Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

December 2013, 20(6): 77–87
www.sciencedirect.com/science/journal/10058885

The Journal of China Universities of Posts and Telecommunications

http://jcupt.xsw.bupt.cn

MapReduce optimization algorithm based on machine learning in heterogeneous cloud environment

LIN Wen-hui[1,2,3] (✉), LEI Zhen-ming[1,2], LIU Jun[1,2], YANG Jie[1,2], LIU Fang[1,2], HE Gang[1,2], WANG Qin[4]

1. Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. School of Information Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
3. Technology Research Institute, Aisino Corporation, Beijing 100195, China
4. School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract

We present an approach to optimize the MapReduce architecture, which could make heterogeneous cloud environment more stable and efficient. Fundamentally different from previous methods, our approach introduces the machine learning technique into MapReduce framework, and dynamically improve MapReduce algorithm according to the statistics result of machine learning. There are three main aspects: learning machine performance, reduce task assignment algorithm based on learning result, and speculative execution optimization mechanism. Furthermore, there are two important features in our approach. First, the MapReduce framework can obtain nodes' performance values in the cluster through machine learning module. And machine learning module will daily calibrate nodes' performance values to make an accurate assessment of cluster performance. Second, with the optimization of tasks assignment algorithm, we can maximize the performance of heterogeneous clusters. According to our evaluation result, the cluster performance could have 19% improvement in current heterogeneous cloud environment, and the stability of cluster has greatly enhanced.

Keywords cloud computing, MapReduce, machine learning, heterogeneity

1 Introduction

In recent years, with the development of information technology and the explosive growth of global data, large data analysis business has produced great challenges to various research institutions and companies. And the challenges further promote the development of cloud computing technology. Hadoop is an open-source distributed computing framework, which is used for distributed processing of large data sets and designed to satisfy clusters scaled from single server to thousands of servers. Hadoop is the most widely used cloud computing platform in recent years and has been adopted by major Internet companies and research institutions. The core technology of Hadoop [1] includes MapReduce and Hadoop distributed file system (HDFS), which are inspired by Google's MapReduce and google file system (GFS). As we'll talk a lot about it later, MapReduce is a distributed computing framework which mainly focus on large-scale parallel processing in cloud computing.

Hadoop has the advantages of high reliability, high scalability and high tolerance. Open source is the greatest advantage of Hadoop, which can provide a low-cost solution for processing large data. However, the initial hardware environment in Hadoop is usually homogeneous, which means that each node in the cluster has the same computing power and tasks in each node have the same operation rate. However, when obsolete hardware replaced by new ones with the development of cloud computing technologies, the homogenous environment will slowly evolve into heterogeneous environment and the nodes' performance will become inconsistent [2–3]. When tasks are assigned to different nodes, the response time will be different. When the speculative tasks are running on different nodes, the efficiency will also be different. These

Received date: 31-05-2013
Corresponding author: LIN Wen-hui, E-mail: linwh16@gmail.com
DOI: 10.1016/S1005-8885(13)60112-0

uncertainties will greatly affect the performance of heterogeneous clusters and make the performance of heterogeneous clusters lower and less stable than homogeneous clusters. And this will cause difficulties for users to predict their job completion time.

In this work, we propose a novel approach for the heterogeneous clusters problems mentioned above. The novelty of our approach is twofold. First, we introduce a machine learning module into MapReduce framework. The module will study job historical information in the cluster, and calculate the data processing capacity of each node. Meanwhile it will learn the statistical information every day to calibrate nodes' performance values and acquire an accurate assessment of cluster performance. Second, after getting the performance measurement results in MapReduce framework, it will be used with other parameters (such as the number of data blocks, the location of data blocks and network performance, etc.) to optimize the reduce task assignment algorithm and speculative execution mechanism, to adapt to the characteristics of heterogeneous clusters and improve the performance and stability of the cloud computing cluster.

The highlights of our work can be summarized in the following points:

1) Developing a machine learning module. This module will make a detailed analysis of job historical information in cluster, to obtain the number of tasks running on each node, task running time, the size of data block and other statistical information. And then calculate the performance value of each node in the cloud computing cluster.

2) Optimizing the reduce task assignment algorithm. Having obtained the performance values of each node, we can choose the best node to run reduce task based on current job-related information, including data sizes of the job, the number of tasks completed by each node, network performance and other parameters.

3) Optimizing the speculative execution mechanism. When JobTracker wants to start a speculative task, it requires a comprehensive analysis and calculations of various kinds of statistical information. The statistical information includes the progress of current task, the remaining amount of data to be processed, network performance, the performance differences between nodes. Then JobTracker can select the appropriate node to run speculative tasks to avoid cluster resource waste.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the task assignment algorithm and speculative execution mechanism in MapReduce framework. Then we carry out a detailed analysis of the problems and bottlenecks encountered in current cloud computing cluster. In Sect. 3, we propose the MapReduce scheduling algorithm based on machine learning and describes our improvement work for MapReduce framework. Simulation and evaluation for above improved MapReduce algorithm is provided in Sect. 4. Finally, we make a conclusion to this paper and present future work.

2 Background and motivation

2.1 Overview of Hadoop

Hadoop has a master-slave architecture, which contains one master node and multiple slaver nodes. The master node contains NameNode and JobTracker modules, while each slaver node has DataNode and Tasktracker modules [4]. Fig. 1 shows the basic architecture of Hadoop. In master node, JobTracker is responsible for scheduling job, managing tasks, and communicating with TaskTracker. And TaskTracker is used to process tasks assigned by JobTracker in slave node. When the user submits a job, JobTracker will initialize the job and the job will be divided into several map tasks and reduce tasks. And then tasks are assigned to TaskTrackers by JobTracker [5].

