Improvement of Local Scheduling Algorithm Policy in Hadoop Cluster Environment

Yue-feng WANG and Xi-bo WANG
School of Information Science and Engineering, Shenyang University of Technology, Shenyang110870, China

Keywords: Data locality, Preemptive, Average completion time for the job.

Abstract. The nature of the scheduling strategy of the local scheduling algorithm is to improve the data locality. However, due to the different completion time of the Map task, the waiting phenomenon of Reduce task affects the completion average time of the job, the completion time of the job is increased, and then the performance parameters of the system are not good. In this thesis, we propose to integrate the preemptive scheduling based on the local requirement of the original algorithm. Based on the above scheduling strategy, this thesis designs the qualitative scheduling of integrated preemptive strategy. In order to validate the improved algorithm, the local scheduling algorithm and the integrated preemptive local scheduling algorithm are compared by experiments. Experimental results show that, on the same data, the average completion time of the integrated preemptive local scheduling algorithm is significantly reduced.

Introduction

Hadoop platform is developed by the Apache Open Source Cloud Computing platform, which is mainly composed of Hadoop proposed Hadoop distributed file system HDFS and distributed computing framework MapReduce computing architecture. There are three original schedulers in Hadoop MapReduce: 1) Default scheduler FIFO Scheduler (FIFO scheduling). 2) Computing power (Capacity Scheduler) scheduler. 3) Fair scheduler (Fair Scheduler).

The MapReduce job scheduling algorithm has a very important influence on the performance of the cluster. The performance of the Hadoop system mainly through the following five indicators: (1) average completion time (the average completion time of all jobs); (2) fairness (refers to the fairness of scheduling algorithm for resource allocation operations); (3) data locality (refers to the tasks performed in the node with the required data on the proportion); (4) scheduling time (the scheduling overhead); (5) scheduling algorithms can meet the minimum requirements of user quota resources. However, these performance indicators are contradictory, that is, to improve the performance of certain performance indicators are bound to other aspects of the performance of the reduction of cost. Usually, scheduling algorithm is to improving one or several performance indicators. In general, the average completion time of the job is the performance index of each scheduling algorithm.

Integrated Preemptive Local Scheduling Algorithm

For the local scheduling algorithm, the priority is to emphasize the locality of data, but also to meet the locality of the job scheduling algorithm also consider the most important indicators, that is, the average completion time. In the above, we mentioned LARTS local scheduling algorithm, which enables the Early Shuffle mechanism, so that the Reduce task in the Map task has not yet been completed when the completion of Copy Map task generated intermediate results. At the end of a copy of the Map task in the middle of the value, it is possible that the next Map task has not yet completed, which will lead Reduce tasks to take up system resources while waiting for the Map task. This wastes the resources of the system and increases the average completion time of the job. The execution wait relationship between the Reduce task and the Map task is shown in Figure 1.
Reduce task idle waiting state leads to the lower utilization of platform resources, thereby reducing the efficiency of job execution. In order to solve the problem of low resource utilization and high average completion time, the thesis integrates the preemptive job scheduling method based on the original algorithm. In the Reduce task is idle state, through to seize its share of the computing resources and computing resources to be allocated to optimize the scheduling of Map tasks, so as to improve the operating efficiency, reduce the average completion time of operation.

The problem of Reduce task idle waiting state preemption needs to solve two key issues: the opportunity of seizing Reduce task computing resources and the resume execution time of Reduce task.

**The Preemption Time Reduce Task Resource**

Reduce task resources to determine the timing of preemption, need to understand the basic process of MapReduce computing framework. Through the process we can see that the idle time of the Reduce task is between the Reduce task pull intermediate value and the next Map task. So it is necessary to determine the Reduce task resources to seize the opportunity to be calculated from these two directions.

**Map Task Execution Time**

Map task execution time is the length of the time from the current time to the end of the Map task execution. Map task can be sorted out the remaining time calculation, such as formula (1)

\[ T_i = \frac{1-P}{P} \times T \]  

(1)

In the formula (1), Ti is the time remaining to perform the Map task, T refers to the execution time of the Map task. P refers to the execution of the Map task, which is obtained by using the size of the completed task divided by the total task size.

**Reduce Task Copy Remaining Time**

The Reduce task copy time is the length of the time from the current time to the intermediate value of the Map task is copied to the Reduce task buffer. Such as formula (2).

\[ T_i = (T_m + nT_b)(1-P) \]  

(2)

In the formula (2), Ti is the copy time of the Reduce task, and Tm is the time to copy the intermediate value of the Map task. Tb is the transmission time between different racks. P refers to the Reduce task copy progress.

The Reduce task was to seize the opportunity of resources need to be estimated according to the estimated residual copy processing time to estimate the residual value of the execution time of Map task and Reduce task value, the estimated results by judging whether to suspend Reduce tasks, such as formula (3).
\[ \text{Min}(T_{\text{map}1}, T_{\text{map}2}, \ldots, T_{\text{map}n}) - \text{Max}(T_{\text{reduce}1}, T_{\text{reduce}2}, \ldots, T_{\text{reduce}n}) \geq D \] (3)

The remaining execution time of executing Map tasks and Reduce tasks corresponding to the estimated value of the remaining copy processing time estimate input parameters as evaluation of Reduce task is pending, the parameter values by the ApplicationMaster module is responsible for collecting. The ApplicationMaster component receives parameters periodically after Reduce to hang through two steps to determine module is pending Reduce tasks: 1) the difference between calculated minimum and maximum residual Map task execution time Reduce task copy time; 2) compare the difference and relationship between the threshold to determine whether to seize. If the difference is greater than the set threshold, the state of the Reduce task is set to wait pending status in the TaskManager component. As shown in Figure 2.

