An Optimization Algorithm for Heterogeneous Hadoop Clusters Based on Dynamic Load Balancing

Wei Yan, ChunLin Li, ShuMeng Du, XiJun Mao
Software Engineering
Wuhan University of Technology
No.1186, Heping Boulevard, Wuchang District, Wuhan, Hubei
CHINA
364560925@qq.com

Abstract—**Hadoop is a popular cloud computing software, and its major component MapReduce can efficiently complete parallel computing in homogeneous environment. But in practical application heterogeneous cluster is a common phenomenon. In this case, it's prone to unbalance load. To solve this problem, a model of heterogeneous Hadoop cluster based on dynamic load balancing is proposed in this paper. This model starts from MapReduce and tracks node information in real time by using its monitoring module. A maximum node hit rate priority algorithm (MNHRPA) is designed and implemented in the paper, and it can achieve load balancing by dynamic adjustment of data allocation based on nodes' computing power and load. The experimental results show that the algorithm can effectively reduce tasks' completion time and achieve load balancing of the cluster compared with Hadoop's default algorithm.**

Keywords—Hadoop; heterogeneous cluster; data allocation; load balancing;

I. Introduction

Cloud computing was first popularized in 2006 by Amazon, which can provide virtualization, dynamic resource pool and high availability [1][2]. Hadoop is a popular cloud computing platform based on HDFS and MapReduce[3].Its main component MapReduce can implement parallel computing of multiple nodes, then returns results to meet the needs of users, and it has high fault tolerance[4][5].

In the process of MapReduce, the Master node sets tasks to each Slave node in average through the default algorithm, and Hadoop's default balance strategy is make each node's load rate consistent. The strategy is generally effective in a homogeneous environment [6].However, in order to make full use of the system resources, the storage space of each node may not be the same. If the computing capacity of nodes is added to the problem, the default strategy of Hadoop obviously can't meet the requirements of cluster's load balancing. For example, if a node's computing power is weak but its storage space is very large, the default strategy will make the node become a high load node which determines the efficiency of cluster. In this case, the default scheduling policy is invalid[7].So load balancing is an important index to evaluate the performance of MapReduce even the entire Hadoop cluster[8].

The main contributions of this paper for the problem are as follows:(1) In order to study the characteristics of Hadoop's heterogeneous cluster, this paper design and implement a heterogeneous Hadoop cluster model based on dynamic load balancing by real-time monitoring. (2) In order to reduce the negative impact of heterogeneous cluster's unbalance load, a maximum node hit rate priority algorithm (MNHRPA) is proposed and implemented in the model to achieve load balancing and improve the efficiency of cluster.

The remainder of this paper is organized as follows: Related work is introduced in Sec. II. A heterogeneous Hadoop cluster model based on dynamic load balancing is built in Sec. III. Sec. IV. describes and implements MNHRPA. Sec. V evaluates the effectiveness of MNHRPA compared with the default priority algorithm and another greedy priority algorithm through experiment. Summary and prospect are proposed in Sec.VI.

II. Related Work

The efficiency of Hadoop is very high in the homogeneous cluster. If each node's storage and the computing ability are the same, the result can be fine. However, for the nodes differences in computing power and storage capacity common situation, which causes the problem of unbalance load [9].A lot of problems can be caused when the data load among nodes becomes unbalanced. It will indirectly lead to reduced efficiency of the cluster, and the execution time of the job will become longer [10]. At home and abroad, many scholars have researched Hadoop's optimization algorithms based on the problem, mainly from two directions, as the adjustment before data allocation and the adjustment after data allocation.

The scholars have done a lot of research in the way of the adjustment before data allocation. For example, reference [11] takes the heterogeneity of nodes into account, and puts forward a placement strategy according to the proportion of the storage of data, and reference [12] considers that the data generated by the reduce phase can be divided into more partitions, adjusts the partition size to balance the load by dynamically, but they don't consider the influence of nodes' computing power differences in a heterogeneous cluster. Reference [13] proposes a greedy priority algorithm, which adjusts the load by the calculation ability of each node in advance, but it doesn't take the dynamic changes in the computing power of the nodes into account.