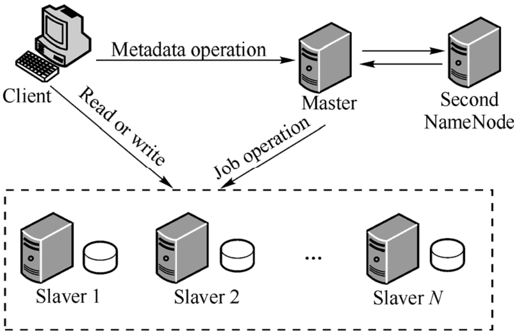


Fig. 1 The basic architecture of Hadoop

2.2 Task assignment algorithm in MapReduce

When the client submits a job, JobTracker first initializes the job, splits the job into multiple map tasks and reduce tasks, and then puts tasks into corresponding task queue for following assignment. At the same time, JobTracker will do a localized pre-assigned job for map tasks and mount map tasks to corresponding nodes according to the position of input data.

When Hadoop cluster is running, TaskTrackers periodically send heartbeat messages to JobTracker. The message is the statistical information of TaskTracker, including the running task information of this node, the node's disk usage, the node can receive new tasks or not, and so on. After JobTracker receives heartbeat messages from TaskTrackers, it will analyze the statistical information. If this slave node has idle ability to run tasks, JobTracker will choose task from the task queue and assign it to this node. Task assignment follows the process of map tasks first and reduce tasks second. A simple task assignment process is shown in Fig. 2.

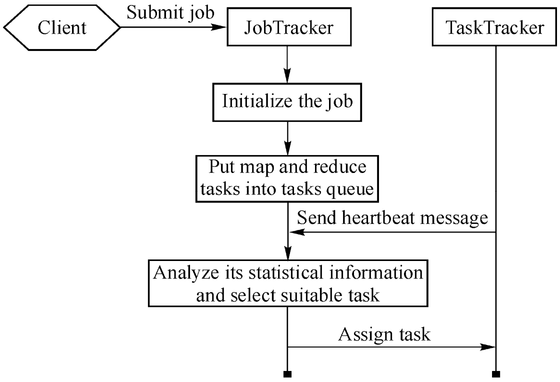

Client
Submit job
JobTracker
TaskTracker
Initialize the job
Put map and reduce tasks into tasks queue
Send heartbeat message
Analyze its statistical information and select suitable task
Assign task

Fig. 2 Simple task assignment process in Hadoop

In the process of task assignment, map tasks will be divided into data-local map tasks and non-localized map tasks. This is primarily determined by the location of the input data for map task and the location of the node which runs the task. If input data of the task and the TaskTracker are in the same server or the same rack, the map task is defined as a data-local map task, otherwise it should be a non-localized map task. Assignment process is described as follows:

Step 1 JobTracker first looks for a failed task which needs to be run again. If this kind of task exists in the task queue, JobTracker assigns it to the TaskTracker. Otherwise, JobTracker continues to look for a suitable task for allocation.

Step 2 JobTracker looks for a data-local map task from the non-running task queue. It first looks for the task whose input data is in the node. If it is not found, it will look for the task in the same rack. If there is such a task, JobTracker assigns the task to this TaskTracker. Otherwise, JobTracker continues to look for a non-localized map task.

Step 3 If Hadoop cluster allows speculative execution, JobTracker will first inquire map tasks which need to start speculative execution. If there is a map task whose job progress lags far behind the job progress, JobTracker will launch a speculative task for this task.

After map task assignment is completed, JobTracker will begin to assign reduce tasks. Reduce tasks are different from map tasks, and there is no localization concept for reduce tasks, since reduce tasks need to copy map tasks output result from each node. Therefore, the reduce task assignment process consists of two steps. First, JobTracker will find out whether there is a reduce task which is not running. If exists, it will be assigned to this node, otherwise JobTracker will check whether there is a Reduce task which needs to start speculative execution. If it exits, JobTracker will assign a speculative task to this node.

2.3 Speculative execution mechanism

High fault tolerance is an advantage of MapReduce. In cloud computing cluster, there are some overloaded nodes which lead to low task processing speed, and there are also some nodes failure and downtime. In this case, JobTracker will launch speculative tasks for these tasks. JobTracker will launch the same tasks in other nodes to avoid the situation that these tasks slow down the job running speed. The purpose of speculative tasks is to use resources to exchange for running speed by running the same tasks in multiple nodes. And JobTracker will use the output result of the node that completes the task fastest to improve the running speed of the job.

In MapReduce framework, JobTracker decides whether speculative execution should be started based on the progress of the task. For map tasks, the progress of a task is the ratio between the amount of the input data which has been processed and the total amount of the input data. For Reduce task, the calculation of task progress is relatively complex. First of all, Reduce task has three phases which are copy, sort and reduce. In copy phase, reduce task needs to copy the key-value intermediate data generated by map tasks from each node. Then the data is sorted so that the key-value pairs for a given key are contiguous in sort phase. Finally the actual reduce operation is to use the reduce function defined by user to process the intermediate data and output results. Therefore, each phase is defined as one-third of the progress of reduce task. JobTracker further needs to calculate the ratio between the amount of the input data which has been processed and the total amount of the input data in each phase to obtain the progress of reduce task.

We can use the following two formulas to calculate the progress of the map tasks and reduce tasks. $P_{\mathrm{M}i}$ denotes the map task process of node i in Hadoop cluster. And $P_{\mathrm{R}i}$ denotes the reduce task process. S represents the total amount of the input data, C is the amount of data processed. K stands for the phase of reduce task, including three values 1, 2, and 3. When K is 1, the reduce task is in copy phase. Value 2 denotes sort phase and 3 means the reduce task is in the last reduce phase. The definition of $P_{\mathrm{M}i}$ and $P_{\mathrm{R}i}$ are shown in below.

$$P_{\mathrm{M}i} = \frac{C}{S} \times 100\% \tag{1}$$

$$P_{\mathrm{R}i} = \frac{1}{3}\left(K - 1 + \frac{C}{s} \right) \times 100\%; \quad K = 1,2,3 \tag{2}$$

After the progress of task is defined, the condition for a task to start speculative task is simple. When a task has not launched speculative execution, and its progress is 20% (Hadoop default value) behind its job average progress in one minute, JobTracker will start a speculative execution for this task.

