ISSN:1738-6764, eISSN:2093-7504

Performance Optimization of Big Data Center Processing System - Big Data Analysis Algorithm Based on Location Awareness

Wen-Xuan Zhao [1], and Byung-Won Min [2,*]

[1] Teaching affairs office of HeXi university; Gansu, China; Associate Researcher; Division of Information and Communication Convergence Engineering, Mokwon University; Ph.D student; bzhpzh@163.com
[2] Division of Information and Communication Convergence Engineering, Mokwon University; Professor; minfam@mokwon.ac.kr
* Correspondence

https://doi.org/10.5392/IJoC.2021.17.3.074

Manuscript Received 10 September 2021; Received 28 September 2021; Accepted 28 September 2021

Abstract: *A location-aware algorithm is proposed in this study to optimize the system performance of distributed systems for processing big data with low data reliability and application performance. Compared with previous algorithms, the location-aware data block placement algorithm uses data block placement and node data recovery strategies to improve data application performance and reliability. Simulation and actual cluster tests showed that the location-aware placement algorithm proposed in this study could greatly improve data reliability and shorten the application processing time of I/O interfaces in real-time.*

Keywords: Bigdata Center Processing System; Location-awareness; HDFS distributed storage; Data Mining Alogorithm; Performance Optimization

1. Introduction

With the massive growth of data, traditional data processing systems no longer meet the data processing requirements. As a result, people have shifted their focus on data processing to cloud storage or cloud architecture, such as HDFS distributed storage management system or MapReduce framework. The former stars from the storage method or the framework of storage. However, HDFS mainly targets homogeneous physical clusters but does not perceive the problem of multi-virtual machine coexistence. Therefore, the default HDFS data placement algorithm is often not suitable for cloud environments, specifically in terms of low reliability of data and degraded performance of upper-level data. Therefore, how to enhance the allocation of data blocks in the cloud environment more scientifically and reasonably has become the focus of current thinking. In response to the above problems, Cong Li adopts the storage architecture of hybrid cloud and fog computing structure and introduces an ant colony algorithm to optimize the data allocation problem [1]; Wen Shilin proposes to store files in different server devices according to the file hotness, a multi-copy policy is used to achieve high data reliability, with three data copies by default. HDFS exposes data block copies to upper-layer applications through APIs, thus facilitating the scheduling of data processing tasks close to data block locations and improving data storage and analysis. HDFS data allocation occurs when data is first written into HDFS (data initialization) and after a failure of one of the cluster nodes (data recovery) [2]. As for HDFS, which is the support base of a big data center processing system, its data block placement strategy is mainly oriented to homogeneous physical clusters, which will lead to the loss of data block copies when the clusters fail, thus damaging the reliability of data. As the data is distributed among different nodes of the virtual machine cluster, the processing capacity of each node is different, which will result in unbalanced data processing load of each node, thus reducing the data processing application performance of HDFS. To solve the above two problems, the most direct approach is to avoid the coexistence of virtual machines. However, this method will reduce the cloud resource utilization and harm the economic efficiency, so Jin GD, Bian HQ et al. proposed to cope with the above two problems from the task scheduling perspective and data block placement perspective, but they only considered the equal number of virtual machines in different physical machines [3], so the above method is not adapted to a wide variety of virtual network topologies in cloud environments.

Li Hui, et al. proposed a new method of represented by a tree hierarchical network topology [4]. Hua Qingsheng, et al. proposed a new big data analysis system and achieved good results [5]. As in Figure 1, where the cluster can consist of different server nodes. Typically, the data allocation of HDFS usually places the first two copies of the data block in different nodes of the same rack and the third copy in other racks. If the replica factor is higher than the default 3, the extra replicas are randomly placed in the racks to improve data reliability and read speed. Moreover, in the data recovery phase, the lost data block replicas are found by HDFS and are reallocated to the active nodes so that all replicas can meet similar data reliability when data is initially allocated.