In addition, the scholars also have a lot of efforts for the adjustment after data allocation. Reference [14] proposes a data migration strategy based on node's load regulation under heterogeneous environment. In reference [15], Fan et al consider to change the map task's assignment to reduce unbalance load's effect on the performance of the cluster. These methods have some effect on load balance of each node. However, they do not take into account these methods will greatly increase the mobility of data, which has a great impact on the performance of the cluster and data security.

In conclusion, most scholars have researched unilateral problem of cluster's heterogeneity, and few combine them. This is not very suitable for heterogeneous cluster model in the actual environment. Based on it this paper starts from MapReduce and take problems into account from the two aspects of computing power and storage capacity of nodes, to design and implement a heterogeneous Hadoop cluster model based on dynamic workload adjustment by real-time monitoring, which can achieve load balancing and improve cluster performance.

III. Model And Method

A. The heterogeneous cluster model based on dynamic load balancing

In order to achieve load balancing and improve cluster performance in heterogeneous cluster, this paper adds the monitoring module and rewriting a new selection algorithm to change the task process of MapReduce. A heterogeneous Hadoop cluster model based on dynamic load balancing can be described by Figure 1.

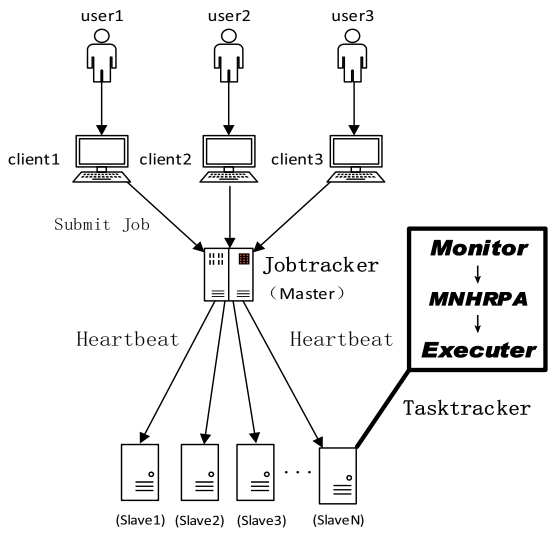

Fig. 1. The overview of the model

As shown in Figure 1, after the user submits the job through the client, the Hadoop's Master node receives jobs and begins Job distribution through the MapReduce process to the Slave nodes. When the master's jobtracker receives heartbeats from Slave's tasktrackers, it will assign suitable tasks to tasktrackers. Heartbeats are passed between the jobtracker and the tasktrackers. The black-bordered box is the most important part of this paper, and that is the refinement of the Slave's tasktracker in MapReduce. Each tasktracker has its own monitoring module and adjust its load dynamically through MNHRPA.

B. Formal description of related parameters and definitions

1) Related definitions and formal description of the nodes

a) Set node in the model to x:Saved with Set J, $J(x)$ is used to represent one node and so $J(1)$ can represent the first node, and so on. Total number of nodes is expressed as N_{node}. Then $x \in [1, N_{node}]$.

b) Set node's CPU performance to P_{cpu}:

$$P_{cpu}(x) = \begin{cases} f_{cpu}(x) \times n_{core}(x) \times \varepsilon (n_{core} > 1) \\ f_{cpu}(x) \qquad (n_{core} = 1) \end{cases} \quad (1)$$

Where f_{cpu} is CPU's basic frequency, n_{core} is CPU's core number. The model selects the two main parameters to evaluate the performance of CPU.Ɛ is a parameter when CPU's core is not only 1,and its reasonable value is generally between 0.8~0.9 according to references.

c) Set node's memory performance to P_{ram}:

$$P_{ram}(x) = size_{ram}(x) \quad (2)$$

Where $size_{ram}$ is size of the memory's RAM. The model selects the main parameters to evaluate the performance of memory.

d) Set storage capacity to S_{node}:

The node's storage in cluster is available for use in the HDFS file system, so $S_{node}(x)$ is storage capacity of the node x.

e) Set used capacity to U_{node}:

The node's storage in the cluster has been used in the HDFS file system, so $U_{node}(x)$ is used storage capacity of the node x.

