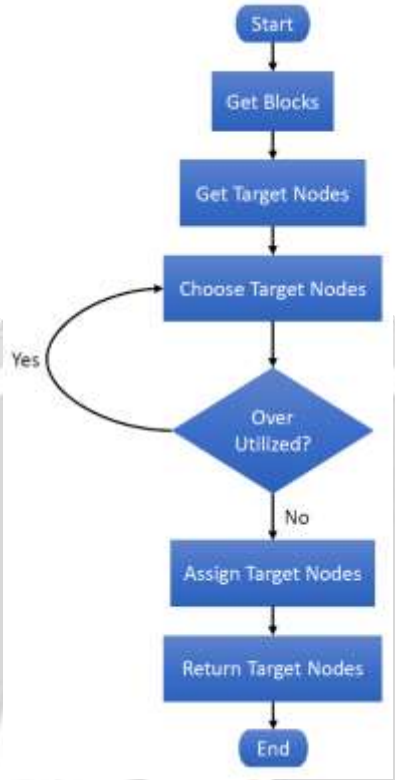


Figure 3: Target DataNode Chooser Execution Flow

**C. Data Placement Strategy**

*Phase 1: Computing the CPU consumption of nodes and creating priority queue*

We integrate a CPU consumption computing into the Resource Manager (RM). This module will take care of computing the CPU consumption of the nodes in percentage (%). This process will be taken care by the DataNode profiler module. The data from this module will be fed to the YARN Resource Manager. The Data Placement module integrated into YARN RM will create a priority queue and maintain and update it from time to time. This priority queue will be clubbed with the YARN RM to make data distribution decisions in the next phase.

*Phase 2: Distributing the data based on the priority queue*

The RM will identify nodes as either under-utilized or over-utilized. It is obvious that we should place less data blocks on the over-utilized nodes and more data blocks on the under-utilized nodes. We will balance the loads and hence create a dynamic data balancing approach by moving the data from heavily consumed nodes to the lightly consumed nodes. This will be done by moving the input file fragments from the Source node which will fall in the over-utilized category to the Destination node which will fall in the under-utilized category. This data distribution will be taken care by the Data Placement module in the Resource Manager. More workloads will be given to the DataNodes with lower CPU usage and less or no workload will be given to the DataNodes with higher CPU usage.

## 4. ENVIRONMENT SETUP AND IMPLEMENTATION

A multi-node Hadoop cluster was created over the Windows Azure cloud. For that, we created 3 virtual machines with Ubuntu 12.04 OS. A virtual network was created among the master and slave nodes. On top of it, we installed Hadoop 2.6.0. One node would act as a NameNode and one DataNode was created on all the nodes respectively. On master node, we have NameNode and DataNode processes running. While on the slave nodes, we have DataNode process running.

