Analysis and Optimization of the Hadoop Speculative Execution Mechanism

Xianjin LUO*, Chenggang ZHEN
School of Control and Computer Engineering, North China Electric Power University, Baoding, Hebei 071003, China

Abstract — The existing Hadoop clusters are mostly composed of heterogeneous nodes, which have different computing and storage capacities, with the speed of maps to reduce tasks performed on the nodes being quite different. However, the finish time of the entire job is determined by the slowest task, so looking for the “drag tasks” strategy has a dominant position in the whole job scheduling process. The current speculative execution mechanism of Hadoop results in shortcomings to find the drag tasks in time. In this paper we try to improve the speculative execution mechanism, and then we apply the improved First-in-First-out (FIFO) scheduler to the Hadoop cluster. Experiments verify that the improved mechanism has better performance when the Hadoop platform has many drag tasks, where it can increase the cluster resource utilization and throughput.

Keywords – Hadoop; Speculative Execution; Heterogeneous Cluster; Job; Scheduler

I. AN OVERVIEW OF HADOOP

The enterprises need to access and process more and more data in today's era of information explosion. In addition to the traditional online transaction processing system and management information system, semi-structured and unstructured data grow quickly, such as internal email archiving, the dialogue recording, customer feedback records, internal network application, cooperation management system, enterprise's external website click record and the market information based on the feedback, and so on [1].

How to process the amount of data effectively and conveniently is an urgent problem, with its help, the enterprises can help the users find the valuable information quickly and get more accurate decision. At present, the traditional data analysis platform (ETL engine, data warehouse and data market) is the main application of online data analysis with structured data, the data processing mode is facing a large amount of data and semi-structured data and can't bear the complexity of the server extensions of the problem [2], therefore, the Google implemented computing model which can make full use of servers and the clusters, and several papers were published in 2004. The MapReduce programming models and the frameworks can provide automatic fault tolerance, restore function and high scalability. Now the open source Hadoop has been widely used in many fields as the basic infrastructure of the analysis system by various enterprises, such as Baidu, Facebook, and so on.

II. THE SCHEDULING MECHANISM OF HADOOP

The Hadoop cluster consists of lots of computing nodes, which are both computing nodes and storage nodes, that is to say they are divided into computing resources and storage resources, and the capacity of the Hadoop is measured by the slots of the cluster.

And the administrator can set the configuration file [3], to determine the numbers of map slots and reduce slots of the CPU, the relationship of the map slots and the reduce slots. The single core CPU is configured as only a slot, and a map slot or reduce slot can take several slots, so if a compute node has idle slot, the TaskTracker will send a request to the JobTracker and will get task from the JobTracker probably. These slots will be assigned appropriate tasks from the TaskTracker combined with the scheduler to improve the usage of resource allocation.

A job in the Hadoop cluster is divided into a number of map tasks and reduce tasks, and for the same job, there is some data dependencies between the map tasks and the reduce tasks because the input data of the reduce task is the output of the map task. By default, the reduce map was set up when the number is to the 5% of the finished map tasks. These tasks can execute only they have the special types, the map task need the map slots and the reduce task needs the reduce slots [4, 5], the slots were assigned by the scheduler separately, they were designed to the corresponding tasks.

The resource of the cluster was assigned to different tasks, which is the problem of the scheduling because the MapReduce is the computing model. The primary task is find the optimal matching between the tasks and the slots according the resource of the different nodes(including CPU, Memory and the network resource), the purpose is to fulfill the users' requirements [6,7].

Because the user requirements for the quality of service is various, therefore, task scheduling problem in distributed systems can be formulated as a multiple objective optimization problem, more specifically, this process is a typical non deterministic polynomial problems(NP, Non-Deterministic Polynomial), the ultimate goal is to find the “optimal matching” between tasks and resources.
III. THE SHORTCOMINGS OF THE CURRENT SPECULATIVE MECHANISM

There is some phenomenon that the velocity of the task is not keeping pace with each other in the distributed environment of the cluster, because some condition such as, program error, hardware failure, the uneven distribution of the resources, load imbalance, and so on. The running speed of some tasks may be significantly slower than other tasks, and this will cause increment of the completion time of the whole job, and it will affect the user’s experience of the platform. To avoid the situation, the speculative execution mechanism in the Hadoop scheduler was introduced [8,9], the mechanism can find the slower tasks in the Hadoop platform according to certain rules, and then some speculative tasks were set up, to process the same input file in the HDFS. The mechanism will choose the earlier completed tasks form the original task and the speculative task, and then another task was terminated [10].

The order to introduce the speculative task in the computing model of the MapReduce is to speed up the job to finish as quickly as possible, to maximize the completion time of the job, but the job’s completion time is determined by the slowest task. Therefore, speculative execution mechanism is to find the drag tasks, set up a new task for the drag task to process the same input file. The mechanism will choose the first one which finished quickly; now, there are some shortcomings in the existing mechanism, as follows:

- The standard to choose the drag tasks of the MapReduce computing model is the progress of the task is less than 20% of the average progress, which can be described as the formula 1.

\[
\text{progress}[i] < \text{average progress} - 20\% \quad (1)
\]

The mechanism will set up a backup task to speed up the task, but this is not considering the different performance of the heterogeneous Hadoop cluster. The gap between the different nodes is very large, so we should use the different strategies to solve these differences because we can’t find the real drag tasks in the heterogeneous cluster, in addition, the cluster resources may be wasted and will increase the load of cluster.