2) Related definitions and formal description of the model

a) Set node's relative computing power to P_{node}:

$$P_{node}(x) = \alpha \frac{P_{cpu}(x)}{\min(P_{cpu})} + \beta \frac{P_{ram}(x)}{\min(P_{ram})} \quad (3)$$

As shown in the formula (3), it's an important definition of the ratio in the model. α,β are resources' weight coefficients, they represent respectively the important proportion CPU and memory .According to the definition, it can be known that $\alpha+\beta=1$.By consulting information and research, Hadoop's task types can be divided into CPU-intensive and I/O-intensive, thus the value of α,β will change with the change of task types, obviously α in the CPU-intensive tasks accounted for a relatively high,β in the I/O-intensive tasks accounted for relatively high. For example, in CPU-intensive task WordCount they should be defined $\alpha=0.8$, $\beta=0.2$.

b) Set the load ratio of node to $R_{node}(x)$:

$$R_{node}(x)=\frac{U_{node}(x)}{S_{node}(x)}\times 100\% \tag{4}$$

This definition is the percentage of the used storage of the node accounted for the total storage of the cluster.

c) Set the load ratio of cluster to $R_{cluster}$:

$$R_{cluster}=\frac{U_{cluster}}{S_{cluster}}\times 100\%=\frac{\sum_{x=1}^{N_{node}}U_{node}(x)}{\sum_{x=1}^{N_{node}}S_{node}(x)}\times 100\% \tag{5}$$

As shown in the formula (5), this value is an important index of cluster's load balancing. That should be the used capacity of all nodes in the cluster to occupy the percentage of the storage capacity of all nodes.

d) Set maximum load rate to R_{max}:

$$R_{\max}=[\gamma+(1-\gamma)\times R_{cluster}]\times 100\% \tag{6}$$

As shown in the formula (6), researches show that clusters of long time in full load operation have great influence on the performance of cluster. So in order to ensure the success rate of the task, this paper sets a threshold, that is, the maximum load rate defined here. When one node load exceeds the threshold value, Hadoop system should not continue to assign tasks to this node. γ is a parameter that is set by the MapReduce system, whose default value is 0.8.When the cluster is fully loaded, γ=1,so the maximum load rate becomes 100% automatically.

e) Set node hit rate to R_{hit} :

$$\mathrm{R}_{hit}(x)=\frac{P_{node}(x)/U_{node}(x)}{\max(P_{node}/U_{node})}\times 100\% \tag{7}$$

As shown in the formula (7), it's a key parameter. If a node's computing power is stronger and it has less used capacity, then the node's hit rate is higher.

C. Optimization of priority algorithm

1) Implementation of monitoring module

Ganglia is an open source's cluster monitoring project launched by Berkeley UC, which is designed to measure the cluster's node information. It can be used to monitor the performance of the system, such as: CPU, memory, hard disk utilization, I/O load, network traffic, etc. The information in this paper such as the number of nodes, load rate and so on can be directly or indirectly acquired through the project. Thus the monitoring module of the Hadoop cluster model can be implemented through the rational use of the function of Ganglia and modify the Hadoop's scheduling code.

2) Selection of slave nodes

After completing the definition and parameters, the model can achieve an optimized priority algorithm: When the job has been submitted and enters the waiting allocation stage, the model Create three Hashtable type's collections named Nodes1、Nodes2 and Nodes3,which store$<x,R_{hit}(x)>$ of the nodes as *<key, value>* key-value pair. Then, according to $R_{node}(x)$, $R_{cluster}$ and R_{max}, The nodes in the cluster are allocated to three Hash tables according to the rules of Table I.

