International Journal of Latest Trends in Engineering and Technology
Vol.(9)Issue(4), pp.133-140
DOI: http://dx.doi.org/10.21172/1.94.22
e-ISSN:2278-621X

A NOVEL APPROACH FOR DATA STORAGE IN HETEROGENEOUS CLOUD ENVIRONMENTS

Dr.K.Purna Chand[1]

Abstract- Cloud computing is a type of parallel distributed computing system that has become a frequently used computer application. MapReduce is an effective programming model used in cloud computing and large-scale data-parallel applications. Hadoop is an open-source implementation of the MapReduce model, and is usually used for data-intensive applications such as data mining and web indexing. The current Hadoop implementation assumes that every node in a cluster has the same computing capacity and that the tasks are data-local, which may increase extra overhead and reduce MapReduce performance. This paper proposes a data placement algorithm to resolve the unbalanced node workload problem. The proposed method can dynamically adapt and balance data stored in each node based on the computing capacity of each node in a heterogeneous Hadoop cluster. The proposed method can reduce data transfer time to achieve improved Hadoop performance. The experimental results show that the dynamic data placement policy can decrease the time of execution and improve Hadoop performance in a heterogeneous cluster.
Keywords – Hadoop; MapReduce; Heterogeneity; Data Distribution.

1. INTRODUCTION

In this data era the faster development of the web services are the most commonly used applications. We are using Search engines like Google and social networking sites like Facebook and many E-commerce websites providing indispensable service to the users. The rapidly increased users for web services may cause performance issue on large data are a challenging thing. While working with huge data, we need to perform in a parallel manner. This will result in better performance. The Google file system GFS proposed by Google.

Big data uses MapReduce model to construct a data center that can process 20 petabytes of data per day. Hadoop used to process data of hundreds of terabytes as well as to store and manage that much of large data. MapReduce model has several advantages which is different from traditional approaches of parallel computing. The main aim for using parallel computing is to maintain scalability. The performance will not be affected negatively even though the input incremented then we can say it as a scalable environment. Data locality is a factor which may affect performance of the Hadoop. We are here to deal with the heterogeneous environment; the required data is often nonlocal to perform tasks, which affect the Hadoop performance. Traditional homogeneous environments having similar configuration and same ability to perform tasks.

When data writing into Hadoop distributed file system HDFS, it will be divided into DataBlocks of equal size. These data blocks will be distributed into all the nodes in the cluster equally. The default size of DataBlock in Hadoop is 64MB. Where all nodes having equal capacity and ability to perform the task. In our proposed, Data Load Distribution in heterogeneous environment, the data nodes having heterogeneous configuration. The hardware configuration for data nodes in the heterogeneous cluster would not be the same. In this approach the node with high configuration will execute the task faster with the local data. The non-local data blocks will be performed after the execution of local data blocks. Which will results in approximately same. The data load distribution will reduce the transmission time.

HDFS maintains data in the form of DataBlocks, and these DataBlocks are distributed equally to all the node in the cluster. If any node fails to connect the cluster it may result in data Reliability problem. To overcome this inconsistency Hadoop maintains replications. The defauls replication factor is 3. If any node fails to connect to the cluster the relicated node will supply the data. This data load distribution will give you the better performance than the traditional one. Where workload will be different for the each node in heterogeneous environment. As per our idea Task allocated based on the computing capacity and configuration which will increase the efficiency Hadoop. Resource usage may be decreased after written into the HDFS due to beginning activities of Hadoop daemons. We will reallocate data load dynamically by this proposal. We compared the performance of wordcount, grep and maxtemp examples using proposed work. We improved execution time nearly 17.5% for word count job, 21% for the grep and 25.5% with the maxTemp.

