Research on Job Scheduling Algorithm in Hadoop

Yang XIA†, Lei WANG1, Qiang ZHAO1, Gongxuan ZHANG2

1 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
2 School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract

On the basis of researching Fair Scheduling Strategy deeply in Hadoop cluster, the Node Health Degree is defined by constructing the relationship function between node load and job fail rate, and a job scheduling algorithm based on Node Health Degree is proposed in this paper. Nodes are grouped, according to Node Health Degree, into three categories in order to assign corresponding job in accordance with load and guarantee resource load balance. By comparing with FIFO and Fair scheduling algorithm, the simulation results show that this algorithm can ensure to reduce job fail rate and improve cluster throughput.

Keywords: Hadoop; MapReduce; Node Health Degree; Fair Scheduling Strategy

1. Introduction

Hadoop[1] is a large distributed software framework[2] and an open source project realizing core mechanism of Google’s GFS[3] and MapReduce[4], most of cloud computing platform[5] use MapReduce parallel programming model of Hadoop to develop programs.

Job scheduling algorithm is one of the core technologies of Hadoop platform, its main function is to control the order of job execution and assign user’s job to run up on resources. Not only many types of application on Hadoop platform shared among multiple users are growing more, but also mixture of batch long jobs and interactive short ones which access same data set has become a typical way of job running. Therefore, in homogeneous environment, the main purpose of multi-user scheduling algorithm is to achieve the balance between efficiency of Hadoop cluster and efficiency of resource allocation among jobs. Many scheduling strategies for MapReduce cluster have been put out, for example, FIFO scheduler[6], HOD scheduler[7], LATE scheduler[8], Capacity scheduler[9] and Fair scheduler[10]. Every scheduling algorithm has its definite applicability, but also has its limitations, for example, balance of resource allocation between long jobs and short jobs is not taken into account in FIFO scheduler, but Fair scheduler balances job resource allocation by setting minimum shares and fair shares between jobs and job pools.

Job scheduling algorithm based on Node Health Degree (named FS-HD) is put forward in this paper. The algorithm improves Fair scheduler strategy and defines the Node Health Degree according to the relationship between node load and task fail rate, in which jobs with corresponding load demand are allocated to nodes in the cluster. It overcomes the problems in Fair scheduler strategy by reducing the fail rate, increasing throughput and improving resource load balance.

† Corresponding author.
Email address: xia-y@163.com (Yang XIA).
2. Fair Scheduler Strategy

Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average, an equal share of resources over time. If there is a single job running, the job uses the entire cluster. When other jobs are submitted, free task slots are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Unlike the Hadoop’s default scheduler, which forms a queue of jobs, it lets short jobs complete within a reasonable time while not starving long jobs.

The scheduler actually organizes jobs by resource pool, and shares resources fairly between these pools. By default, there is a separate pool for each user. There are the limits of concurrently running Map and Reduce tasks on “TaskTracker” of node. The fair scheduler can limit the number of concurrently running jobs from each user and from each pool. Running job limits are implemented by marking jobs as not runnable if there are too many jobs submitted by the same user or pool. When there are idle task slots, the scheduler processes in two steps, at first idle task slots are allocated between the job pools and secondly are allocated between jobs in the job pools.

2.1. Minimum Shares and Fair Shares

Minimum Shares are set by the users and can always ensure the job to get enough resources. Pools can be given weights to achieve unequal sharing of the cluster. The computation of Fair Shares of job pools is to assign total cluster resources to all running job pools according to weight of job pools. In the case of default, the weight of job pool is based on the priority and jobs are ordered first by priority. The priority increases one level with the corresponding doubling weight. Usually minimum shares are less than fair shares in a job pool.

2.2. Task Slot Scheduling Algorithm between Pools

Fair scheduling algorithm between pools is described as follows:

1) If there are job pools not reaching Minimum Shares, then \( c \) is the sum of the task slots assigned to a pool, \( m \) is the Minimum Shares, task slots will be allocated to the job pool with the minimum value of \( c/m \) firstly;

2) if each pool has gotten the Minimum Shares, \( w \) is the weight of job pool, then task slots will be allocated to the job pool with the minimum value of \( c/w \) firstly;

3) Fair scheduler maintains two variables for each job pool, the time limit of Minimum Shares \( T_{\text{min}} \) and the time limit Fair Shares \( T_{\text{fair}} \). When a job pool does not reach Minimum Shares in \( T_{\text{min}} \) or does not reach half of the Fair Shares in \( T_{\text{fair}} \), the scheduler will kill the most recently launched tasks from overscheduled pools, to minimize the amount of computation wasted by preemption.

There are limits to the number of the concurrently running jobs in each job pool. If there are too many jobs submitted, the follow-up jobs will wait in the scheduling queue until previous jobs complete and release task slots.