TABLE I. THE CONDITION OF NODE GROUPS

Storage mode	Storage content
Hashtable	Nodes1：$\{x\mid R_{node}(x)\le R_{cluster}\}$ Nodes2：$\{x\mid R_{cluster}<R_{node}(x)<R_{max}\}$ Nodes3：$\{x\mid R_{node}(x)\ge R_{max}\}$

The utilization rate of the node in Node3 is higher than the maximum load rate, so a new task shouldn't be assigned into it.The node in Node3 can send the heartbeat messages of refusal of receiving task to the Master node. Sort notes in Nodes1 and Node2 according to the value that nodes' hit rate from high to low. When the new task is assigned to the Reduce stage, the first node of the Node1 is chosen. If the node is unable to meet the requirements of storage space, then select the second nodes of the hashtable1, and so on. If the Nodes1 is empty, select the node in the Nodes2 in the same way. When the Slave node have been chosen successfully, it can send the heartbeat messages of preparation for receiving task to the Master node. Finally the Master node can complete the task allocation.

3) Allocation of the tasks

After each successful assignment of a new task, the node data should be updated in the three tables by reacquiring the monitoring information, then repeat the process of selection of reduce nodes for the next task assignment. If the task can't be assigned to a proper node by the process, it is indicated that the cluster is fully loaded now, and it needs to wait for the cluster to perform the task. After the whole tasks are assigned, the algorithm is over.

Ⅳ. IMPLEMENTATION OF ALGORITHM MNHRPA

This section introduces the maximum node hit rate priority algorithm (MNHRPA) and list its pseudo code described in Table II, the explanation about the pseudo code is given. At last the paper conducted a time complexity analysis of the algorithm.

Lines 1~5 represent when a job complete the submission phase, the model firstly gets the current resource usage information of each node through the monitoring module, through the process parameters and preconditions of the initialization algorithm can be built. Lines 6~28 describe the specific process of the algorithm. According to the difference of the node load, each node is assigned to the three hash table. Then the model can select the appropriate nodes for task allocation by using the node's load rate and hit rate. Line 29~30 indicates allocation of the tasks after a successful assignment of new task. After repeating the node selection methods until all tasks are assigned. The cluster can continue to the next phase and the algorithm is over.

TABLE II. THE PSEUDO CODE OF MNHRPA

```
Input: Monitoring information
        $R_{node}(x)$:the load ratio of node,$R_{cluster}$:the load ratio of cluster
        $R_{max}$:maximum load rate,$R_{hit}(x)$:node hit rate,$x$:node number
Output: Statistical  results of task allocation of nodes

Steps:
1.   Number the nodes and receive a new job
2.   Run the job and process tasks
3.   Get nodes' current resource information
     through the monitoring module
4.   Calculate $R_{node}(x)$、 $R_{cluster}$、 $R_{max}$ and $R_{hit}(x)$
5.   Create three Hashtables:Nodes1、Nodes2 and Nodes3
     //<Key,Value>→<$x$, $R_{hit}(x)$>
6.   for each node $x$
7.     if $R_{node}(x)≥R_{max}$
8.       Put <$x$, $R_{hit}(x)$> into Nodes3,ignore these nodes
9.     else
10.      if $R_{node}(x)>R_{cluster}$
11.          Put <$x$, $R_{hit}(x)$> into Nodes2
12.      else
13.        Put <x, $R_{hit}$(x)> into Nodes1
14.  end for
15.  if  Nodes1 isn't empty
16.    Sort descending the Nodes1 by value
17.    for each node in Nodes1
18.      if the node satisfies the condition
19.        Select the node as the result
20.    end for
21.  else
22.    Sort descending the Nodes2 by value
23.    for each node in Nodes2
24.      if the node satisfies the condition
25.        Select the node as the result
26.    end for
27.  Choose the selected node and send the heartbeat messages
28.  Schedule the current task to the selected node
29.  Update the information of all nodes and repeat step3~step28
30.  Run the task and collect result
```

Through the analysis of the above algorithm, it can know that its time complexity is O (n) which compare the size of cluster utilization and classify all the nodes into three Hash tables. And its time complexity is O ($nlog_2n$) which uses the Collections.Sort method to sort the three hash table according to its value. So according to the definition of the time complexity of the algorithm, the time complexity of the improved algorithm is O ($nlog_2n$).It is better than the same type of greedy priority algorithm whose time complexity is O (n^2).