[1] Associate Professor, Department of Computer Science and Engineering, B V Raju Institute of Technology, Narsapur, Telangana State, India

2. RELATED WORK

2.1. Hadoop

Hadoop implementing the MapReduce Model to run tasks, it is an open source which is keep on updating and keep releasing newer releases with fixing bugs and error free. Hadoop presently having two main versions: hadoop-1.x in Fig.1. and Hadoop-2.x Fig.2.. Hadoop having two major modules in it: one is HDFS (Hadoop Distributed File System) and other is Mapreduce. HDFS is responsible for storing and managing the data in hadoop environment. Coming to the MapReduce, it is responsible for processing the data. Hadoop-2.x version YARN module was included. YARN also known as MapReduce Version.2.0. Maintaining the Integrity of the Specifications.

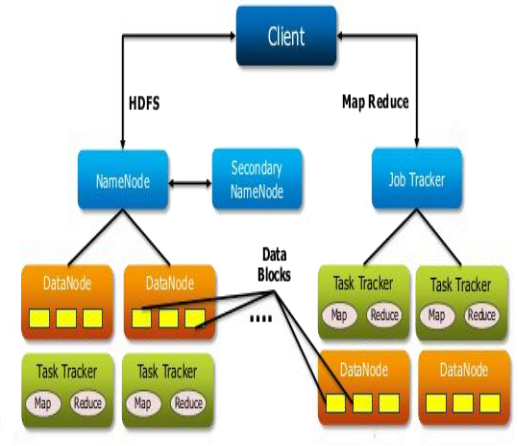


Figure 1. Architecture of Hadoop-1.x

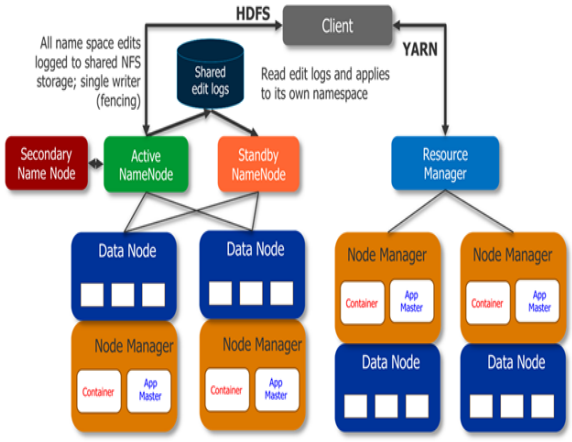


Figure2. Architecture of Hadoop-2.x

2.2 Hdfs

Yahoo implemented HDFS depending on the GoogleFile System GFS. Hadoop distributed File System HDFS is mainly designed to follow the google file system to store the data and manage the data. HDFS consists of single NameNode(Master) and Multiple DataNodes(Slaves). HDFS having 3 major parts: NameNode, SecondaryNameNode and DataNode.NameNode, also called asMasterNode which maintains the overall data which is having all the information about the cluster and corresponding slaves data. NameNode allocates the jobs/tasks to DataNodes. SecondaryNameNode is an alternative for the NameNode, Whenever NameNode fails to get the information about DataNodes the meta data will be supplied from the secondaryNameNode. We can call it as a copy of NameNode. DataNode is a slave node which stores the data written by the namenode in the form of data blocks. The default data block size is 64MB. Which performs all the tasks allotted by NameNode using the task slots of the datanode and retuns the result to the NameNode, then the individual results from the Datanode can be result in single aggregate output.