2.4 Related works and motivation

The heterogeneous environment for cloud computing is widespread in practical applications. For example, in IT companies or research institutes, hardware always needs to be upgraded, which will lead to the condition that the cluster consists of different performance servers and the homogenous environment evolves into heterogeneous environment. Therefore, the heterogeneous clusters performance improvement work is very important.

Our work is inspired by the early works of Zaharia et al. who proposes longest approximate time to end (LATE) scheduler in Hadoop. The LATE scheduler can improve MapReduce performance by predicting Hadoop job progress more accurately and take system overhead into account [2]. Works by Xie et al. [6] proposed a data storage mechanism which takes into account data locality to MapReduce performance of heterogeneous clusters. Polo et al. proposed an algorithm based on the adaptive scheduler which can provide dynamic resource allocation and minimize job completion time [7]. In Ref. [8], Fischer et al. introduced a mathematic model to evaluate the cost of task assignment and presents algorithms for the task assignment problem with costs that reflect data locality. Another scheduler algorithm was proposed by Zhang et al. [9]. The scheduler is used to determine whether Non-Local tasks could be assigned when JobTracker cannot find a data-local task. In Ref. [10], Sandholm et al. introduced a system to manage and dynamically assign the resources in a cloud computing cluster shared by multiple users. An improved data placement strategy was proposed by Lin. This strategy determines data blocks placement by evaluating the nodes' network distance and data load in Hadoop cluster. This strategy can reduce the time for data placement and improve the cluster performance [11]. Xue et al. [12] proposed a method to solve the problem of storing small files on HDFS. Their method aims at improving I/O performance of small meteorological files. In Ref. [13], Jiang et al. introduced five factors that affect the performance of Hadoop, and investigates alternative but known methods for each factor. They analyzed the changing of Hadoop performance through tuning these factors. A hybrid scheduler for scalable and heterogeneous Hadoop systems was proposed in Ref. [14], where they proposed a combination of the FIFO, fair sharing, and classification and optimization based scheduler for heterogeneous Hadoop (COSHH) schedulers. In Ref. [15], Guo et al. proposed a benefit aware speculative execution which can speed up task execution by using available resources more aggressively in heterogeneous network environments.

The MapReduce mechanism is suitable for homogenous cloud computing cluster. Above works improve the performance in heterogeneous clusters, but there is still much room for improvement.

In task assignment algorithm, the reduce task assignment is based on the order in which the node requests. The advantage of this approach is to assign reduce task to node in the easiest way, and it can reduce the pressure of JobTracker. However, the performance of nodes in heterogeneous clusters is different. When reduce task is assigned to the high performance node, the task will be completed faster, otherwise the task will run a long time. When the difference of each node's performance is big, task running time enjoys great uncertainty, which causes the instability of the time for completing the same job in a cloud computing cluster.

The speculative execution mechanism in MapReduce framework has certain irrationality. When speculative task is started, JobTracker does not consider data localization, node performance and other factors. JobTracker assumes a number of conditions, for example the performance of each node is basically the same, the time of dealing with the same types of tasks roughly equal in each node, speculative task is launched only because of the overloaded of current nodes and so on. In heterogeneous clusters, the performance of each node is quite different, the running of speculative tasks is prevalent, and the excessive speculative task execution leads to the decline of the cluster performance. This will make the job running speed lower than that when speculative execution mechanism is not started, which will result in resources waste and performance decline. Therefore, the speculative execution mechanism can enhance the performance in homogenous cloud computing cluster. However, in heterogeneous clusters this mechanism may not improve performance; on the contrary, it may even reduce the performance of the cluster.

3 MapReduce algorithm based on machine learning

This section describes our MapReduce algorithm based on machine learning. Firstly, we introduce a node performance indicator in our algorithm, and try to obtain the value of the indicator through machine learning approach. We design a machine learning module in MapReduce framework to help JobTracker acquire the computing capabilities of each node through analyzing job historical data in the cluster. And then, according to the load and performance of each node, we improve the reduce task assignment algorithm and speculative execution mechanism to improve the performance and stability of heterogeneous clusters.

3.1 Node performance measurement based on machine learning

Excellent Scalability is an advantage of cloud computing clusters. Nowadays, most of the cloud computing clusters are built in stages and gradually upgrade the hardware. Meanwhile, the hardware update speed is very fast and this will inevitably lead to node performance differences in cloud computing cluster. Therefore, the existing cloud computing clusters are mostly heterogeneous.

Currently, cloud computing cluster cannot accurately predict job completion time or reasonably assign tasks to nodes according to nodes’ performance. To solve this problem, we propose a scheduling algorithm based on machine learning. The algorithm combines the time series analysis and machine learning algorithms, which can solve cloud computing cluster job scheduling problem in heterogeneous environment.

In this paper，we designed an algorithm works on the master node. Through this machine learning module, JobTracker can assign tasks more suitable by analyzing the job processing statistics. The machine learning module collects information of map tasks and reduce tasks that each job was broken down. Then the module will statistically analyze map tasks historical information, calculate the number of tasks completed by each node and task completion time, and finally get the value of the performance of each node. Through the analysis of the relevant statistical information, we use node data processing speed to represent the performance of each node. Node data processing speed is the amount of data processed by one node per unit time. In heterogeneous clusters, the node data processing speed can more vividly demonstrate the performance differences of each node. Meanwhile, this module calculates job historical information every day to correct the performance value of each node. The detailed calculations are shown below.

We assume that current cluster has N nodes. Node data processing speed are v_1, v_2,…,v_N. The size of data block is C. As each map task corresponds to one data block in the cluster, the value of C also indicates the amount of input data for each map task. We define the vector $\boldsymbol{V} = (v_1, v_2,…,v_N)$, which is a collection of data processing speed of nodes. Node data processing speed is also the demonstration of node performance value.