2.3. Task Slot Scheduling Algorithm in Pools

Because of the poor data locality, fair schedule strategy also provides a Delay Scheduling Algorithm between jobs in pool to improve data locality.

Within a pool, jobs queue according to Fair Scheduler or FIFO. When there are idle task slots, if the head-of-line job cannot launch a local task on the TaskTracker, then it is skipped, and other running jobs are looked at in order of pool shares and in-pool scheduling rules to find a job with a local task. However, if the head-of-line job has been skipped for a sufficiently long time, it is allowed to launch rack-local tasks.

But there are still some problems. In Hadoop cluster, load of nodes depends on the resource cost of the running tasks rather than the number of tasks. Fair scheduler does not consider the case, so the load of
nodes is not balanced. In addition, the cluster running Hadoop platform is shared by many systems, such as MPI, MemCache, ..., so for each node the stability of running jobs will inevitably be affected, which makes task error rate on nodes higher.

3. Job Scheduling Algorithm based on Node Health Degree

This algorithm improves the fair schedule strategy. Fair scheduler is still used between job pools, and an algorithm based on Node Health Degree is adopted between jobs of pool. It assigns corresponding tasks to nodes according to the Node Health Degree.

3.1. Node Load Model

Homogeneous and multi-users of networks and correct jobs are premises of the scheduling strategy. We define node set on cluster as $R=\{r_1 \ldots r_j \ldots r_n\}$ ($r_j$ denotes the $j$ nodes, there are $n$ nodes on the cluster) and set of load weight of $n$ nodes as $L=\{l_1 \ldots l_j \ldots l_n\}$ ($l_j$ is dynamic load weight of node $r_j$). If CPU usage, memory usage and I/O access rate of node $r_j$ separately is $C\%$, $M\%$, $I\%$, then the dynamic load weight of node $r_j$ can be expressed as:

$$ l_j = \lambda_1 \times C_j \% + \lambda_2 \times M_j \% + \lambda_3 \times I_j \% $$

In this expression, $\sum_{n=1}^{3} \lambda_n = 1$, weights $\lambda_n$ reflect the importance of the various load parameters, $j=1, 2\ldots m$.

3.2. Characteristic Variables

The characteristic variables of task describe resource usage and demand for MapReduce task. The values of these variables can be obtained by analyzing historical information of job and stored in Job Tracker. The characteristic variables used in this paper are the average CPU utilization $c_{usage}$, the average I/O utilization $I_{usage}$ and average memory usage $M_{usage}$ of task. If the values of these characteristic variables of MapReduce task of a job are small, the job is a light one, otherwise is a relative heavy job.

The characteristic variables of node represent the status and quality of computing resources on a node in a period of time and the amount of resources supplied to the task. Node’s characteristic variables used in this paper are the average CPU idle rate $C_{idle}$, the average I/O idle rate $I_{idle}$ and the average free memory $M_{idle}$. These variables on nodes are computed periodically and sent to Job Tracker from heartbeat information of Task Tracker.

3.3. Node Health Degree

Fair scheduler chooses these tasks whose progress is less than the average 20% as backward tasks and launches backup tasks for them. If there are too many backward tasks and error tasks on a node, the load of the node may be too high or the running environment of the node may be caught by problems.

Many Delay Scheduler experiments indicate that average fail rate of node is directly proportional to node load. The dynamic load weight of node $l$ indicates independent variable and the average error rate of node $E=\frac{\alpha}{\beta}$ represents dependent variable, by computing large amounts of data generated in experimental and using curve fitting method, approximate function expression is found:

$$ E(l) = -0.0126 \times e^{(l-0.387)^2/0.0384} + 0.0629 \times l $$

(2)
If $E_j$ is the average fail rate of node $j$ actually received in the improved algorithm experiment, then the Node Health Degree is defined as $E_j / f(l_j)$. Nodes can be classified according to Health Degree as table 1 below:

<table>
<thead>
<tr>
<th>Health degree</th>
<th>$E_j / f(l_j)$ range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health node</td>
<td>$E_j / f(l_j) \leq 1.1$</td>
</tr>
<tr>
<td>Subhealthy</td>
<td>$1.1 &lt; E_j / f(l_j) \leq 1.5$</td>
</tr>
<tr>
<td>Unhealthy</td>
<td>$1.5 &lt; E_j / f(l_j)$</td>
</tr>
</tbody>
</table>

3.4. Algorithm Description

Between job pools this algorithm (FS-HD) uses Fair scheduling algorithm, a better strategy which has been proven. In job pool the job scheduling algorithm based on Health Degree of node is described as follows:

1) When an idle task slot can be used, a pool is chosen according to the Fair scheduler;
2) if the node on which the idle task is located is healthy, then go to 5;
3) if the node on which the idle task is located is sub healthy, characteristic variables of the node are separately multiplied by the reciprocal of Health Degree of node , then go to 5;
4) if the node on which the idle task is located is unhealthy, then go to 8, and set timer on node, its Health Degree is multiplied by 0.9 every x minutes until it reaches the state of sub healthy;
5) Computing Euclidean distances between characteristic variables of job task and characteristic variables of node, sorting them from small to large;
6) Choosing an appropriate job according to Delay Scheduler in the sorted set;
7) Assigning the task of a chosen job to a task slot to run;
8) Finish.