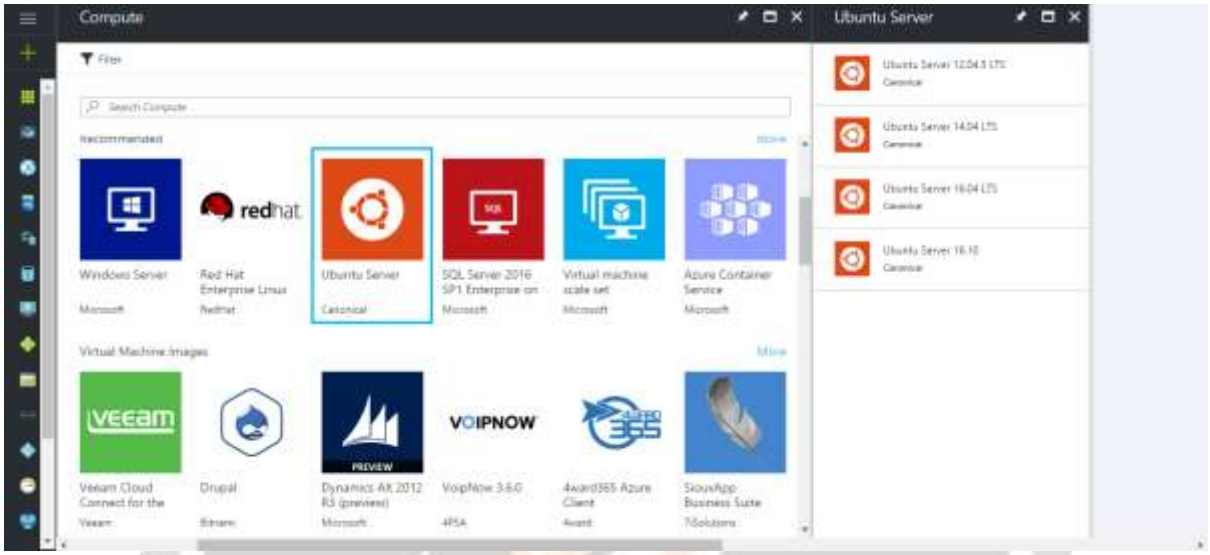


Figure 4. Creating Ubuntu Server on Virtual Machine

The above image depicts the creation of Ubuntu servers on the VMs using Windows Azure cloud service.

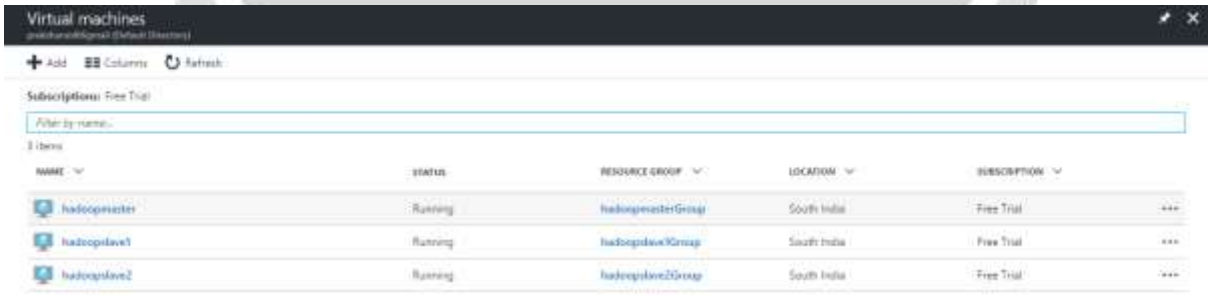


Figure 5. One Master and Two Slave Nodes

The above image depicts that for our experimental purpose, we have created one master and two slave nodes.