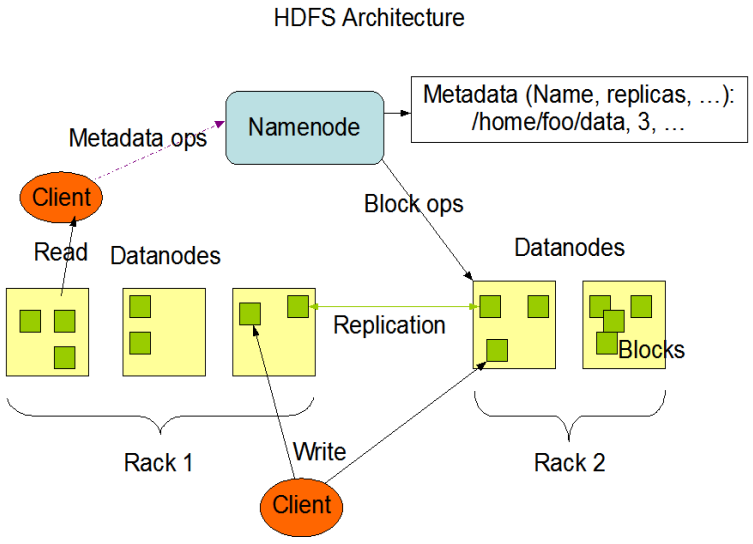


Figure.3. Architecture of HDFS

2.3 Mapreduce

MapReduce is a data processing model for Hadoop, HDFS works as storage and manipulation and MapReduce is to process the data by Sorting and Shuffling. MapReduce is used to compute or process more than one task parallel. In the year 2004 Google proposed MapReduce. Any application which is developed in MapReduce is called as MapReduce job. Mapreduce job contains two tasks one is map task another one is reducer task. Map task runs the map() method and Reducer tasks runs the reduce() method. The data processed by the MapReduce is assigned by the Master and produces the many intermediate <key,value> pairs. Based on the <key value> pairs map and reducer methods shuffle and sort the data. MapReduce mainly follows Divide and conquer policy. It distributes the large amount of data into the all DataNodes for parallel processing leads to reduction of execution time and improves performance. MapReduce distributes equal no of equal sized datablocks to the slave nodes or DataNodes. Job Tracker allots slots and jobs, TaskTrackers completes the task and submits result to the master and then Master Makes it as a single and final output.