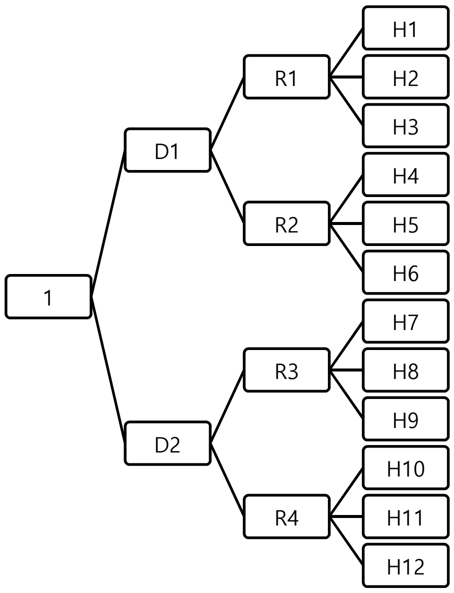

Figure 1. HDFS tree hierarchy network topology (D denotes the data center, R denotes the rack switch and H denotes the data storage node)

In the case of HDFS, although it is feasible in the physical environment, the default placement policy set by itself brings two problems: first, the data reliability decreases. As in the cloud environment, it is generally coexisting for multiple virtual machines, and if the physical host fails, then multiple virtual nodes will have problems, i.e., multiple copies of data blocks will be lost; second, the application processing performance is reduced. Due to the performance differences between different nodes, the execution will reduce the application processing performance. To solve the above problems, a straightforward solution is to avoid the coexistence of virtual machines. However, this imposes significant limitations on the placement of virtual machines in the cloud, such as the number of physical machines available in the cloud data center must not be less than the number of virtual nodes required, and many nodes are not suitable for deploying HDFS nodes. Therefore, it is impractical to avoid coexistence between VMs, and a reasonable placement algorithm must be used.

2. Location-aware data block placement algorithm

2.1. Principles for building location-aware data block placement strategies

The fundamental role of the location-aware data block placement strategy proposed in this paper is to solve the problem of data reliability and application processing performance degradation, so the following principles are followed in solving the problem: first, avoid placing two identical copies in one virtual node; second, do not place the same two copies in one physical host; third, try to place three copies under different racks. Furthermore, to solve the problem of application performance degradation of the system, it is proposed to improve the application performance by controlling the number of each file on the node, thus balancing the expected processing time of the data as much as possible. In other words, based on the information of the data blocks that have arrived in the file, the copies of the currently arriving data block requests are placed at the node with the least expected completion time at present as much as possible. The expected processing time per node is [6].

$$\text{ProcessTime} = \text{BlockSize} \times \frac{\text{blockNumber}}{\text{ProcessingCapacity}} \tag{1}$$

Where BlockSize indicates the size of the selected data block, BlockNumber indicates the number of data blocks assigned to a node by a file, and PC indicates the amount of data that the node can process per second during the Map task phase.

2.2. Data initialization write allocation strategy

2.2.1 Objective function construction

Let the HDFS cluster consists of many virtual nodes (vm1, vm2, vm3, ... , vmn), define a distance matrix D to reflect the network distance of different nodes, and the size of the matrix is m*m (m is the total number of virtual nodes). Based on this topology, the total number of switches $\boldsymbol{n_{rack}}$ and the total number of physical machines $\boldsymbol{n_{pm}}$ are obtained. Meanwhile, to characterize the performance processing capability of different nodes in the cloud environment, a vector P of length m is defined, where $\boldsymbol{P_i}$ is the processing capability of node $\boldsymbol{VM_i}$.

When a data block write request reaches the NameNode, the node location of length r is returned to the client. Let the total number of data blocks in the file be n, then the number of locations for writing data blocks is r*n. Set the matrix A as the final data block distribution, which is m*n dimensional. Where $\boldsymbol{A_{ij}}$ indicates whether the data block $\boldsymbol{j}$ is placed at the location of the node $\boldsymbol{VM_i}$, the objective function is set to make the standard deviation of the expected processing time of the local data as small as possible in the Map phase [7]. Let the expression for the expected time $\boldsymbol{T_i}$ be:

$$T_i = \frac{\sum_{j=0}^{n} A_{ij} * \delta}{P_i}, i \in [0, m) \tag{2}$$

Where S denotes the size of the data block.