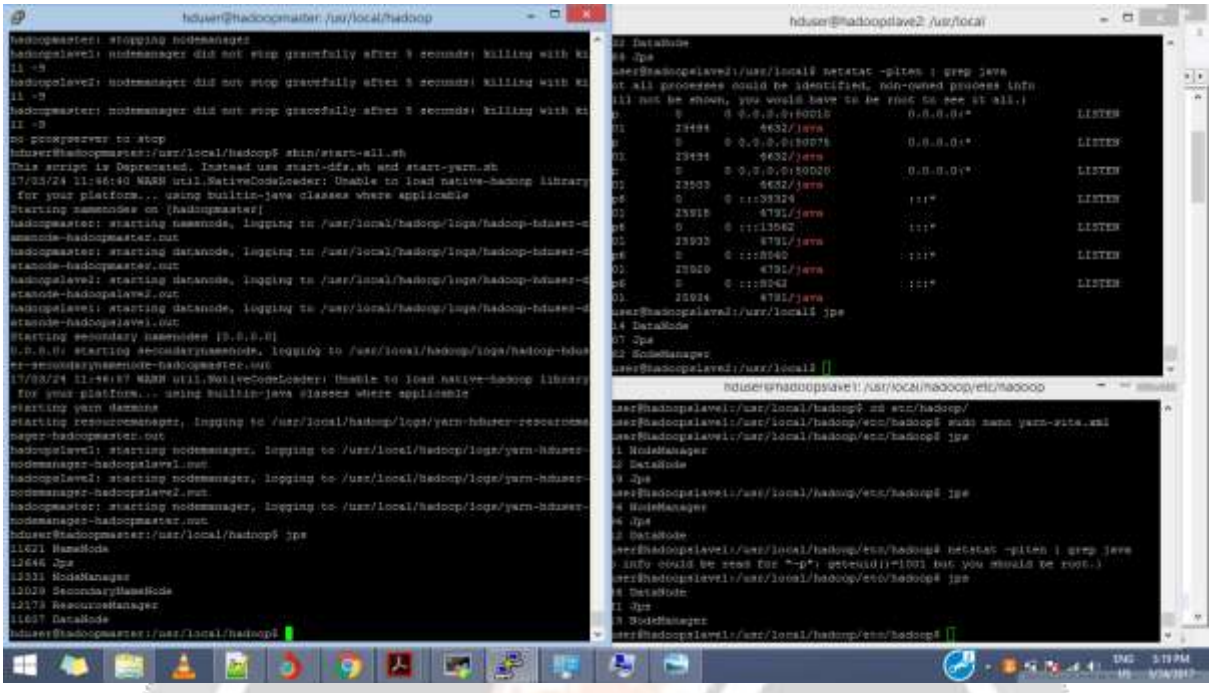

Figure 6. Daemons Running on the Master and Slave Nodes

The above image depicts the various daemon processes running on the master and slave nodes. The processes include ResourceManager, NameNode, Secondary NameNode, NodeManager, and DataNode.

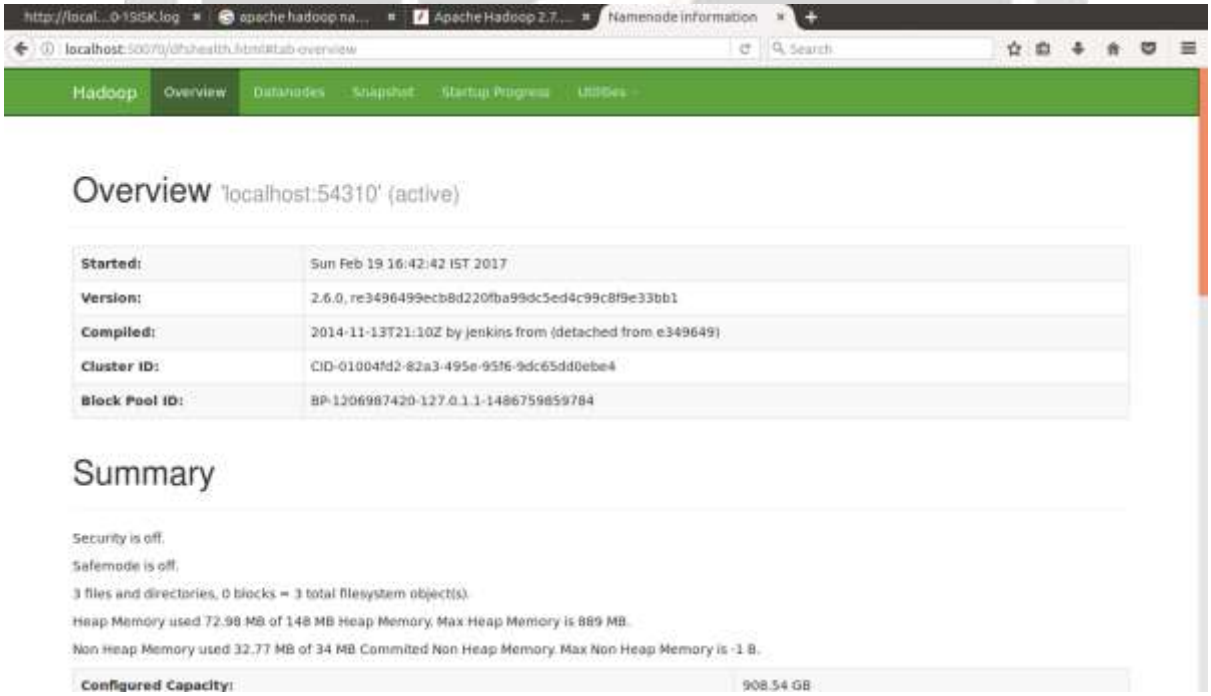


Figure 7. Hadoop Dashboard

The above image depicts how Hadoop Dashboard looks when we access the IP at which our host runs.