- There is a phenomenon that the progress of all tasks is more than 80%, the mechanism can’t find the drag task, and the mechanism will can’t set up the backup tasks for the drags. The drag tasks will have to wait for a certain time, only the system mark them failed.

- The existing policy can’t guarantee the backup task can finish quickly, the new backup speculative task need process the primary data split, so we need set up a new speculative task to speed up the progress, otherwise, the speculative task can’t have any mean. That is to say, the current solution did not consider the difference between these nodes, and this solution chooses the speculative task to other special nodes. For example, there is a task T which is a drag task, it will need 60s to be done according the speculative mechanism, and there is another task T1 will be created to process the data split, because the current mechanism can’t guarantee the execution efficiency, it need 100s to finish the task, and we can see that the task T will complete in 60s, and the speculative task T1 will be terminated. Thus, the original speculation mission is used to speed up processing speed of original data fragmentation, but it will complete behind the original task. So, we should consider the situation such as the performance of the nodes and the location of the split.

IV. THE OPTIMIZATION OF THE SPECULATIVE EXECUTION MECHANISM

As we can see in the part III, there are many faults of the speculative execution mechanism. In this part, we will improve this aspect, the detail as follows.

The JobTracker will compute the progress of all tasks for each job, and we consider only one of the tasks on the same node, assuming the task of T, which run on a computing node A. Since the JobTracker can get the process of all the tasks. There is a task T, it is process is

\[
\text{progress}[t_{\text{closed}}] \quad \text{at the time of } t_{\text{closed}}
\]

\[
\text{progress}[t_{\text{closed}} - c] \quad \text{at the moment of } t_{\text{closed}}
\]

we use the average increment of the progress with the increment between 0 and \(t_{\text{closed}} - c\), with the variance to detect the condition of the task, the formula is as follows:

\[
s^2 = \frac{1}{2} \sum_{i=1}^{2} (\text{progress}_i - \text{progress}_{\text{avg}}) \quad (2)
\]

The \(\text{progress}_i\) represents the increment at the period of \(c (i = 1, 2, \text{while } i=1 \text{ indicates the amount of the task at the period of the } 0 \text{ and the } t_{\text{closed}} - c\), and the \(\text{progress}_{\text{avg}}\) represents the average increment of the task in the time of the 0 and the \(t_{\text{closed}}\), that is to say, it is the average growth rate of progress this time.

If the variance exceeds a certain threshold, then the task is running too slow, the job which the task belongs to is related to the task’s completion time, and we need start up a new backup task to speed up the time to process the input file.
V. EXPERIMENTS AND ANALYSIS OF THE RESULTS

We build a Hadoop cluster; write the MapReduce programs which were executed on the cluster. Comparing with the original scheduler, the FIFO scheduler with the improved speculative execution mechanism have a better performance in looking for the drag tasks, and the job can be completed in a short period of time. In addition to, it can increase the cluster resource utilization and throughput of the cluster.

A. The Hadoop Cluster

Our Hadoop cluster consists of 10 computing nodes, the network bandwidth is 100Mbps Ethernet, 4-core CPU, cup frequency is 3.1GHz, and the memory is 4G. The master has the namenode and the secondary namenode daemon, the rest of the 8 nodes have the lots of the datanodes and tasktrackers. The software environment as follows: the operating system is CentOS 6.3, jdk version 1.7.0_11, Hadoop version is 1.0.4.

B. Experiments and Analysis

The experiment uses the TeraSort and WordCount MapReduce programs which can execute on the different nodes at the same time, The TeraSort was used to sort the input file which was stored in the HDFS, and the WordCount MapReduce was used to count the number of occurrences.

In the group of the first and the second experiments, we use the TeraSort MapReduce program generate 100M, 400M and 1000M files, and then we use the FIFO scheduler and the improved FIFO scheduler sort the input data sets in the HDFS with the different load factor of the cluster, as shown in Fig.1 and Fig.2.

Fig.1 and Fig.2 show that there are less slow nodes in the cluster when the load factor is normal, and most of the tasks were running on local computing nodes, so there is less drag tasks, the two scheduler have the same performance. But, there are many slow nodes in the clusters when the load is large, so the drag tasks will increase quickly. With the improved speculative mechanism, the scheduler can find the drag tasks as quickly as possible, and the job can be completed in a short period of time. In addition to, it can increase the cluster resource utilization and throughput of the cluster.

VI. CONCLUSIONS

For the reason of the shortcomings of the mechanism can’t find the drag tasks in time, this paper mainly improved the speculative execution mechanism of the Hadoop scheduler. Comparing with the original scheduler, the FIFO scheduler with the improved speculative execution mechanism has a better performance in looking for the drag tasks, and the job can finish in a short period of time. In addition, it can increase the cluster resource utilization and throughput.

ACKNOWLEDGEMENTS

This paper is supported by the Fundamental Research Funds for the Central Universities (13MS102).

REFERENCES