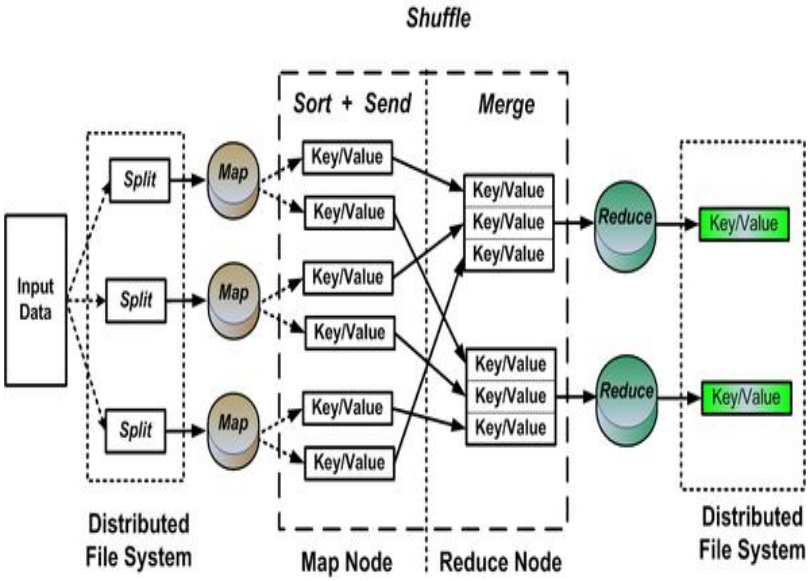


Figure.4.Mapreduce model overview

2.4 Yarn

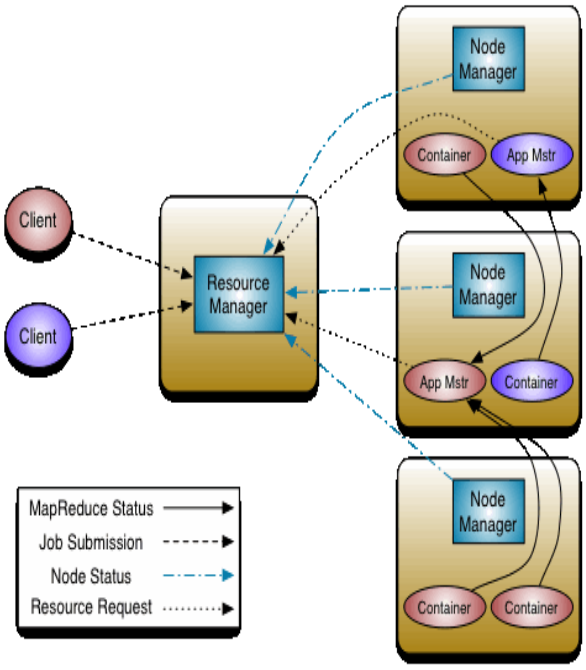


Figure 5. Architecture of YARN

Yet Another Resource Negotiator. YARN is nothing but the MapReduce Version-2.0. To overcome the limitations of MapReduce YARN introduced as update version MapReduce. The main idea of the YARN is to split up the job scheduling and monitoring into daemon separately. It contains ResourceManager and NodeManager to handle the master and slace mode. Yet another resource negotiator maintains resource reservation.

2.5 Impact

Hadoop cluster contains no of nodes in cluster which are homogeneous in nature. The hardware configuration and specification of the nodes are same and each node is assigned to the same load of data. In real world clusters often worked in heterogeneous environment. In those type of clusters there should exist nodes with different configurations. This will have different amount of capacity to process the allotted tasks. The main reason behind the differently configured nodes is, it will improve the overall performance where executing local data blocks faster than the slower nodes. Let us consider an example that having three nodes in a cluster namely Node A, Node B and Node C. These nodes are dissimilar by their hardware configuration and processing capability. Node A is faster followed by Node B and C, Node C having least performance. Let us assume that Node A is two times faster than Node B and three times faster than Node C. Coming to processing job requiring data blocks are equally distributed to each node. When the job begins, Node A finished processing first. At this point Node B and Node C finishes only one data block. Here, The unprocessed data blocks transferred to the Node A by NameNode . Unless the Node A should waits until the remaining data blocks got processed by the Node B and Node C completely. This will affect the performance of the Hadoop. To avoid these conflicts we proposed DLD.

3. DATA LOAD DISTRIBUTION

In heterogeneous Hadoop cluster, the computing capacity of each node will not be the same. Different types of jobs allotted as per Ratio of Computing capacity of a node. Ratio of computing capacity also not same for every data node of cluster. DLD (Data Load Distribution) is proposed to adjusting the distribution of data blocks. While writing data to HDFS (Hadoop distributed file system) as per PerformanceTable discussed below.

3.1. Performance Table

A PerformanceTable was created when the hadoop starts. This performance table used to write the data into HDFS. It decides that whether the data blocks are needs to be reallocated or not. The performance table job is to stores type of jobs and Ratio of computing capacity (RCC) of each node. The computing capacity of each node based on the average time taken for execution of a task. The master node calculates the ratio of computing capacity based on the time taken for the execution of tasks.

Table 1, showing an example, a cluster containing 4 nodes and computing capacities of each node. And are not same. Node A is Fastest followed by node B, Node c and node D may approximately be same. If we compare the time of execution while performing the wordCount, grep and maxTemp. The performance ratio returns as 4:2:1:1 for word count, 3.5:2.5:1:1 for grep and 3:2:1:1 for maxTemp.

3.2. Data Load Distribution Strategy

After writing the data into HDFS (Hadoop distributed file system), master node initially checks with the PerformanceTable. The data in performance table contains the information about the type of job has to be performed. If performance table has the information about the job, newly allocated data will be allotted to the each node with respect to the computing capacity in the performance table. If performance table has nothing about the job. Them the data will be distributed equally to all the nodes in the cluster, Name node will create a new record of this type of job in the performance table.

As shown in Table 1, there is data to be written into the hadoop distributed file system. He data should get partitioned into data blocks of equal size. Let us assume here created 8 data blocks when we want to run the wordcount job then the data will be distributed according to the PerformanceTable. Node A assigned with 4 blocks, Node b assigned with 2 data blocks, Node C and D with as data block each. The allocation of data will done as per data in performance table.