Based on the node times calculated from each of the above equations, the mean and standard deviation Ti can be calculated as in equations (3) and (4) [8].

$$\mu = \sum_{i=0}^{m} T_i / m \tag{3}$$

$$\sigma_{time} = \sqrt{\frac{\sum_{i=9}^{m} (T_i - \mu)^2}{m}} \tag{4}$$

The constraints are set based on the above objective function at the virtual machine level, the physical machine level, and the rack switch level, respectively. The constraints are as follows [9]:

$$\forall_i \in [0, m), j \in [0, n) : A_{ij} \in [0,1) \tag{5}$$

$$\forall_j \in [0, n) : \sum_{i=0}^{m} A_{ij} = r \tag{6}$$

$$\begin{aligned} &\forall_j \in [0, n) : D_{ab} \geq 4 \quad and \quad D_{ac} \geq 4 \quad and \quad D_{bc} \geq 4 \\ &ifn_{pm} \geq 3 \quad and \quad A_{aj} = A_{bj} = A_{cj} = 1 \end{aligned} \tag{7}$$

$$\begin{aligned} &\forall_j \in [0, n) : D_{ab} \geq 6 \quad or \quad D_{ac} \geq 6 \quad or \quad D_{bc} \geq 6 \\ &ifn_{rack} \geq 2 \quad and \quad A_{aj} = A_{bj} = A_{cj} = 1 \end{aligned} \tag{8}$$

Equation 5 and Equation 6 are the constraints at the virtual machine level. Equation 5 indicates that only one copy of each data block is distributed in the same virtual machine, and Equation 6 indicates that there are a total of r different copies of each data block in m virtual nodes, and Equation 5 and Equation 6 ensure data reliability. Equation 7 is the constraint to be followed for data reliability at the physical machine level, while Equation 8 is the constraint that should be followed for data reliability at the rack switch level.

2.2.2 Solving the objective function

When the data block is matched to the optimal copy placement location, combining any three nodes theoretically satisfies the data reliability constraints, thus minimizing the objective function. However, the study concluded that the total time complexity of finding the best combination of nodes is O(m4) when solved according to this theoretical approach, which indicates that the data block request writing time is long. When there are 128 data nodes, the time is taken to respond to a single data block request can exceed 1 second[10], which dramatically increases the system response time. The greedy algorithm is thus proposed to solve the above problem as shown in Figure 2.

```
    input:  m:Number of nodes
            D:Matrix of network distances between nodes
            P:Vectors of node processing power
            s,r,n:Data block size, data block copy factor, and total number of data blocks in the current
            file
    Output
    :       A:Short array of allocation positions for copies of all data blocks of a file
1   Initialize $A_{ij}$←0,$time_i$←0;
2   For block=I to n do
3   update $time_i$; //using the equation 4.2
4   Initialize setA←{l, 2,..., m}, setB←NULL. setC←NULL;
5   nodeA←the node in setA where $tinie_{nodf}$ is the minimum;
6   for distance AB in [6, 4, 2] and set B is NULL do
7           for i=I to m do
8               if $D_{node}$A.j==distance AB then setB.append(i);
9           end
10  end
11  nodeB←the node in setB where $time_{node}$is the minimum;
12  if node B is NULL then update A and return A;
13  for (distance AC. distance BC) in [(4, 6),(6, 6).(4, 4),(2, 2)]and sei C in NULL do
14          for i to m do
15              if ( $D_{nodeA,i}$,$D_{nodeB,i}$=)is (distance AC,distance AC) then set C append(i);
16          end
17  end
18  node C←the node in set C where $time_{node}$ is the minimum;
19  if node C is NULL then update A and return A;
20  update A;
21  allocate the rest replicas on random nodes with the minimum time;
22  end
23  return A;
```