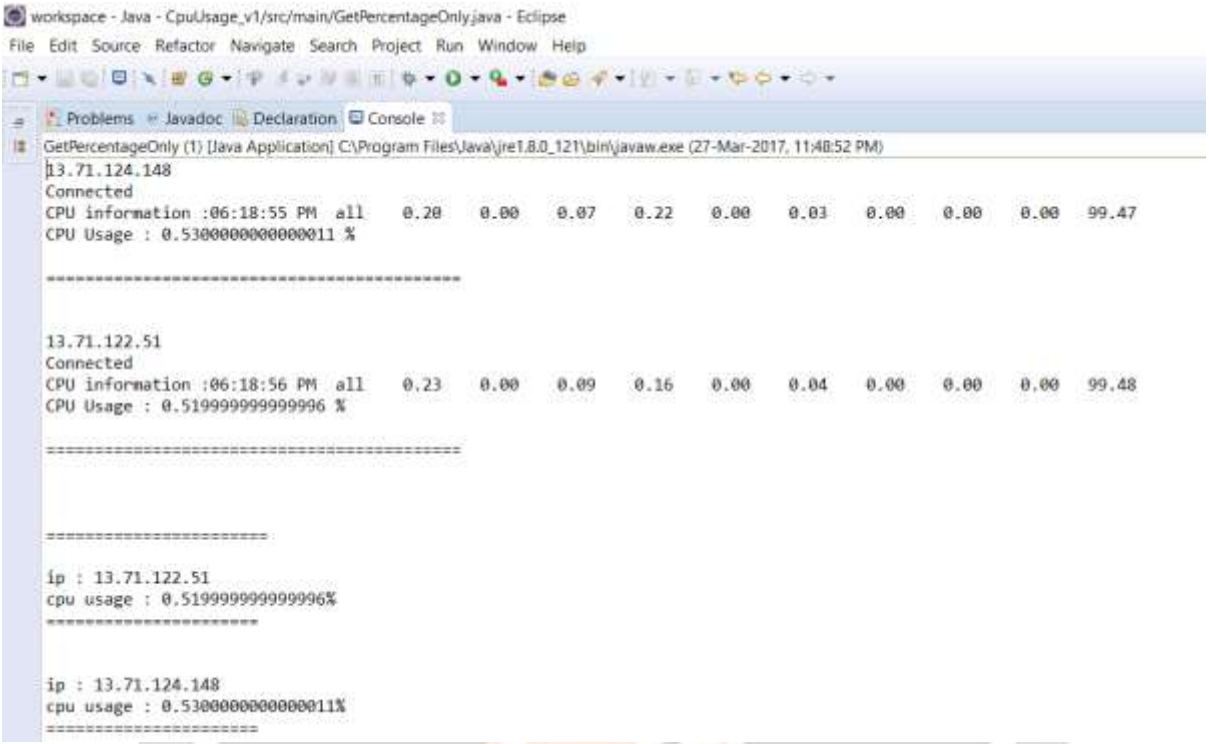


Figure 8. Fetching the CPU Usage Percentage and Creating a Sorted Queue

The above image depicts the output after Fetching the CPU Usage Percentage and Creating a Sorted Queue. This will be fed to the Resource Manager for making correct data placement according to our strategy.

## 5. PERFORMANCE EVALUATION

### A. Evaluation Parameters

1. Execution Time/Response Time of the Application
   Response Time = Job Completion Time – Job Submission Time (in seconds)
2. Size of The Input Data
   This refers to the size of the input file used for testing purpose.