Ⅴ. EXPERIMENTAL DESIGN AND ANALYSIS

A. Experimental design

In order to verify the feasibility and effectiveness of the proposed algorithm (MNHRPA), in this paper, a heterogeneous Hadoop cluster whose each node's configuration is different is built, and classical test cases WordCount and TeraSort are used to carry out the experiment. The experimental results show the advantage of the algorithm.

1) Experimental environment

In this experiment, the total heterogeneous cluster with 6 nodes, there are differences in the hardware configuration. Its specific configuration chart is as follows Table III.

The 6 physical machines are installed Ubuntu14.04 operating system, and its Hadoop version is 0.20.2.After the understanding of the Hadoop copy placement strategy and setting methods. This paper decides to put 6 nodes in a rack. And the number of copies is set to 1.Then the effects of Hadoop copy scheduling on the scheduling algorithm can be eliminated.

2) Test case

In order to obtain more reasonable experimental results, in this paper, mature CPU-intensive task WordCount and I/O-intensive task TeraSort are chose to complete the comparison experiment. In order to ensure the accuracy of the results, several sets of different data are randomly acquired.

3) Evaluation index

Because the algorithm used in this paper is mainly used in heterogeneous clusters, and the algorithm is proposed for the load balance of the cluster and shorten the execution time of the job. Therefore, the main evaluation indexes should be focused on cluster's load condition and job's completion time.

TABLE III. HETEROGENEOUS HADOOP CLUSTER'S NODE CONFIGURATION

Number	CPU	Core	Memory	HDFS	Role	IP
1	3.3GHz	4	8G	80GB	Master	192.168.1.10
2	3.0GHz	2	4G	40GB	Slave1	192.168.1.11
3	2.3GHz	1	4G	60GB	Slave2	192.168.1.12
4	1.9GHz	1	2G	30GB	Slave3	192.168.1.13
5	2.3GHz	2	2G	20GB	Slave4	192.168.1.14
6	2.4GHz	2	4G	40GB	Slave5	192.168.1.15

4） Benchmark algorithm

The maximum node hit rate priority algorithm (MNHRPA) proposed by the paper is compared with the Hadoop default priority algorithm (DPA) and the greedy priority algorithm (GPA) proposed by reference [13].They are set the appropriate type of operation and the appropriate parameters to carry out the experiment.

B. Experimental results and analysis

In order to prove that the optimization algorithm proposed in this paper can balance the load and improve the efficiency of the cluster, this paper designs and carries out the following comparison experiments from two aspects of the job's completion time and the cluster's load condition. In the following experiments the parameters have been set that Ɛ=0.9 in formula (1) and γ=0.8 in formula (6).

1） The comparison experiment of job's completion time

The paper selects the job WordCount and TeraSort to complete experiment. The job's configuration including α,β in formula (3) is displayed in table IV in detail:

TABLE IV. JOB'S CONFIGURATION IN THE EXPERIMENT

Job's Type	Job's Size					Job's Parameter
WordCount	200M	400M	2G	4G	10G	α=0.8,β=0.2
TeraSort	200M	400M	2G	4G	10G	α=0.2,β=0.8

Through the configuration of the comparison experiment of job's completion time, in this paper, the experiment was carried out successfully, and the comparison of the three algorithms was drawn.

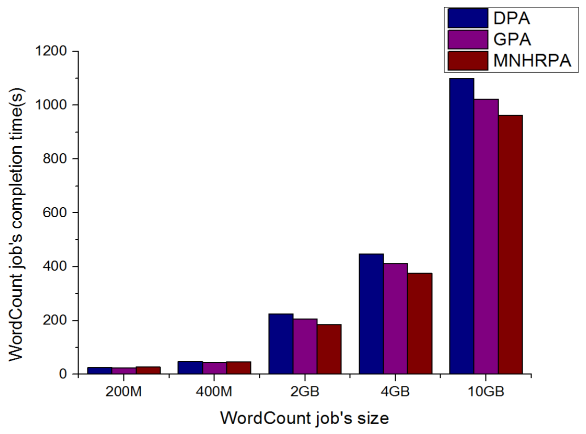

Fig. 2. Comparison of the completion time of the data sets with different sizes under WordCount

Figure 2 and Figure 3 respectively show that using three kinds of algorithm's execution of 200M, 400M, 2G, 4G, 10G sizes of WordCount ($\alpha=0.8,\beta=0.2$) and TeraSort ($\alpha=0.2,\beta=0.8$) to the comparison of job completion time.