When job starts the execution every data node will receive the first load of tasks. In this heterogeneous environment each node contains different computing capacity. When the job allocated to data node, the execution time will be return to the master node. Like this the name node allocates different jobs to the data node and calculates the average execution time. The calculation of execution time will be done as per the task slots. Each node will have their no. of task slots. If any having highest no of task slots then it will give the best performance than other nodes. Because the task slots performs simultaneously. The task execution time calculated based on the ratio of average computing capacity of the nodes and no of task slots it has.

Consider an example there are four nodes in a cluster : Node A, Node B, Node C and Node D.

Node A has 4 task slots,
Node B has 2 task slots,
Node C has 1 task slot,
Node D has 1 task slot.

The time taken for executing 4 tasks with the Node A taken as 44,38,41 and 37 seconds. For Node B it has taken 39 and 37 seconds. For nodes C and D it has taken 39 and 42 seconds respectively. If we compare the average time taken to execute a single task we get 40 sec for node A, 38 sec for node B, 39 and 42 for node C and D, here the result saying Node B having the higher performance.

The resultant calculated as follows

Table-I –Cluster Nodes

Job	**Node A**	**Node B**	**Node C**	**Node D**
Word Count	4	2	1	1
Grep	3.5	1.5	1	1
Max Temp	3	2	1	1

Initially we said Node A has highest computing capacity and it will be calculated based on the no of task slots in the data node. By this approach we get actual values which are real. We got 10 seconds for Node A, 19 seconds for node B, 39 seconds for node C and 42 seconds for node D.

SN(X) ← no of map task slots in node X

T_{avg} ← average time taken for execution

T_{ET} ←Execution time of a Task

As per our discussed example $T_{ET}(A) = 10$,

$T_{ET}(B) = 19$,

$T_{ET}(C) = 39$,

$T_{ET}(D) = 42$.

These values will be recorded in the performance Table and will be considered for new type of jobs. If no the Name node will make a record.