### B. Performance Evaluation of the Proposed System

For the evaluation purpose, we ran two sample applications – WordCount MapReduce MapReduce job. The test was performed on two sizes of input file – 512 MB and 1024 MB respectively. WordCount job was performed for the same files. And then the results of the execution time taken by default Hadoop and proposed solution were compared.

The observed execution times for sample MapReduce job are as under:

| **Method Used** | **WordCount Execution Time for 512 MB Data Size** |
|---|---|
| Default Hadoop | 117 seconds |
| Proposed Solution | 115 seconds |

Table 2: WordCount job Execution Time for 512 MB Data Size

| **Method Used** | **WordCount Execution Time for 1024 MB Data Size** |
|---|---|
| Default Hadoop | 212 seconds |
| Proposed Solution | 207 seconds |

Table 3: WordCount job Execution Time for 1024 MB Data Size

As it is evident form the table, we observed a reasonable difference in the execution times between the default approach and the proposed approach in Hadoop.

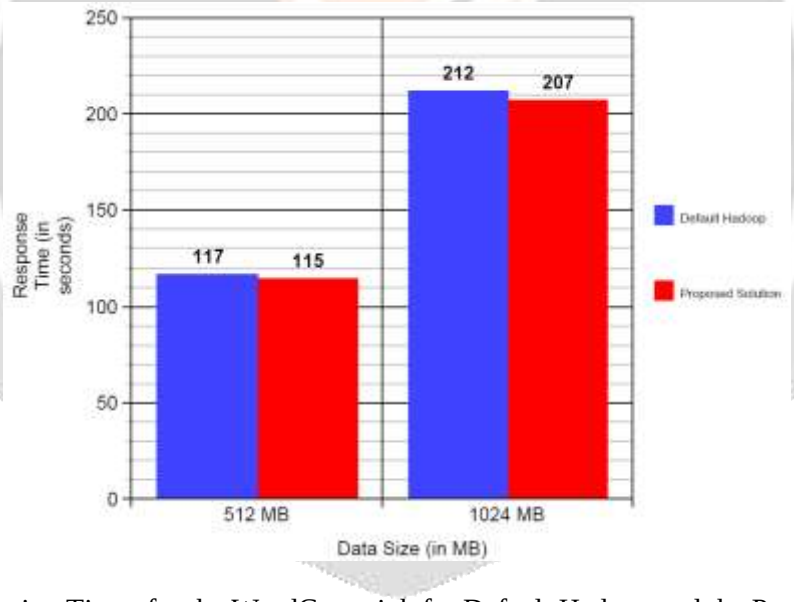


Figure 9: Execution Times for the WordCount job for Default Hadoop and the Proposed Solution

The proposed solution enhanced the response time of the job as clearly seen from the graph. It reduced the execution time by 1.71% for the 512 MB size input data and 2.359% for 1024 MB size input data. The evaluation was performed on a cluster of 4 nodes. Still it yielded notable enhancement in the response time of the jobs and improved the overall performance of the Hadoop cluster. This tends to imply that it could possibly bring considerable improvement of the Hadoop cluster by efficiently utilizing resources on large-scale Hadoop clusters. The proposed approach not just betters the performance but also resolves the straggler node issue as straggler nodes are not utilized due to their slow or poor performance.

## 6. CONCLUSION AND FUTURE WORK

This dissertation work presents an optimal data placement strategy in Hadoop environment. It describes a proper data placement design for Hadoop clusters. This dissertation explores the Hadoop data placement policy in detail and proposes a modified data placement approach that increases the performance of the overall system. The idea of placing data across the cluster according to the utilization of the nodes resources is presented, which will enhance the job execution/response times and improving the overall Hadoop cluster performance by efficient utilization of resources. The proposed solution fairly considers each node's resources and distributes the data based on their utilization. The new data placement strategy could bring reasonable improvements in large-scale Hadoop clusters. In future, the proposed work remains to be tested over a larger number of nodes to find out the true potential of the new system.