Assuming that the number of running jobs is M_k in the day k. For the jth job, the average time for processing a task in node i is t_{ij}, and C/t_{ij} denotes the average speed of the node i processing tasks of job j. the matrix $\boldsymbol{A}_k$ shows the speed that each node in the cluster processes each job on the kth day.

$$\boldsymbol{A}_k = \begin{bmatrix} \frac{C}{t_{11}} & \cdots & \frac{C}{t_{1j}} & \cdots & \frac{C}{t_{1M}} \\ \vdots & & & & \vdots \\ \frac{C}{t_{i1}} & \cdots & \frac{C}{t_{ij}} & \cdots & \frac{C}{t_{iM}} \\ \vdots & & & & \vdots \\ \frac{C}{t_{N1}} & \cdots & \frac{C}{t_{Nj}} & \cdots & \frac{C}{t_{NM}} \end{bmatrix} \tag{3}$$

For cloud computing cluster running speed issues, we believe that it is similar to a lot of problems in the time series analysis. Cluster running speed changes every moment, and we can use some ideas from the time series to carry out modeling and analysis of the cluster running speed. The essential characteristics of time series analysis which different from the normal analysis is the neighboring correlation between observations. It is very similar to the data attributes in our paper. The speed of previous moment will greatly affect current calculation speed, thereby affecting the overall performance of the cloud computing cluster. Moving average method and exponential smoothing method are based method of time series analysis, which are suitable for simple system model. Compared with the moving average method, exponential smoothing method includes moving average processing idea and considers the debilitating impact of the past data. Combined with the running data in cloud computing cluster, we believe that the cluster running speed at each moment is the weighted average value of the observation value of a certain moment and its previous moment.

We define the initial data processing speed as V_0, as shown in Eq. (4). V_k is the result after k iterations.

$$\boldsymbol{V}_0^{\mathrm{T}} = \boldsymbol{A}_0 \frac{\boldsymbol{J}_{m1}}{M_0} \tag{4}$$

$$\boldsymbol{V}_k^{\mathrm{T}} = \boldsymbol{A}_{k-1}\boldsymbol{J}_{m1}\frac{1-\alpha}{M_{k-1}} + \alpha\boldsymbol{V}_{k-1}^{\mathrm{T}} \tag{5}$$

In Eq. (5), $\boldsymbol{J}_{m1} = (1,1,\dots,1)_{n\times 1}$, α is a weight parameter, which is used to represent the impact of the calculation result of $k-1$ on the calculation result of k. In this paper, we use k to define the number of days, that is, we calculate the value of the vector V once a day. In the actual operation of the process, the number of running jobs cannot be the same every day, so we randomly select M job historical information every day.

In our algorithm, select an appropriate value of α is a very important, which will directly affect the forecast results. As shown in Eq. (5), with the larger value of α, the measurement result is more reliance on historical data. Combined with the type of data to be processed in cloud computing cluster, we believe that the running speed of cloud computing cluster is irregular ups and downs, but its long-term trend is relatively stable. And refer to the relevant information of time series, its value between 0.05 and 0.2 is reasonable. After several experiments, we verify that 0.1 as the value of α is more reasonable in Hadoop experimental cluster. This means that the impact of previous day's data on current Hadoop cluster is small and the cluster performance is relatively stable over time.

Through adding machine learning module to the master node, JobTracker can analyze the job historical information every day. The machine learning module needs to know the number of map tasks and reduce tasks that each job is broken down. Then the module will statistically analyze map tasks history information, calculate the number of tasks completed by each node and task completion time, and finally get the value of the performance of each node. Meanwhile, this module will calculate job historical information every day to correct performance value of each node. The detailed calculations are shown below.

3.2 MapReduce optimization algorithm based on machine learning

In the previous section, we use machine learning approach to calculate the performance indicator of nodes. Using this indicator as a basis, we propose a MapReduce optimization algorithm. The algorithm is an innovative approach for heterogeneous clusters problem. Firstly, we optimize the reduce task assignment. When JobTracker needs to assign reduce tasks, it will be combined with node performance value, to-be-transferred data amount, network performance and so on, and then choose the best node to run reduce tasks. Then, we carry out targeted improvements for speculative execution mechanism of map tasks and reduce tasks. The advantage of our speculative execution mechanism is to improve the effectiveness of speculative task, reduce the waste of resources and speed up job running speed in cluster. Using this algorithm, the performance of heterogeneous clusters will be greatly improved and the stability of cluster performance will also be enhanced.

In heterogeneous clusters, the high performance node will run more map tasks and store more intermediate data. When these intermediate data are temporarily stored in the slave node which will not launch reduce task, the cluster will spend a lot of network resources for data transmission. And reduce task will complete more quickly when it runs on the high performance node. The default MapReduce mechanism does not consider node performance problems in heterogeneous clusters. Therefore, job completion time will enjoy greater fluctuation when it runs in heterogeneous clusters.

Under such circumstances, we optimize the reduce task assignment algorithm to improve the performance and stability of the cloud computing cluster. Firstly, JobTracker acquires the performance value of each node from the machine learning module. Then, with the heartbeat messages from TaskTrackers, JobTracker builds a list of node status. When JobTracker begins to assign reduce tasks, it will use the active form of assignment rather than wait for TaskTracker to initiate a request. Following the high-to-low sequence of node performance, JobTracker will inquire node information in the node status list and the load of the node, and then assign a reduce task to the slave node which has idle reduce tasks running ability and best performance value. JobTracker selects corresponding number of nodes to run tasks according to the number of Reduce tasks that need to be run. The flow chart is shown in Fig. 3.