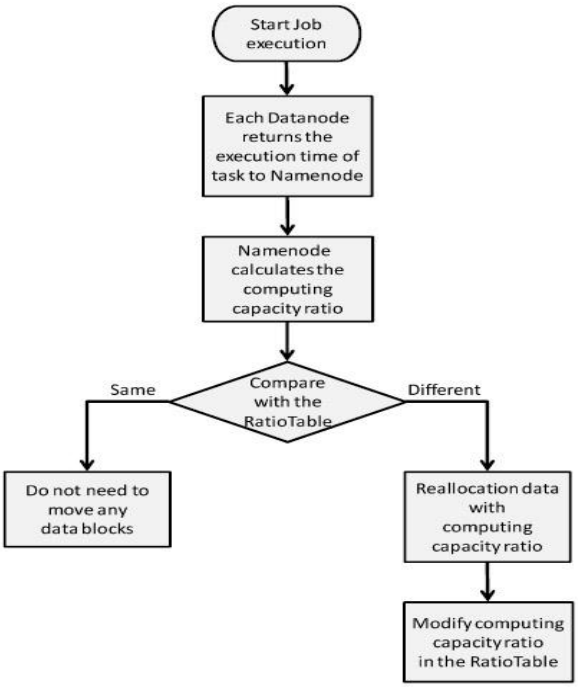


Figure. 6. Execution Flow of DLD

Algorithm : Data Load Distribution

When data load into HadoopDistributedFileSystem:
JT ← job type of the data to be performed;
DS ← obtain from data information;
BS ← block size will be decided by user;
NumOfBlocks = $\left\lceil \frac{DS}{BS} \right\rceil$;
set Same = 0;
NN ← NodeNumber obtain from NameNode;
CN [NN] ← set 0 for all entries;
ET ← task execution time;
TET [NN] ← set 0 for all entries;
for each record in the PerformanceTable **do**
if compare JT with record are the same **then**
Same=1;
RCC ← Ratio of Computing Capacity obtain from record;
for each DataNode in the cluster **do**
NC ← obtain from Ratio of Computing Capacity;
BN = NumOfBlocks∗$\left\lceil \frac{NC}{\sum NC} \right\rceil$;
Allocate BN data blocks to the DN;
if Same = 0 **then**
RCC ← set 1 for each node;
Add JT with RCC to PerformanceTable;
for each DN in the cluster **do**
NC = 1;
BN = NumOfBlocks∗$\left\lceil \frac{NC}{\sum NC} \right\rceil$;
Allocate BN data blocks to the DataNode.
When a job start:
while receive the task execution time from DataNode[i] **do**
SN ←obtain from DataNode[i];
TET [i] =TET [i] +ET;
CN [i] =CN [i] +1;
if CN [i] = SN **then**
Tavg = $\left\lceil \frac{TET[i]}{SN} \right\rceil$;
Tt = $\left\lceil \frac{Tavg}{SN} \right\rceil$;
CN [i] = 0;
TET [i] = 0;
if obtain Tt for each node **then**
PerformanceRatio ← PerformanceRatio and Tt are inversely proportional;
for record in the PerformanceTable **do**
if compare PerformanceRatio with record are different **then**
Reallocation data blocks according to PerformanceRatio;
Modify the record according to PerformanceRatio.

4. EXPERIMENTAL RESULTS

We have used word count, grep and maxTemp jobs to evaluate the performance of data load distribution in hadoop heterogeneous cluster environment.

Table-2 –Each Node Specifications

Node	CPU cores	RAM	DISK SPACE
A(Master)	2	2GB	100GB
B (Slave)	4	4GB	100GB
C (Slave)	2	2GB	100GB
D (Slave)	1	1GB	100GB
E (Slave)	1	1GB	100GB

These job are used to return in HDFS. Generally word count is an application used to count the no of words in an input file. Grep is used to query for regular expressions and maxTemp used to calculate the maximum temperature of an area in particular period of time using recorded temperature data over years. Experimental data runs in 10 round of size 1GB to calculate average time of execution.

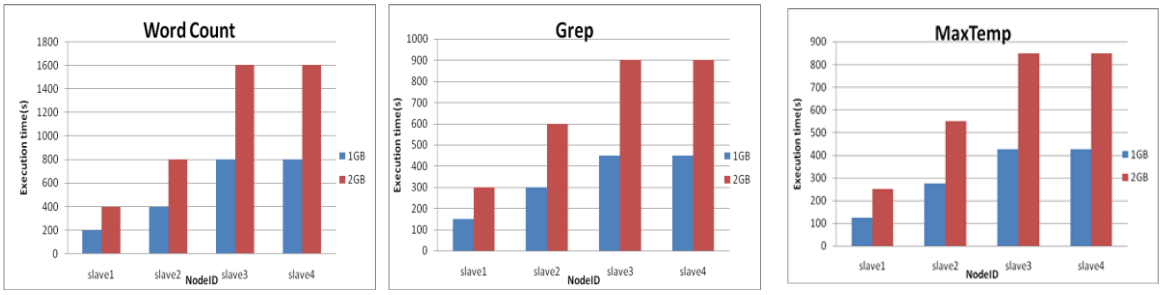


Figure.7. Execution time of different commands on each node

Firstly, the execution time of each job, word count, Grep and maxTemp calculated with different sizes of data. Figure 7., showing the performance with 1GB and 2GB of word count,Grep and data at each heterogenous node of hadoop cluster individually. The execution time of each data node is proportional to the size of the data.

The information deployed in figure 7 is considered to calculate the computation of ratio for each individual node for 3 types of jobs, word count, Grep and maxTemp respectively. According to the experimental results, ratio of computing capacity maintained. For word count job, based on the time of execution slave 1 is four times faster than slave 3&4, and two times faster than slave 2.Whenever the job performed in the hadoop HDFS, It took extra time to start daemons and it not only performs the map task also performs the reducer task. But, the intension is to evaluate the best performance at mapping job by optimizing.