Figure 2. on the greedy algorithm-based data block allocation algorithm

According to the above algorithm, it is seen that the algorithm first calculates the expected processing time of this file on different nodes by the distribution of data blocks it has reached; then, three steps are used to process the typical three-copy distribution separately. First, it searches for the node with the least required time - NodeA - and places the first copy; second, it searches for the node with distanceAB from NodeA as a candidate node for the second copy and selects the node with the least expected processing time as nodeB; third, it constructs the combination of nodeA and distance combinations of nodeA and nodeB, setting the priority as (4, 6) > (6, 6) > (4, 4) > (2, 2) [11]. According to this priority, finding the node that can place the third copy with the least expected processing time is possible. Thus, the above improvement shows that the placement time complexity of finding each copy is O(m) [12], which dramatically reduces the time overhead.

2.3. Node-failure-oriented data recovery strategy

Once a node fails, a copy of the data block is lost, and NameNode finds the lost copy of the data block by scanning the file and recovers it in time to prevent data loss. The lost block copies are uniformly appended to the recovery column BlockToReplication. Thus, in the case of cluster node failure, the goal of recovery is to redistribute these lost data blocks, and the data reliability needs to be satisfied.

2.3.1 Solving the objective function

Define a matrix D to describe the network distance between m nodes. Let each data block in the recovered data list be d, the total number of copy factors be Td, the total number of data blocks be nd, and the distribution of data blocks be Ad with size m*nd. The steps of data block copy recovery are: one is to find all available copy locations of data block d and select one of them as the copy source of the new copy. Denote by Wt the data recovery task being redistributed in node t and MaxLimit the maximum number of data recovery tasks running simultaneously on the node; second, find the lost data block d recovery location nodes as Nd. The data block recovery problem is defined as finding the set of recovery node locations Nd of the lost data block d and minimizing the expected processing time standard deviation of the nodes, from which then the following is obtained[13]：

$$\sigma_{time} = \sqrt{\frac{\sum_{i=0}^{m}(T_i^d - \mu^d)^2}{m}} \tag{9}$$

2.3.2 Data recovery solution

The solution algorithm in Figure 3 is proposed according to the data block allocation principle for the data block recovery problem. The algorithm solves the process by first determining the replica source nodes and selecting the set of target nodes for data recovery. Based on the position of the data block at the head of the current queue, all available copies of the data block are determined, and the node with the least amount of data block recovery tasks among them is selected as the replication source for data recovery. Subsequently, the distribution matrix A is updated based on the positions of all available replicas of the data block and the replicas that are about to be recovered. Based on matrix A, the nodes, and the processing capacity information, the expected processing time of each node in the data block locally is updated. Finally, based on the above-determined information about the replica source nodes and the pre-processing time of the nodes, the set of nodes to be recovered as lost replicas are found TargetNodes [14].

This paper can determine all available copies of the current data block regarding the data block at the current head location. From the node where the available copy location is located, the node with the least amount of recovery tasks for the current data block is selected as the replica source for data recovery, thus avoiding as much as possible too many recovery tasks in the same node and speeding up the data recovery progress. Subsequently, the distribution matrix A is updated based on the locations of all available replicas of all data blocks of the file to which the data block belongs and the locations of the replicas to be recovered [15]. Based on this matrix and the node and processing capacity information, the algorithm updates the expected local processing time of the node for the file belonging to this data block at each node. Finally, based on the determined source location nodes and the pre-processing time information of each node, the algorithm finds the set TargetNodes as the set of nodes for the recovery of the lost copy of this data block.

```
input:   D:Matrix of network distances between nodes
         P：Vectors of node processing power
         Block：The data block of the current queue head in the Block To Replication queue

         Distribution Matrix for Copies of All Survivable Data Blocks in the File to which
         $A^{block}$ Block belongs
Output:  srcNode: Block's Replication Source Node
         Target Nodes: Target recovery node for new copy of block
1   Live Nodes ←get Live Replicas Nodes(block);
2   srcNode←the node with the minimum work and (node.work ≦Max Limit) in live Nodes;
3   update $A^{block}$ according to the information of blocks which has chosen the target Nodes;
4   update $T_i$ according to the information of ,$A^{block}$ and P //using the equation 4.2;
5   Target Nodes ←choose Targel Node(src Node, block);
6   return src Node, target Nodes;
7   Function choose SrcNode (block)
8   if block, num Live Replicas ==I and block numNeedReplicas >0 then
9           dnl← the node whose processing time is minimum and where $D_{sreNode,dn1}$==6
10          targetNodes.append(dn I);
11  end
12  if-numNeedRcpIicas ==0 then return target Nodes;
13  if block.numLiveReplicas ≦2 then
14          if $D_{srcNode.dni}$ == 6 then
15              dn2←the node whose processing lime is minimum and where $D_{dn2,dn1}$==4
                or$D_{dn2,srcNsde}$==4;
16          else
17              dn2←the node whose processing time is minimum and where $D_{dn2dnl}$ == 6;
18          end
19          Target Nodes.append(dn2);
20  end
21  choose the nodes whose processing time is minimum for additional replicas;
22  return target Nodes;
```