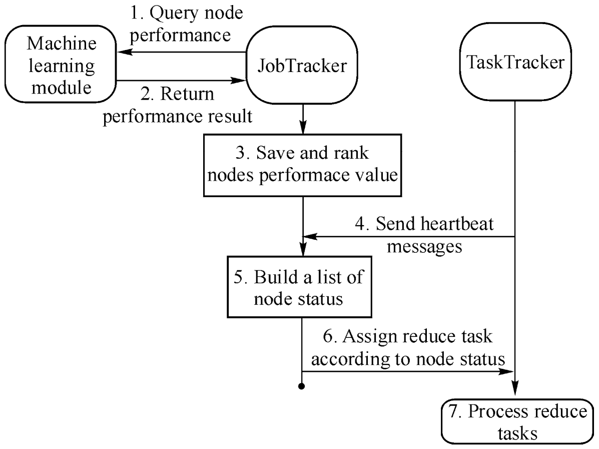

1. Query node performance
Machine learning module
JobTracker
TaskTracker
2. Return performance result
3. Save and rank nodes performace value
4. Send heartbeat messages
5. Build a list of node status
6. Assign reduce task according to node status
7. Process reduce tasks

Fig. 3 Flow chart of reduce task assignment

JobTracker will launch speculative task for the task that runs more slowly. However, JobTracker does not consider the performance of nodes when it assigns speculative tasks. In practical applications, the phenomenon that the running speed of speculative task is lower than that of the original task often appears in heterogeneous clusters. In this case, the speculative task is unable to expedite the completion of the job and there will be a waste of cluster resources. Therefore, in the heterogeneous clusters, the launch condition of speculative execution needs to be reasonably assessed. Meanwhile, we need to design a node selection algorithm for launching speculative task to improve the efficiency of the speculative task, avoid waste of resources and speed up the completion of the job.

In this algorithm, when JobTracker needs to start a speculative map task, it should firstly obtain performance value of the node that runs the map task and the locations of the map task input data. In these nodes that store the input data, if there is a higher performance node which has idle slot that can run the map task, JobTracker will choose the node that meets the conditions and has best performance to run the speculative task. Otherwise, JobTracker will continue the search from the list of all nodes. When the performance of the selected node is better than that of the original node and has idle slot to run the map task, JobTracker will launch a speculative task in this node. In other words, if the selected node satisfies the condition of Eq. (6), JobTracker will choose this node.

$$\frac{D}{v_j}+\frac{D}{v_{ij}}-\frac{D(1-P_{\mathrm{M}i})}{v_i}>0 \tag{6}$$

In Eq. (6), v_{ij} denotes inter-node network transmission speed in cluster, $P_{\mathrm{M}i}$ denotes the task progress in node i, and D is the input data size. v_i stands for the task processing speed of the node I and v_j represents the task processing speed of the node j. Since D is a common part, Eq. (6) can be simplified to Eq. (7).

$$\frac{1}{v_j}+\frac{1}{v_{ij}}-\frac{1-P_{\mathrm{M}i}}{v_i}>0 \tag{7}$$

We assume that S is the size of data that the reduce task needs to handle. The Reduce task progress in node i is $P_{\mathrm{R}i}$, v_i denotes the data processing speed of node i, and the remaining time of the reduce task is T_i which is shown in Eq. (8). When the reduce tasks are running, reduce task must copy the intermediate result generated by map tasks from each node. The time for copying data cannot be accurately calculated, so it is necessary to add a threshold value β. We assume that the selected node is j, its data processing speed is v_j, and the remaining time of reduce task is T_j. The definition of T_j is shown in Eq. (9). If T_i is larger than T_j, JobTracker will launch a speculative reduce task in node j.

$$T_i=S\frac{1-P}{v_i} \tag{8}$$

$$T_j=\frac{S}{v_j}(1+\beta) \tag{9}$$

When JobTracker needs to start a speculative reduce task, it should firstly obtain performance value of the node that runs the reduce task. Then JobTracker selects a node from the node list. This node should have a higher performance value and idle slot that can run the reduce task. If the node satisfies the conditions of Eq. (10), JobTracker will launch a speculative Reduce task in this node. Otherwise, JobTracker will not launch speculative execution. Since S is a common part, Eq. (10) can be simplified to Eq. (11). The pseudo-code of our algorithm is provided in Fig. 4

$$S\frac{1-P}{v_i}-\frac{S}{v_j}(1+\beta)>0 \tag{10}$$

$$v_j(1+\beta)>\frac{v_i}{1-P_{\mathrm{R}i}} \tag{11}$$

Pseudo-code interpretation of MapReduce optimization algorithm

Algorithm 1 Reduce task assignment algorithm

```
Input:
  $P_{\mathrm{node}}$: Node performance list sorted by node performance
  NodesList$_{\mathrm{s}}$: Node status list
  TaskList$_{\mathrm{reduce}}$: List of reduce tasks need to be assigned
Output:
  AssignFlagList: Flag list indicates whether reduce tasks are assigned.
For each reduce task in TaskList$_{\mathrm{reduce}}$ do
  $i = 0$;
  For each node in $P_{\mathrm{node}}$ do
    SelectedNode = Node which is sequentially read from P$_{\mathrm{node}}$
    NodeFlag = check whether SelectedNode has idle ability to run reduce task from NodesList$_{\mathrm{s}}$
    if ( NodeFlag == True) then
      Assign this task to SelectedNode
      AssignFlagList[$i$] = True
    end if
    if (AssignFlagList[$i$] == True ) then
      break;
    end if
  end for
  if(AssignFlagList[$i$] == True) then
    update the SelectedNode status in NodesList$_{\mathrm{s}}$
  end if
   i is incremented by one
end for
```

Algorithm 2 Speculative execution algorithm

```
Input:
  P$_{\mathrm{lownode}}$: performance value of the node $i$ that runs the task
  Percent$_{\mathrm{lownode}}$: the task progress in node $i$
  V$_{\mathrm{lownode}}$: the data processing speed of node $i$
  $S$: the size of data that the task needs to process
  P$_{\mathrm{node}}$: Node performance list sorted by node performance
  NodesList$_{\mathrm{s}}$: Node status list
  $\alpha$: threshold for data tranmission
Output:
  CheckFlag: Flag to indicate whether this node meets the condition
For each Node in $P_{\mathrm{node}}$ do
  if ( $P_i < P_{\mathrm{lownode}}$) then
    continue;
  end if
  NodeFlag = check whether SelectedNode has idle ability to run task from NodesLists
  if ( NodeFlag == False) then
    continue;
  end if
  $T_{\mathrm{lownode}} = S / V_{\mathrm{lownode}} * (1 - \mathrm{Percent}_{\mathrm{lownode}})$
  $T_i = S / V_{\mathrm{i}} * (1 + \alpha)$
  if ($T_i < T_{\mathrm{lownode}}$) then
    continue;
  else
    select this node to run speculative tasks
    ChenkFlag = True
    break;
  end if
end for
```

4 Experimfnt and analysis

In this section, in order to verify the effect of the MapReduce algorithm based on machine learning, we will introduce the improved algorithm into a Hadoop cluster. In this experiment, we add the machine learning module to the master node and edit the task assignment algorithm in Hadoop cluster. Then we will run the same jobs in the improved Hadoop cluster and the default Hadoop cluster. Finally, we analyze the experimental results.