We need to calculate the average time of each job where as the execution times are same for local and non-local task. To calculate the accurate result, we did it for 3 jobs. It also depends on the data block size. If we change size of the datablock. It is set to be larger than time taken for executing a local task compared to the non-local one. Therefore, when we are calculating the ratio of computing capacity, we need to consider data block size. So the experimental data default data block size. We have even used 128 MB block size to differentiate the execution time and ratio of computing capacity. After calculating the ratio of computing capacity, as per DLD, we have got this table as a result.

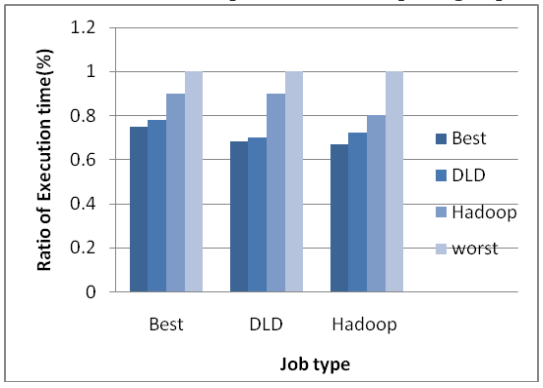
Table 3-The job computing capacity ratio of each node

Node	Job Type		
	Word Count	GREP	Max Temp
Slave1	4	3.5	3
Slave2	2	1.5	2
Slave3	1	1	1
Slave4	1	1	1

Table 4- Task computing capacity ratio

Node	Job Type		
	Word Count	GREP	Max Temp
Slave1	4	3.5	3
Slave2	2	1.5	1.5
Slave3	1	1	1
Slave4	1	1	1

The purpose of DLD algorithm will be compared with hadoop default strategy, best case of distribution, and worst case distribution. The best case resulted in local data, which does not require transfer of data. If, the data process as per our logic it will be distributed as per ratio of computing capacity.



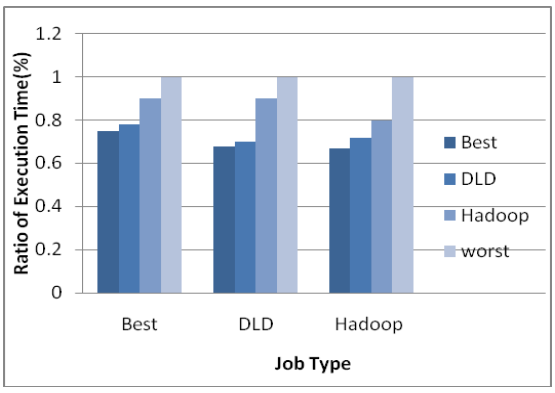


Figure.10 Average Execution Time

When the word count job is executing as per our logic, the data distributed as per DLD. The data creates 16 data blocks. Node A assigned with 8 blocks, node B assigned with 2, node C&D assigned 2 each. The best performance evaluated at node A and worst case of node C&D. It follows table 4.

Hence the average execution time of DLD is close enough to optimal case. If you performing for the first time, it does not follow the ratio of computing capacity, the little delay may occured. The proposed algorithm will balance the location of data block. After this data written to HDFS and will be allocates as per RCC. Best case and DLD results almost identical. Compared to the hadoop default strategy with data load distribution; DLD reduces 17.5% of execution time and compared worst case 25.7% improved approximately with the word count. For the Grep job, the execution time will reduce reduce by 27.5% alnd worst case are improved by 30%, for maxTemp job 24.5% and 32.5% approximately.