![Figure 2. Task execution time chart.](image)

Reduce task rescheduling timing

The Reduce task is preempted to release the occupied resources, and the resources are assigned to the Map task. But this time the Reduce task is still in a suspended state, when the end of the suspended state is the need to study the content. When part of the Map task is completed, the Reduce task needs to pull the intermediate value from the newly created Map task, which is the trigger condition for the Reduce task to resume execution from the pending state. The Reduce task rescheduling model is based on the number of Map tasks that are newly generated after the Reduce task is suspended, and whether it is necessary to restore the Reduce task from the suspend state, so as to continue the execution. Scheduling time increased by Reduce in order to avoid frequent tasks suspend execution and recovery operation, when the Reduce task hangs after the end of the Map task execution number reaches a set threshold, the Reduce task from the pending resumption of execution. Such as formula (4).

\[ \frac{N_s - N_f}{N_s} \geq D \] (4)

Ns represents the number of Map tasks that have been completed, and Nf indicates the number of intermediate values that have been successfully copied to the Reduce buffer. The Reduce scheduling module through the two steps to determine whether the recovery of Reduce tasks: 1) the ratio of the number of Map in the process of calculation is done to seize the share; 2) the calculation results and the threshold comparison, determine whether the rescheduling. If the difference is greater than the set threshold, the state of the Reduce task is set to run in the TaskManager component.

Experimental Results and Performance Analysis

In this paper, a virtual machine is used to build heterogeneous test environment. Define two racks, each with a rack of 5 virtual machines, each of which is assigned 1GB memory. Test job is WordCount. The average waiting time of the Reduce task is tested by the local job scheduling and the original algorithm. The results are shown in Figure 3, the integrated scheduling algorithm for preemptive task Reduce after the average waiting time is reduced by 84.21% on average, the highest
was reduced by 86.61%, it of Reduce integration to seize the local scheduling task based on the average waiting time is greatly reduced, as shown in Figure 3.

![Figure 3. Comparison of average waiting time for Reduce tasks.](image)

The results are shown in Figure 4 for the average completion time of the local scheduling algorithm for the integrated preemption strategy. In this paper, we propose an integrated preemptive local scheduling algorithm, a no preemptive local scheduling algorithm and an integrated preemptive local scheduling algorithm. Through the observation, it is found that the average completion time of the Reduce task is 14.12% lower than that of the no preemptive local scheduling, and the maximum reduction of. It shows that the average completion time of the preemptive local scheduling algorithm is reduced. At the same time, it can be found that the completion time is reduced, and the algorithm can reduce the completion time and ensure the data locality. The experimental results are shown in Figure 4.

![Figure 4. Comparison of average completion time.](image)

**Concluding**

This paper analyzes the improvement of data locality scheduling in Hadoop cluster, and points out that the average completion time of jobs can be improved by the way of preemption under the premise of ensuring the data locality of the original algorithm. Based on the analysis of the MapReduce computing framework, this paper finds out the timing of preemption and rescheduling, so as to achieve the purpose of preemptive idle Reduce task. It can be found that the average waiting time and the average completion time of the preemptive local job scheduling can be reduced to some extent.

There are still some deficiencies in the preemptive local scheduling algorithm, such as adding preemption and rescheduling mechanism, which increases the system overhead. The experimental operation and data types are not comprehensive, in the case of large data performance test is not a lot of experiments in general there is insufficient. The next work focuses on how to reduce the overhead of the system, and how to accept the problem when the cost is acceptable.
Acknowledgements

This is supported by Liaoning Province "baiqianwan Talents Project" training funding (Project Number: 2012921041).

References

[1] Hadoop[EB/OL].[2014-2-01].http://hadoop.apache.org/.

[2] Shvachko K., Kuang H., Radia S., et al. The hadoop distributed file system, 2010 [C]. Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies. IEEE, 2010: 1-10.

[3] Dean J, Ghemawat S. MapReduce. Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[4] Zaharia M., Borthakur D., Sarma J.S., et al. Job Scheduling for Multi-user MapReduce Clusters [J]. EECS Department, University of California, Berkeley, Tech.2009, 55: 1-16.

[5] Dong Xi-cheng. Hadoop technology insider: In-depth Analysis of the MapReduce Architecture Design and Implementation of the Principle [M]. Beijing: Mechanical Industry Press, 2013.

[6] Hu Dan, Yu Jiong. Improved LATE scheduling algorithm based on Hadoop platform [J]. Computer Engineering and Application, 2014, 50(4): 86-89.

[7] He Wen-feng. Research on Hadoop job scheduling algorithm based on task characteristics and fairness strategy [D]. Hubei: Huazhong University of Science and Technology, 2013.

[8] Yan Ming-lei. Research on Job Scheduling in Hadoop Cluster [J]. Journal of Software, 2015, 14(4): 1-2.

[9] Chu Ya, Ma Ting-huai. Cloud computing resource scheduling: strategies and algorithms [J]. Computer Science, 2013, 40(11).

[10] Tao Chang-jun. Hadoop platform job scheduling algorithm [D]. Anhui: University of Science and Technology China.

[11] B. Palanisamy, A. Singh, L. Liu, B. Jain."Purlieus: locality-aware resource allocation for MapReduce in a cloud, "2011[C]. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. 2011.

[12] Mohammad Hammoud, Majd F. Sakr. Locality-Aware Reduce Task Scheduling for MapReduce, 2011 [C]. Proceedings of International Conference on Cloud Computing Technology & Science, Beijing, 2011: 570-576.