4.1 Experimental setup

The experimental platform is built on a cluster with one master node and five slave nodes. And operating system in each node is Centos 5.4, Kernel version is 2.6.18, Hadoop version is 1.0.2, and Java version is 1.6.0_33.

The experimental cluster is a heterogeneous environment, the hardware configuration of the servers in the cluster are different. The master node is IBM System x3650 M2, CPU Intel (R) Xeon (R) E5520 2.27 GHz, 2×4 GB DDR2, 146 GB HDD. The slave nodes include three types of servers. Two nodes are IBM x236 which have been used for seven years. Hardware configuration is CPU Intel (R) Xeon (TM) 2.40 GHz, 1×4 GB DDR2, 146 GB HDD. Another two nodes are Lenovo Server.

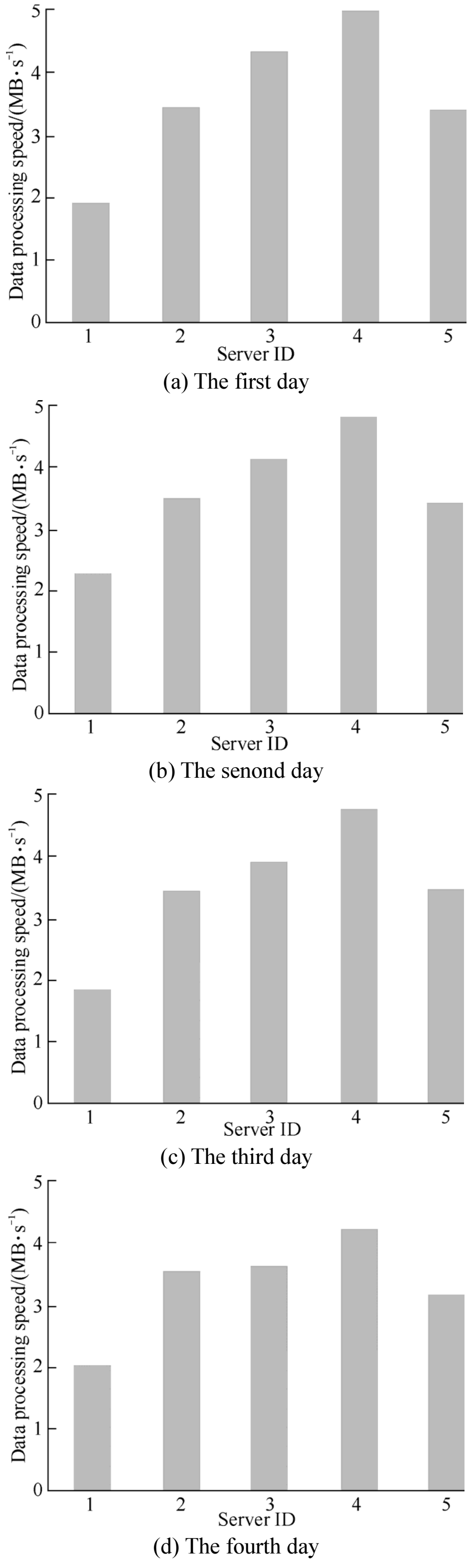
Hardware configuration is Pentium (R) Dual-Core 3.20 GHz, 2×2 GB DDR2, 500 GB HDD. And a slave node is IBM x3850, CPU Intel(R) Xeon (TM), 2.66 GHz, 1×4 GB DDR2, 2×146 GB HDD. Server configuration cannot quantify the level of server performance, so it is necessary to use machine learning method to accurately quantify the performance of nodes.

Terasort is a classic Hadoop benchmark. It uses map/reduce to sort the input data into a total order. TeraSort is a recognized standard which can be used to measure the cluster's data processing capabilities in the framework of distributed data processing. In this experiment, we use TeraGen to generate test data. TeraGen is a map/reduce program which can write out a specified amount of data. Each line of test data is 100 B, which contains a 10 B key and a 90 B value.

4.2 Node performance measurement based on machine learning

In this experiment, we run several jobs every day in the experimental Hadoop cluster. We use the machine learning module to analyze and calculate the daily job historical information, and obtain the performance values for each node. Firstly, we use the machine learning module to analyze the four-day job running data in the cluster. This module calculates job historical information every night. And it calculates the number of tasks completed by each node, task completion time, the input data sizes, and so on. After calculating the daily job historical information, the module can get the average time for processing tasks in each node and get the value of $\boldsymbol{A}_k$ in Eq. (3). Then using the first day's job information and $\boldsymbol{A}_k$ in Eq. (4), this module can get the value of V_0. In the subsequent three days, the machine learning module analyzes the job historical information every day and use the Eq. (5) to get the value of V_k. Fig. 5 is the four-day performance value of nodes which is calculated by the module. The horizontal axis indicates the slave nodes in cluster. The vertical axis represents the data processing speed, which indicates the size of data processed by a node per second, and unit is MB/s.

As shown in Fig. 5, we can see that the performance values of nodes are tending to stabilize, but there will be slight fluctuations. In the long term, with the aging of the server hardware or equipment replacement, the node's performance will have big changes.



(a) The first day

(b) The senond day

(c) The third day

(d) The fourth day

Fig. 5 Four-day performance value of nodes calculated by machine learning module

Therefore, in actual operation, we do not need to analyze the job historical information or calculate the node's performance value every day. We can recalculate the performance value of nodes weekly, monthly or when hardware changes in cluster to reduce the pressure of the master node. Meanwhile, as can be seen from Fig. 5, server hardware parameters and the use of time both affect node performance.