5. CONCLUSION

This paper proposes a data placement policy (DDP) for map tasks of data locality to allocate data blocks. The Hadoop default data placement strategy is assumed to be applied in a homogeneous environment. In a homogeneous cluster, the Hadoopstrategy worstcase.can makes full use of the resources of each node. However, in a heterogeneous environment, a produces load imbalance creates the necessity to spend additional overhead. The proposed DDP algo-rithm is based on the different computing capacities of nodes to allocate data blocks, thereby improving data locality and reducing the additional overhead to enhance Hadoop performance. Finally in the experiment, for two types of applications, WordCount and Grep, the execution time of the DDP compared with the Hadoop default policy was improved. Regarding WordCount, DLD reduces 17.5% of execution time and compared worst case 25.7% improved approximately with the word count. For the Grep job, the execution time will reduce reduce by 27.5% alnd worst case are improved by 30%, for maxTemp job 24.5% and 32.5% approximately.

6. REFERENCES

[1] Chia-Wei Lee, kuang-Yu Hseieh,sun-Yuan Hseieh, “A dynamic Data displacement strategy for hadoop in heterogeneous Environments”national cheng kung university,university road,Taiwan, Bigdata research 1 (2014) 14-22..

[2] Amazon elastic Mapreduce, http:// aws.amazon.com/elastic mapreduce.

[3] Apache http://httpd.apache.org/.

[4] Hadoop, http://hadoop.apache.org/.

[5] Hadoop distributed file System, http://hadoop.apache.org/docs/stable/hdfs

[6] Hadoop MapReduce, http://hadoop.apache.org/docs/mapred_tutorial.html.

[7] A comprehensive view of hadoop map reduceschedulingalgorithms-seyed reza pakize . department of computer science, Islamic azad university,international jounal of computer networksand communications security- vol 2, no. 9 sept 2014, 308-317.

[8] Improving mapreduce performance through data placement in heterogeneous hadoop clusters – Jiong Xie, shu yin Zhiyan ding,department of computer science and networks, auburn university, Auburn,Al 36849-5347.

[9] Andy konwinski, improving mapreduce performance in heterogeneous environments technical report number UCB/EECS-2009-183.

[10] Krish, K.R.; Anwar, A: Butt,”hatS”: A Heterogeneity-Aware Tiered Storagefor Hadoop,”cluster,cloud and grid computing(CCGrid),”2014 14th IEEE/ACM international symposiumon, vol.,no,pp.502,511,26-29may2014.

[11] [11] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, and X. Song, "Accelerating Spatial Data Processing with MapReduce", in Proc. ICPADS, 2010, pp.229-236 .

[12] [12] Jun Zhang, N. Mamoulis, D. Papadias, and Yufei Tao,“All-nearest-neighbors queries in spatial databases,” June 2004, pp.297–306.

[13] [13] Haojun Liao, Jizhong Han, Jinyun Fang, "Multi-dimensional Index on Hadoop Distributed File System," Fifth International Conference on Networking, Architecture, and Storage, 2010, pp.240-249

[14] [14] A. Guttman, “R-trees: a dynamic index structure for spatial searching,” in Proceedings of the ACM SIGMOD. Boston, Massachusetts: ACM, 1984, pp. 47–57.

[15] [15] Halim, F.; Yap, R.H.C.; Yongzheng Wu; , "A MapReduce-Based Maximum-Flow Algorithm for Large Small-World Network Graphs," Distributed Computing Systems (ICDCS), 2011 31st International Conference on , 20-24 June 2011, pp.192-202

[16] [16] Pavlo, a., Paulson, e., rasin, a., abadi, d.J., deWitt, d.J., Madden, s.r., and stonebraker, M.A “comparison of approaches to large-scale data analysis”. In Proceedings of the 35th SIGMOD, International Conference on Mgmt. of Data. ACM Press, New York, 2009, pp.165–178.

[17] [17] Abouzeid, a., bajda-Pawlikowski, K., abadi, d.J., silberschatz, a., and rasin, a. Hadoopdb: an architectural hybrid of Map-reduce and DBMS technologies for analytical workloads. In Proceedings of the Conference on Very Large Databases, 2009.