Figure 3. Data Recovery Solving Algorithms

3 Experimental evaluation

3.1. Data reliability measurement

The Randon Writer tool is utilized to generate datasets of size 4GB, 8GB, and 16GB, and these three datasets are written to the HDFS cluster based on the greedy algorithm-based data block allocation algorithm and the default data block placement algorithm. The experimental data blocks are set to 64GB with a 3-replica factor, and the data reliability metrics of these three datasets are counted, and the results are shown in Figure 4. Based on the default data block allocation algorithm, there are only one or two copies of data blocks in different physical machines, and the data blocks that do not achieve the desired reliability are 29.7% on average. In contrast, based on the location-aware data block placement algorithm proposed in this paper, the data blocks were all fee allocated to different physical machine nodes and the data blocks all achieved reliability. Subsequently, the data reliability metrics after recovering the data in different algorithms were tested by simulating node failures in the experiments. The datasets were written into two different algorithms-default and data block copy recovery algorithms- and obtained in Fig. 5. The results show that the data fast recovery data reliability metrics are low based on the default algorithm in HDFS, while the ratio of three copies across different physical machines based on the algorithm in this paper reaches 100%, which proves the effectiveness of the algorithm for recovering data block copies.

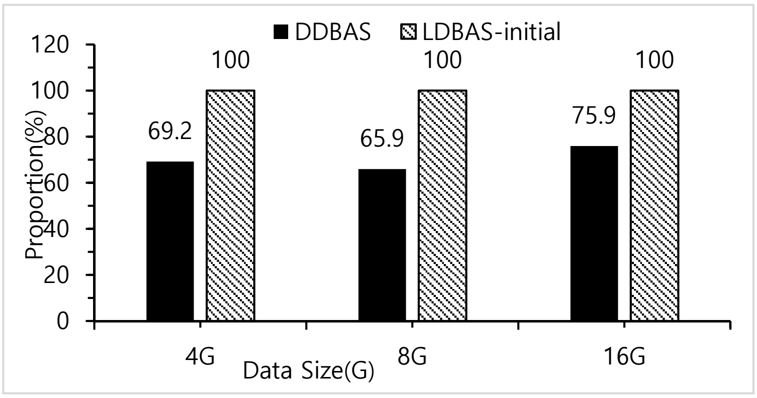

Figure 4. Comparison of data reliability after data initialization and writing

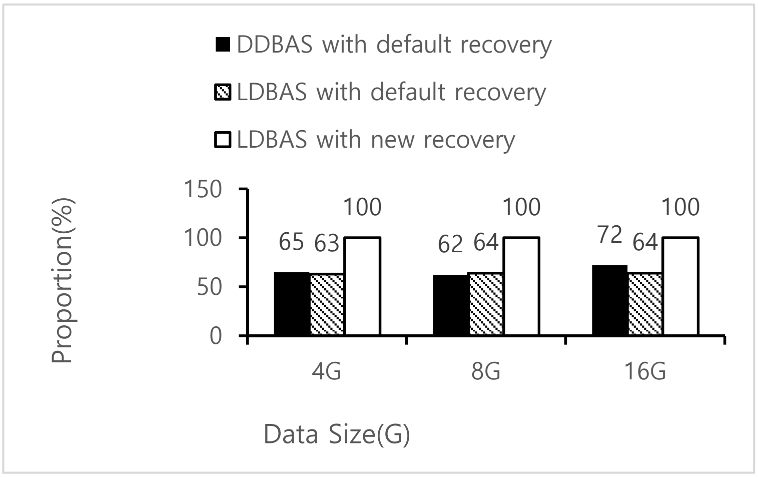