We use Eq. (5) for iterative calculation and finally get V_3 = (2.01, 3.55, 3.62, 4.24, 3.16). This value is the performance value of nodes in the cluster after four days running as well as the basis for task assignment in our algorithm. In the next section we will launch jobs based on the value of V_3 and launch the same jobs with Hadoop default mechanism for comparison to analyze the impact of the improved algorithm on Hadoop cluster performance.

4.3 Job running experiment based on improved MapReduce algorithm

In order to objectively describe the impact of the improved MapReduce algorithm, we use Terasort as a benchmark. Firstly, we use TeraGen program to generate standard test data. Then we run Terasort job with different sizes of input data. We use the improved MapReduce algorithm and default algorithm to launch the same jobs in the Hadoop cluster. Experimental results will be compared and analyzed.

The test data size is from 10 GB to 60 GB in this experiment. Fig. 6 is a comparison chart of job completion time. From this figure, we can see that the improved algorithm can improve the running speed of the job to a certain extent. Our work mainly consists of two parts. First, according to the performance and load of nodes, JobTracker can select suitable nodes to launch reduce tasks to improve job performance and job running speed stability. Second, we improve the launch conditions and nodes selection algorithm of the speculative task.

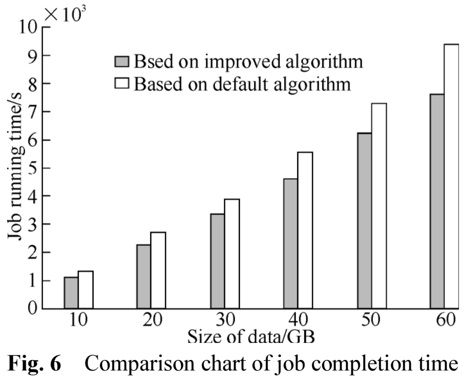


Fig. 6 Comparison chart of job completion time

This algorithm avoids the problem of the default MapReduce mechanisms which may start too many speculative tasks and invalid speculative tasks. And it can save resources and speed up the completion of jobs in cloud computing cluster. In this experiment, the cluster performance is improved about 19%.

After analyzing job historical information, we can get statistical information of Reduce tasks including its location and running time of all stages. Fig. 7 is the running time comparison chart of Reduce tasks with different data sizes in each node, horizontal axis represents five slave nodes in the experimental cluster, vertical axis represents the average time of each reduce task processed by slave nodes. In the figure we can see that there is a large time gap between different nodes when processing reduce tasks. When a Reduce task runs in the higher performance node, its running time will be significantly reduced. This also shows that our improved algorithm mainly increases the running speed of reduce tasks. In the default Hadoop cluster, reduce tasks randomly run on the node , and the improved algorithm running reduce tasks node is in accordance with the performance and load of the node, which to some extent, improve the Hadoop performance and stability. Reduce tasks randomly run on the node in default hadoop cluster. But with the improved algorithm, JobTracker will select a suitable node to launch a reduce task in accordance with the performance and load of the node. So this algorithm can improve the performance and stability of cloud computing cluster.

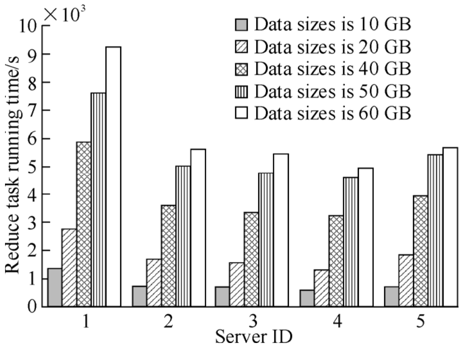


Fig. 7 Running time comparison chart of reduce tasks with different data size in each node

Fig. 8 shows the time of the data copy phase when the reduce task is running with different input data sizes. As the comparative experiment uses the same cluster, the network configuration is the same, so this figure also indicates the size of the amount of data that each node needs to copy during reduce task running. When the node which will run a reduce task saves more intermediate result，it needs to copy less data. After the analysis of the job historical information, the time consumed by the copy stage occupies the entire reduce task running time from 64% to 52%. When the input data is 10 GB, this ratio is 60% in node 4 which has higher performance and 64% in node 1 which has lower performance. When the input data is 60 GB, this ratio is 52% in node 4 which has higher performance and 60% in node 1 which has lower performance. Meanwhile, when reduce tasks run on a higher performance node, the time for processing data will be less. Therefore, reducing the amount of data copied and selecting high performance node to launch reduce task are two key factors that influence the cluster performance in this algorithm.

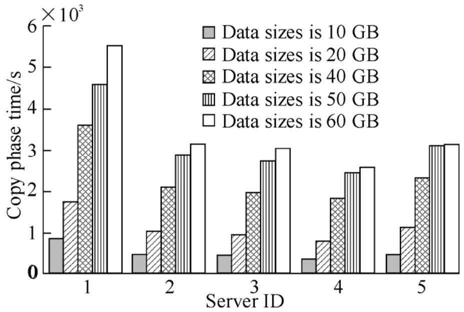


Fig. 8 Running time comparison chart of copy phase with different data sizes

Based on above analysis, we know factors that affect the cluster performance including task assignment algorithm, map task processing speed, network performance, data sizes in copy phase, jobs running parameters and so on. During job running, the higher performance node will run more map tasks to store more map output results, and reduce task need to obtain the data generated by map tasks from each node. Therefore, the data transmission time will be different when reduce tasks run on different nodes. Meanwhile, when a reduce task runs on a lower performance node, it will take more time to complete task. Therefore, this optimization of MapReduce mechanism is the key to improve the performance of cloud computing clusters in a heterogeneous environment. When we optimize the performance of MapReduce mechanism, we need comprehensive consideration of various factors, including job scheduling, task assignment, cluster parameter configuration, and so on.