In step 4, generally the timer interval of unhealthy node is set to 5, when the node Health Degree reaches sub health, the node can be schedule again. In step 5, the computational method of Euclidean distance is

$$D = \sqrt{(C_{idle} - C_{usage})^2 + (I_{idle} - I_{usage})^2 + \frac{(M_{idle} - M_{usage})^2}{M_{idle}}}$$

(3)

In step 6, Delay Scheduler is used so that the algorithm achieves higher locality.
4. Experiment and Assessment

4.1. Experimental Environment

The experimental environment is deployed on a private cluster composed of 16 physical machines in two racks. The version of Hadoop is 0.20.2. Name Node and Job Tracker are installed on one machine and Data Node and Task Tracker are installed on others. These nodes are connected via Gigabit Switch. Hardware configuration is: Cpu: Intel Dual Core, 2.8GHZ, Memory: 2G, disk: 250G.

Some important parameters of Hadoop are set as below:

| Table 2 Hadoop Parameters |
|---------------------------|------------------|------------------|
| Hadoop Parameter          | Value            | Describe         |
| Replication               | 3                | Replication amount of each HDFS data block |
| HDFS Block size           | 16m              | Size of data block |
| Map tasks maximum         | 8                | Maximum number of concurrently running Map task |
| Reduce tasks maximum      | 8                | Maximum number of on currently running Reduce task |
| Heartbeat interval        | 3seconds         | Time interval of sending heartbeat information |

Three job pools based on multi-user are set in the experiment. 25 jobs are submitted to the pool every minute and the experiment continues 40 minutes. The job sets including cpu-intensive, I/O-intensive, memory-intensive and mixed type of job are randomly selected and submitted. FIFO and Fair Scheduler (FS) are used as comparison algorithm. The data used in analysis are the average data of ten experiments.

4.2. Experiment of Average Fail Rate of Job

In the experiment, the average fail rate of job is defined as

$$\text{Fail} = \frac{\lambda_1 \times \text{timeout} + \lambda_2 \times \text{error}}{\text{sum}}$$

Where: \(\text{sum}\) is the total number of Map/Reduce tasks which are submitted to all nodes within \(m\) minutes, \(\text{timeout}\) is the sum of timeout tasks, \(\text{error}\) is the sum of tasks which fail and are canceled, \(\lambda_1\) is the weight of timeout task, \(\lambda_2\) is the weight of error task. In the experiment \(\lambda_1 = 0.5, \lambda_2 = 1, \ m = 5.\) The experiment results are shown in figure 2. Before the average load rate of node reaches 40%, the average fail rate of job differs little with the three algorithms, however, after the average load rate of node exceeds 40%, the average fail rate of job which uses FS-HD is significantly lower than the other algorithms. Because FS-HD assigns a job with corresponding load demand to node according to the Node Health Degree, the average fail rate of job is significantly reduced.

4.3. Job Throughput Experiment

The job throughput is the average number of finished jobs per minute. Because of the lower average fail rate of job used FS-HD, Hadoop does not have to frequently launch backup tasks for timeout or error tasks, so redundancy is reduced and resource usage and job throughput are improved. In summary, FS-HD is better than Fair Schedule Strategy. Since jobs run in serial manner in a pool, the job throughput used FIFO is also poor. The experiment results are shown in figure 3.
4.4. Resource Load Balancing Experiment

Resource load balancing experiment is defined as the variance of node’s average load in unit time. According to node load, FS-HD selects the jobs with corresponding load demand to assign, so the nodes in cluster have better resource load balancing, and FS-HD is superior to FIFO and fair schedule strategy. The experiment results are shown in figure 4.

5. Conclusion

Hadoop is a popular open source implementation platform of MapReduce model and used to process and analyze large-scale data sets in parallel. FIFO, the default scheduler of Hadoop, Fair scheduler and Delay
scheduler are researched and the relationship between node load and job fail rate in Hadoop cluster is explored in MapReduce parallel programming environment. Further, the node Health Degree is defined and the job scheduling algorithm based on it is introduced in this paper. This algorithm is the improvement for Fair schedule strategy to improve resource load balancing and job throughput and reduce the average fail rate of job. And the effectiveness of the algorithm is verified by experiments.

References