Figure 5. Comparison of data reliability after data recovery

3.2. Algorithm time overhead measurement

The time overhead of the location-aware data block placement algorithm is measured in a real cluster environment, and the HDFS source code is modified to record the response time of the algorithm to data block write requests, as shown in Figure 6. From the figure, we can see that based on the algorithm proposed in this paper, the response time of data block request writing is between 1.5 ms to 2 ms [16], there is no additional time overhead, and it can meet the real-time requirement of data block writing request.

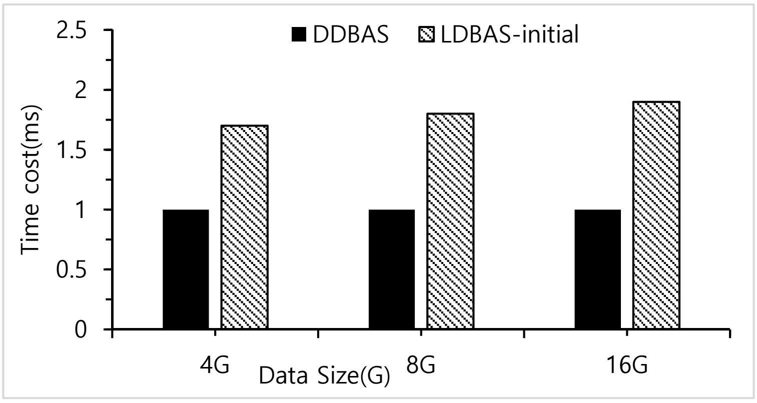

Figure 6. Time overhead of the location-aware data block placement algorithm for a single request

3.3. Efficiency measurement of the algorithm in real application jobs in Hadoop

Whether the algorithm can impact real big data processing applications and, the algorithm needs to be verified. The BigDateBench tool was used to test the two types of applications, Sort and WordCount, and the percentage of map task execution and job completion time of the two applications were counted.

The Sort application was applied to sort three datasets of different sizes (4GB, 8GB, and 16GB) with 65, 129, and 257 map tasks, respectively. Each of these three datasets was run 10 times, and the percentage of map tasks and job completion time were counted. The obtained results are shown in Figure 5 and Figure 6. From the figures, it can be seen that the task proportion increases by 2.1% and reduces the job completion time by 8.9%. From these two figures, it is clear that the location-aware data block placement algorithm based on this paper can improve the task proportion and reduce the job completion time.