## 7. REFERENCES


[1] Suhas V. Ambade, Prof. Priya R. Deshpande, "Heterogeneity-based files placement in Hadoop cluster", 2015 IEEE, Pages: 876 - 880, DOI: 10.1109/CICN.2015.325

[2] C.-W. Lee et al., A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments, Big Data Research (2014), http://dx.doi.org/10.1016/j.bdr.2014.07.002

[3] Jia-xuan Wu, Chang-sheng Zhang, Bin Zhang, Peng Wang, A New Data-Grouping-Aware Dynamic Data Placement Method that Take into Account Jobs Execute Frequency for Hadoop, Microprocessors and Microsystems (2016), DOI: 10.1016/j.micpro.2016.07.011.

[4] B Ganesh Babu, Shabeera T P, Madhu Kumar S D, "Dynamic Colocation Algorithm for Hadoop", 2014 IEEE, Pages: 2643 – 2647, DOI: 10.1109/ICACCI.2014.6968384

[5] Vrushali Ubarhande, Alina-Madalina Popescu, Horacio Gonzalez-Velez, "Novel Data-Distribution Technique for Hadoop in Heterogeneous Cloud Environments", 2015 IEEE, Pages: 217 - 224 DOI 10.1109/CISIS.2015.37

[6] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters", 2010 IEEE, Pages: 1 – 9, DOI: 10.1109/IPDPSW.2010.5470880

[7] Yuanquan Fan, Weiguo Wu, Haijun Cao, Huo Zhu, Xu Zhao, Wei Wei, "A heterogeneity-aware data distribution and rebalance method in Hadoop cluster", 2012 IEEE, Pages: 176 – 181, DOI 10.1109/ChinaGrid.2012.22

[8] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, William Money, "Big Data: Issues and Challenges Moving Forward", 2012 IEEE, Pages: 995 – 1004, DOI 10.1109/HICSS.2013.645

[9] Jun Wang, Qiangju Xiao, Jiangling Yin, and Pengju Shang, "DRAW: A New Data-gRouping-AWare Data Placement Scheme for Data Intensive Applications with Interest Locality", 2013 IEEE, Pages: 2514 – 2520, DOI: 10.1109/TMAG.2013.2251613

[10] Hellerstein, Joe (9 November 2008). "Parallel Programming in the Age of Big Data". Gigaom Blog

[11] *Segaran, Toby; Hammerbacher, Jeff (2009).* Beautiful Data: The Stories Behind Elegant Data Solutions. *O'Reilly Media. p. 257.* ISBN 978-0-596-15711-1

[12] The Apache Software Foundation. Hadoop. http://hadoop.apache.org

[13] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107–113, 2008.

[14] New Horizons for a Data-Driven Economy – Springer, doi:10.1007/978-3-319-21569-3.

[15] The World's Technological Capacity to Store, Communicate, and Compute Information". MartinHilbert.net.

[16] JASON. 2008. "Data Analysis Challenges", The Mitre Corporation, McLean, VA, JSR-08-142

[17] "Welcome to Apache Hadoop!". hadoop.apache.org

[18] "What is the Hadoop Distributed File System (HDFS)?". ibm.com. IBM

[19] Malak, Michael (2014-09-19). "Data Locality: HPC vs. Hadoop vs. Spark". datascienceassn.org. Data Science Association

[20] Yu Tang; Abdulhay, E.; Aihua Fan; Sheng Su; Gebreselassie, K., "Multi-file queries performance improvement through data placement in Hadoop", Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on, vol., no., pp.986,991, 29-31 Dec. 2012

[21] https://dzone.com/articles/how-hadoop-mapreduce-works, Accessed: 23/11/2016, 4:40 PM

[22] http://www.slideshare.net/xqin74/performance-issues-on-hadoop-clusters, Accessed: 23/11/2016, 3:50 PM

[23] HDFS Architecture: http://hadoop.apache.org/docs/r1.2.1/images/hdfsarchitecture.gif

[24] Wiley "Hadoop for Dummies" By Dirk deRoos, Paul C. Zikopoulos, Roman B. Melnyk, Bruce Brown, Rafael Coss