5 Conclusions and future work

In this paper, we propose a MapReduce algorithm based on machine learning for solving heterogeneous clusters problem. Compared with existing efforts, our approach has significantly different philosophy. The novelty of our approach lies in two key features: first, a machine learning module is introduced into MapReduce framework. This module is used to study job historical information and calculate the data processing capacity of each node in cluster. Second, based on the learning result, two aspects of optimization have been done:

1) Reduce task assignment algorithm. The improved task assignment algorithm will assign reduce tasks based on node performance to improve the job running speed.

2) Speculative execution mechanism. The new speculative execution mechanism will fully consider the performance and load of slave nodes before launching speculative execution in suitable nodes. This mechanism can avoid launching invalid speculative execution that results in cluster resources waste.

Finally, our results show that in current experimental environment, the cluster performance is improved about 19%.

The future directions of our research is performance optimization for cloud computing. The cloud computing architecture is complex, it involves various aspects of the storage, parallel computing, network architecture, and so on. Therefore, there is a great need for comprehensive consideration of performance optimization and reasonable optimization of different cloud computing scenarios.

Acknowledgements

This work was supported by the Important National Science & Technology Specific Projects (2012ZX03002008), the 111 Project of China (B08004), and the Fundamental Research Funds for the Central Universities (2012RC0121).

References

1. White T. Hadoop: the definitive guide. 3rd ed. Beijing, China: O'Reilly Media Inc, 2012
2. Zaharia M, Konwinski A, Joseph A D, et al. Improving mapreduce performance in heterogeneous environments. Proceedings of the 8th USENIX Symposium on Operation Systems Design and Implementation (OSDI'08), Dec 8-10, 2008, San Diego, CA, USA. Berkeley, CA, USA: USENIX Association, 2008: 29–42
3. Borthakur D. The Hadoop distributed file system: architecture and design. http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf. 2007
4. Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system. Proceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST'10), May 3–7, 2010, Lake Tahoe, NV, USA. Los Alamitos, CA, USA: IEEE Computer Society, 2010: 10p
5. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107–113
6. Xie J, Yin S, Ruan X, et al. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Ph, d. Forum (IPDPSW−10), Apr 19–23, 2010, Atlanta, GA, USA. Los Alamitos, CA, USA: IEEE Computer Society, 2010: 9p

To p. 121

Communications in Nonlinear Science and Numerical Simulation, 2012, 17(12): 4544–4550

6. Zhou M, Ma J X. The influence of fiber dispersion on the transmission performance of a quadruple-frequency optical millimeter wave with two signal modulation formats. Optical Switching and Networking, 2012, 9(4): 343–350
7. Purwins H G, Bodeker H U, Amiranashvili S. Dissipative solitons. Advances in Physics, 2010, 59(5): 485–701
8. Zhang H, Tang D Y, Zhao L M. Dissipative vector solitons in a dispersion-managed cavity fiber laser with net positive cavity dispersion. Optics Express, 2009, 17(2): 455–460
9. Ouyang C M, Chai L, Hu M L, et al. Characteristics of three operation schemes in a passively mode-locked all fiber ring laser. Proceedings of the International Conference on High-Power Lasers and Applications IV: Proceedings of the SPIE, Vol 6823, Nov 12–14, 2007, Beijing, China. Bellingham, WA, USA: SPIE, 2007: 179–184
10. Francisco J D, Pedro C P. Propagation properties of strongly dispersion-managed soliton trains. Optics Communications, 2012, 285(2): 162–170
11. Cho S B, Song H, Gee S Y, et al. Pulse width and peak power optimization in a mode-locked fiber laser with a semiconductor saturable absorber mirror. Microwave and Optical Technology Letters, 2012, 54(10): 2256–2261
12. Karlsson S, Yu J, Akay M. Time-frequency analysis of myoelectric signals during dynamic contractions: A comparative study. IEEE Transactions on Biomedical Engineering, 2000, 47(2): 228–238

(Editor: ZHANG Ying)

From p. 87

7. Polo J, Carrera D, Becerra Y, et al. Performance management of accelerated mapreduce workloads in heterogeneous clusters. Proceedings of the 39th International Conference on Parallel Processing(ICPP'10), Sep 13–16, 2010, San Diego, CA, USA. Los Alamitos, CA, USA:IEEE Computer Society, 2010: 653–662
8. Fischer M J, Su X, Yin Y. Assigning tasks for efficiency in hadoop: Extended abstract. Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'10), Jun 13–15, 2010, Santorini, Greece. New York, NY, USA: ACM, 2010: 30–39
9. Zhang X, Feng Y, Feng S, et al. An effective data locality aware task scheduling method for mapreduce framework in heterogeneous environments. Proceedings of the 2011 International Conference on Cloud and Service Computing (CSC'11), Dec 12–14, 2011, Hong Kong, China. Los Alamitos, CA, USA: IEEE Computer Society, 2011: 235–242
10. Sandholm T, Lai K. Mapreduce optimization using regulated dynamic prioritization. Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/ Performance'09), Jun 15–19, 2009, Seattle, WA, USA. New York, NY, USA: ACM, 2009: 299–310
11. Lin W. An improved data placement strategy for hadoop. Journal of South China University of Technology: Natural Science, 2012, 40(1): 152–158 (in Chinese)
12. Xue S J, Pan W B, Fang W. A novel approach in improving I/O performance of small meteorological files on hdfs. Applied Mechanics and Materials, 2012, 117/118/119: 1759–1765
13. Jiang D, Ooi B C, Shi L, et al. The performance of mapreduce: an in-depth study. Proceedings of the VLDB Endowment, 2010, 3(1/2): 472–483
14. Rasooli A, Down D G. A hybrid scheduling approach for scalable heterogeneous hadoop systems. Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (SCC'12), Nov 10–16, 2012, Salt Lake City, UT, USA. Los Alamitos, CA, USA: IEEE Computer Society, 2012: 1284–1291
15. Guo Z, Fox G. Improving mapreduce performance in heterogeneous network environments and resource utilization. Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12), May 13–16, 2012, Ottawa, Canada. Los Alamitos, CA, USA: IEEE Computer Society, 2012: 714–71

(Editor: WANG Xu-ying)