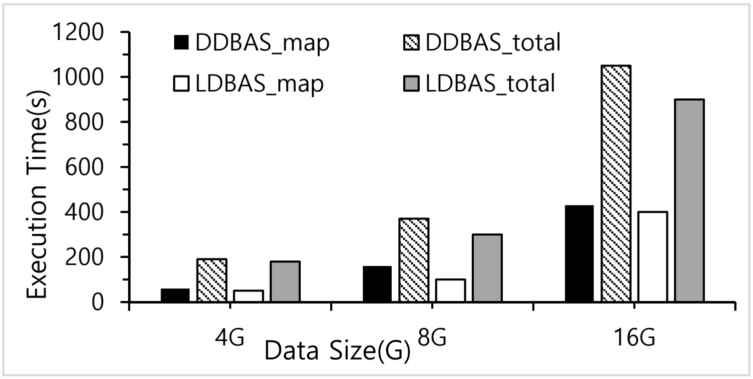

Figure 7. Sort assignment completion time

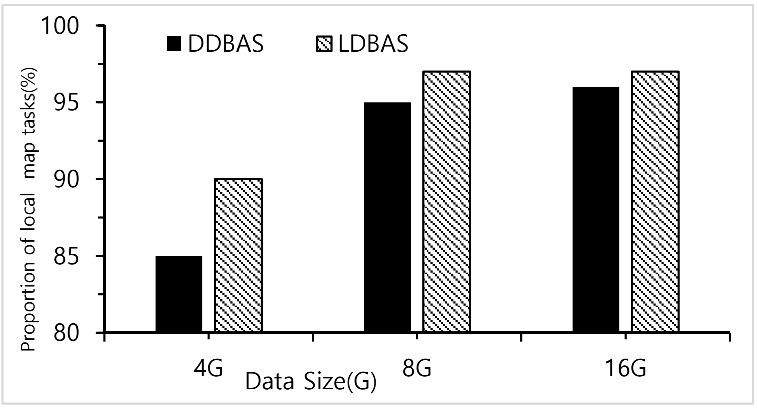

Figure 8. Sort processing map task ratio

WordCount is used similarly to the Sort application, and a data set of inconsistent size is written, and the WordCount application is run 10 times to count the related performance metrics. The results are shown in Figure 9 and Figure 10. As can be seen from the figures, the location-aware data block placement algorithm reduces the job completion time by 2.8%. Since WordCount is a central processor-intensive application with relatively few input and output operations, it is ineffective in reducing the job completion time. However, it increases the proportion of local map tasks, which is beneficial to reduce the additional network overhead.

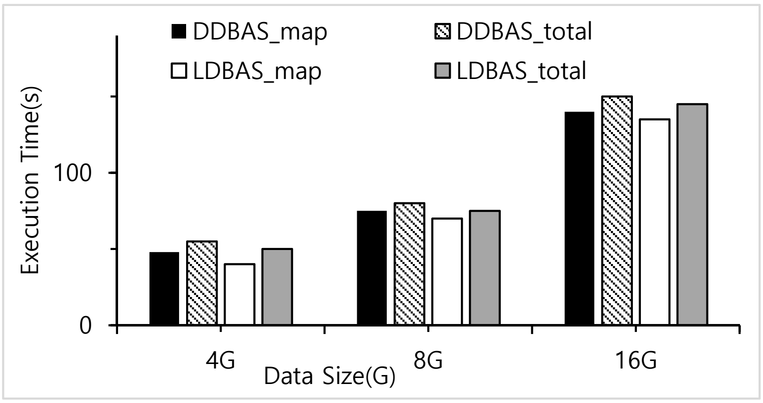

Figure 9. WordCount Assignment Completion Time

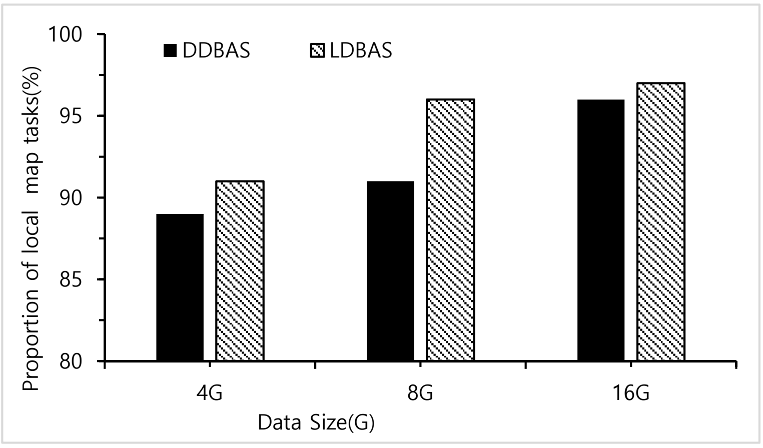

Figure 10. WordCount task ratio

4. CONCLUSION

The above tests prove that the location-aware data assignment algorithm proposed in this paper can improve data reliability and processing performance and has effectiveness on data block assignment location. In addition, The fast data recovery algorithm proposed in this paper is designed to recover lost copies of data blocks., reduce time overhead and job completion time, and improve task ratio, which impacts the big data center processing systems and optimize its performance system. However, this study is small in scope and does not delve into other issues of optimizing data processing performance, which needs to be further improved and then optimize the big data center processing system from all aspects in a comprehensive manner.