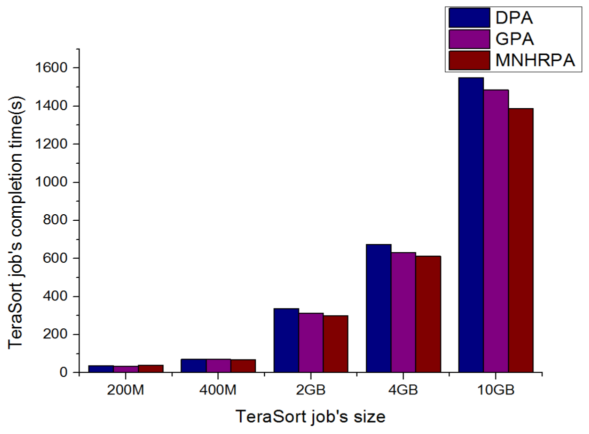

Fig. 3. Comparison of the completion time of the data sets with different sizes under TeraSort

According to the two comparison experiment some results can be indicated: With the increase of the amount of data, the efficiency of the use of MNHRPA will be more and higher with the increase in the size of the job, and its efficiency is better than GPA. In the data to 10G, compared with DPA, it can reduce the execution time of about 15.6% in WordCount and 10.7% in TeraSort.

2） The comparison experiment of cluster's load condition

The paper selects the job WordCount and TeraSort to complete experiment. The job's configuration is the same as table IV except that job's size is just 20G, and the job is executed ten times in a row.

Through the configuration of the comparison experiment of cluster's load condition, in this paper, the experiment was carried out successfully, and the comparison of the three algorithms was drawn.

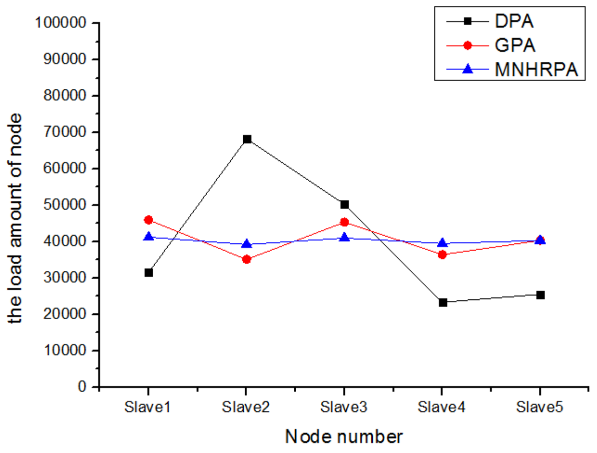

Fig. 4. Load comparison on different nodes in 20G data under WordCount

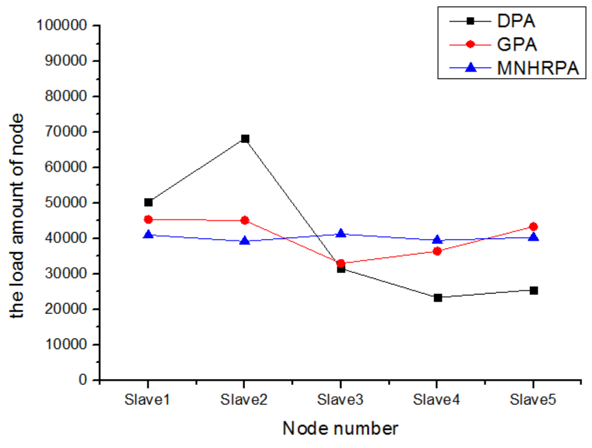

Fig. 5. Load comparison on different nodes in 20G data under TeraSort

Figure 4 and Figure 5 respectively show that using three kinds of algorithm's execution of 20G size of WordCount (*α=0.8,β=0.2*) and TeraSort (*α=0.2,β=0.8*) to the comparison of cluster's load condition. Then some results can be indicated MNHRPA is essentially the same for each Reduce load, while the load of each Reduce using DPA has a large data fluctuation. GPA is better than DPA, but it is less than MNHRPA.MNHRPA's load balancing degree increases 44.4% compared with DPA and its job's execution time reduces 8.4% compared with GA.