Conflicts of Interest: The authors declare no conflict of interest

References

[1] M. Markel, Writing in the Technical Fields-A Step-by-Step Guide for Engineers, Scientists, and Technicians, IEEE Press, New York, 1994.

[2] Y. S. Park, B. N. Yoon, and J. H. Lim, "Comparing Fault Prediction Models Using Change Request Data for a Telecommunication System," ETRI Journal, vol. 21, no. 3, pp. 6-15, Sep. 1999, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[3] M. M. Chiampi and L. L. Zilberti, "Induction of electric field in human bodies moving near MRI: An efficient BEM computational procedure," IEEE Trans. Biomed. Eng., vol. 58, pp. 2787-2793, Oct. 2011, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[4] R. Fardel, M. Nagel, F. Nuesch, T. Lippert, and A. Wokaun, "Fabrication of organic light emitting diode pixels by laser-assisted forward transfer," Appl. Phys. Lett., vol. 91, no. 6, Aug. 2007, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[5] J. Zhang and N. Tansu, "Optical gain and laser characteristics of InGaN quantum wells on ternary InGaN substrates," IEEE Photon. J., vol. 5, no. 2, Apr. 2013, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[6] W. P. Risk, G. S. Kino, and H. J. Shaw, "Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle," Opt. Lett., vol. 11, no. 2, pp. 115-117, Feb. 1986, doi: http://dx.doi.org/10.1000/0-000-00000-0. [Online] Available:http://ol.osa.org/abstract.cfm?URI=ol-11-2-115

[7] D. Caratelli, M. C. Viganó, G. Toso, and P. Angeletti, "Analytical placement technique for sparse arrays," presented at *the 32nd ESA Antenna Workshop*, Noordwijk, The Netherlands, Oct. 2010, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[8] T. Schubert, "Real challenges and solutions for validating system-on-chip high level formal verification of nextgeneration microprocessors," in *Proc. 40th Design Automation Conf. (DAC'03)*, Jun. 2003, doi: http://dx.doi.org/10.1000/0-000-00000-0, [Online] Available: http://www.computer.org/csdl/proceedings/dac/2003/2394/00/2394001-abs.html

[9] A. Amador-Perez and R. A. Rodriguez-Solis, "Analysis of a CPW-fed annular slot ring antenna using DOE," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, pp. 4301-4304, Jul. 2006, doi: http://dx.doi.org/10.1000/0-000-00000-0.

[10] J. O. Williams, "Narrow-band analyzer," Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, 1993.

[11] N. Kawasaki, "Parametric study of thermal and chemical nonequilibrium nozzle flow," M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[12] N. M. Amer, "The effects of homogeneous magnetic fields on developments of tribolium confusum," Ph.D. dissertation, Radiation Lab., Univ. California, Berkeley, Tech. Rep. 16854, 1995.

[13] L. Breimann, Manual on Setting Up, Using, and Understanding Random Forests v4.0. (2003). Accessed: Apr. 16, 2014. [Online] Available: http://oz.berkeley.edu/users/breiman/Using_random_forests_v4.0.pdf

[14] M. Kuhn, The Caret Package. (2012). [Online] Available: http://cranrproject.org/web/packages/caret/caret.pdf

[15] Antcom, Torrance, CA, USA. Antenna Products. (2011). Accessed: Feb. 12, 2014. [Online] Available: http://www.antcom.com/documents/catalogs /L1L2GPSAntennas.pdf

[16] Antenna Products. (2011). Antcom. Accessed: Feb. 12, 2014. [Online] Available: http://www.antcom.com/documents/catalogs/L1L2GPSAntennas.pdf

[17] Ngspice. (2011). [Online] Available: http://ngspice.sourceforge.net

[18] IEEE Std. 1394, IEEE Standard for a High Performance Serial Bus, IEEE, Piscataway, N.J., 1995.