Through the above experiment, some conclusion can acquired that in terms of load and time efficiency, the improved algorithm MNHRPA is obviously better than the default algorithm.

Ⅵ. Summary And Prospect

After study the Hadoop's default algorithm is found that it is not adapted to heterogeneous cluster, and it is easy to prone to unbalance load. A heterogeneous Hadoop cluster model dynamic load balancing is proposed. The model uses the monitoring module to monitor the indicators of the node in real time, and through the design of the improved algorithm to change the task's node allocation, so that the cluster can achieve load balance. Through the experiment can be seen that the designed algorithm can effectively balance the cluster load and speed up the job completion. The effect will be more obvious when the cluster size is larger and the heterogeneous situation is more complex.

In future research, the research on the load balancing of heterogeneous clusters should be paid more attention. Special attention is paid to the influence and optimization of different rack and replication strategy on heterogeneous clusters.

Acknowledgment

The work was supported by the National Natural Science Foundation (NSF) under grants (No.61472294, No.61672397) , Key Natural Science Foundation of Hubei Province (No.201 4CFA050).

Reference

[1] Mell P, Grance T. The NIST definition of cloud computing [J].Communications of the ACM, 2011, 53(6):50-50.

[2] Aggarwal M. Introduction of Cloud Computing and Survey of Simulation Software for Cloud [J]. TIJ's Research Journal of Science & IT Management - RJSITM, 2013, 2(12).

[3] Taylor R C.An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics[J]. Bmc Bioinformatics, 2010, 11 Suppl 12(Suppl 12):3395-3407.

[4] Mmel R. Google's MapReduce programming model — Revisited[J]. Science of Computer Programming, 2008, 70(1):1-30.

[5] Chen H, Luo W, Wang W, et al. A novel real-time fault-tolerant scheduling algorithm based on distributed control systems[C].Computer Science and Service System (CSSS),2011 International Conference on. IEEE, 2011:80-83.

[6] Zaharia M, Konwinski A,Joseph A D, et al. Improving MapReduce Performance in Heterogeneous Environments.[J]. Osdi, 2008:29-42.

[7] Ananthanarayanan G, Agarwal S, Kandula S, et al. Scarlett: coping with skewed content popularity in MapReduce clusters[J]. In EuroSys, 2011:287-300.

[8] Srirama S N, Jakovits P, Vainikko E. Adapting scientific computing problems to clouds using MapReduce[J]. Future Generation Computer Systems, 2012, 28(1):184-192.

[9] Xie GL, Luo SX. Study on application of MapReduce model based on Hadoop[J]. Microcomputer & Its Applications, 2010.

[10] Kwon Y C, Balazinska M, Howe B, et al. SkewTune in action: mitigating skew in MapReduce applications[J]. Proceedings of the Vldb Endowment, 2012, 5(12):1934-1937.

[11] Farhat, F., Tootaghaj, D., He, Y., Sivasubramaniam, A., Kandemir, M. and Das, C., 2016. Stochastic modeling and optimization of stragglers. IEEE Transactions on Cloud Computing.

[12] Slagter K, Hsu C H, Chung Y C, et al. An improved partitioning mechanism for optimizing massive data analysis using MapReduce[J].Journal of Supercomput- ing, 2013, 66(1):539-555.

[13] Chen R, Zeng W H, Fan K J. Research on Hadoop Greedy Scheduler Based on the Fair[J]. Applied Mechanics & Materials, 2011, 145:460-464.

[14] Shen Q, Zhang L, Yang X, et al. SecDM: Securing Data Migration between Cloud Storage Systems[C].IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing. IEEE Computer Society, 2011:636-641.

[15] Fan Y, Wu W, Cao H, et al. A Heterogeneity-aware Data Distribution and Rebalance Method in Hadoop Cluster[C].Chinagrid Conference. 2012:176-181.