The Research on Optimization Method of Hadoop MapReduce

Jian-bin Hu and Peng Wang

ABSTRACT

Hadoop MapReduce, with characteristics of large volume, various modes, fast speed and so on, is widely used in large-scale data parallel processing. In recent years, the research on Big data is more and more hot and lots of researchers are interested in optimizing the performance of Hadoop framework from all aspects. This paper firstly introduces the related work of researchers, and then analyzes the execution flow of Hadoop MapReduce framework, including assignment of state transitions and the task of life cycle. It also analyzes the reasons that affect the efficiency of the system and propose a Hadoop strategy of inner workings based on MapReduce Task.

INTRODUCTION

Big data is another disruptive technology change after Cloud computation and Internet, Meanwhile it is also the inevitable result of economic and technology development. Lots of individuals and institutions has proposed the concept of Big data. But so far there has no unified definition. Big data has the characteristics of large volume, various modes, fast speed and Low value density, which increased the difficulty of data management and information storage, therefore storage and management of Big data has been the focus in the nowadays. Hadoop is a handy windows and distributed computing platform and application program which can let users develop and Handle huge amounts of data, with the features of reliability, high efficiency, high fault tolerance, low cost and so on.

RELATED WORK

The basic framework of Hadoop was shown Fig.1, the core of the basic framework of Hadoop is the Hadoop distributed file system (HDFS) and MapReduce programming framework which is a programming model proposed by Google[1-3], using the idea of divide and rule. MapReduce distributed the operation of large data sets to a master node managed by the nodes together to
complete, and then through the middle of each node integration results to get the final result. It is widely used in the log analysis, text processing, image processing, machine learning and other fields because it can provide high performance computing capacity for the system.

MapReduce provides two very simple programming interface which are map and reduce for users. Map tasks in a completely parallel way to handle the job. The output of the map framework will first sort, and then output the results to the reduce task. Reduce tasks to the results of the protocol, and then the results of the output. Usually, the input and output of the job are stored in a distributed file system (HDFS). The application can realize automatic parallel operation on the frame when developers realized these two interfaces, while developers need not too much care for load balancing and task scheduling.

![Figure 1. Framework of Hadoop.](image)

In recent years, researchers have developed many parallel algorithms based MapReduce, such as machine learning algorithm. Some researchers has pointed out the different between the research work of MapReduce and other parallel programming model, and puts forward some problems need to be solved. In order to solve the problem of resource sharing in Hadoop cluster, FaceBook put forward a fair scheduling algorithm[4]. He deposited the intermediate data generated by the map task into a distributed memory to speed up the execution of the MapReduce task[5]. There is also proposed that the application of MapReduce can be optimized from the viewpoint of the application. Since MapReduce is iterating the data files by line-by-line parsing which can optimize the application by reusing the Writable type and using the StringBuffer[6].

It is found that the performance bottleneck in the implementation of the framework after in-depth analysis of the implementation process of MapReduce Hadoop in this paper and to reduce the heartbeat cycle by combining the setup and cleanup tasks, and then put forward a kind of optimization method from the internal implementation mechanism of Task MapReduce.

**HADOOP MAPREDUCE EXECUTION FRAMEWORK**

Hadoop MapReduce parallel computing framework based on Hadoop distributed file system (HDFS)[7], including a master node and several slave nodes, the core node is JobTracker, the function is responsible for task scheduling, management of jobs. TaskTracker is the task node, responsible for the implementation of JobTracker distribution over the task.
Jobs and tasks are two important concepts of the parallel computing framework[8]. The following describes the state transition process of the job execution and the sequence flow of tasks.

As shown in the Fig.2, the initial state of the job starts from NEW and then enters the PREP.INITIALIZING state to initialize. The purpose is to read the data block description information of the input data and create all the map and reduce tasks. After initialization, enter PREP. INITIALIZED state. At this time, a special setup task is started, which creates the operating environment for the job, and then the job enters the RUNNING phase, where the job is not processed immediately. The job in the RUNNING.WAIT state wait to be scheduled, when the task began to execute, then the job into the RUNNING.TASK state when all the map and reduce the task is completed, the job enter into RUNNING.SUC.WAIT state, this time, the other special cleanup task started, the purpose of this task is to clean up the operating environment, and then work into the end stage, the figure in each state, the job may be killed by the users, enter into the KILLED state, The process of the implementation of the task may be failed for various reasons to enter the FAILED state[9].

![Figure 2. The state transition of a job.](image)

Task is the basic unit of Hadoop MapReduce framework for parallel computing. The whole life cycle of Task is an important object to be analyzed. In the implementation of framework, both ends correspond to TaskInProgress and
TaskTracker.TaskInProgress. When a job is submitted to the Hadoop system, JobTracker will initialize the job, then the task within the job is all created, waiting for TaskTracker to request the task, we can simply analyze the task life cycle.

When TaskTracker accept the task, create TaskInProgress object, Task in UNASSIGNED state at this time, after a heartbeat cycle, TaskTracker requests to assign task, JobTracker assign a TaskInProgress task to the TaskTracker after received the request, this is the first heartbeat.

The TaskTracker creates a TaskTracker.TaskInProgress object when it receives the task and starts the Child process to perform the task. The TaskTracker updates the task status to RUNNING. After a heartbeat cycle, the TaskTracker reports the Task status changes to the JobTracker, which also updates the status to RUNNING. This is the second heartbeat cycle. TaskTracker also periodically sends heartbeat messages to the JobTracker during task execution. After that, TaskTracker changes the status to COMMIT_PENDING and sends it to JobTracker.

After a message is received, the JobTracker will return a confirmation message indicating that it is allowed to commit[10]. The TaskTracker will submit the results and update the task to SUCCEEDED. Another cycle, the TaskTracker will send the heartbeat message again. The JobTracker will also update the task status to SUCCEEDED, thus, the life cycle of task ends.

PROBLEM ANALYSIS AND OPTIMIZATION STRATEGY

After a message is received, the JobTracker will return a confirmation message indicating that it is allowed to commit. The TaskTracker will submit the results and update the task to SUCCEEDED. Another cycle, the TaskTracker will send the heartbeat message again. The JobTracker will also update the task status to SUCCEEDED, thus, the life cycle of task ends.

By analyzing the state transition process of the job, we can see that there are three tasks in the life cycle of a job. Before executing all the common tasks, it will perform a setup task and then execute the normal MapReduce task. After all the tasks are executed Upon completion, a cleanup task is performed. According to the life cycle of the task we can know the setup task execution must go through two steps.

1) Assign the setup task, after a heartbeat cycle, When JobTracker receives a task request from the TaskTracker, then Setup task is distributed to the TaskTracker.

2) When the Setup task is complete by TaskTracker, The completion information is reported to the JobTracker through heartbeat information, This is the second heartbeat.

add another point, The TaskTracker also periodically sends heartbeat messages to the JobTracker during the execution of the task. But the above two heartbeat information is inevitable, there must exist.

Likewise, the Cleanup task consumes at least two heartbeat cycles. The heartbeat message contains task request information, task execution status, node information, and so on. In the standard Hadoop system, a heartbeat period of less than 100 nodes is 3s. When the cluster is greater than 100 nodes, each additional 100 nodes, heart rate increased by 1s. we can calculated that the Setup and Cleanup tasks took at least 12 seconds. Although 12s time is not long. However, for a 1-minute segment of work, the creation and cleanup of the entire operating
environment costs about 20%. This has a significant impact on the performance of the system.

By analyzing source code of Hadoop, we found that the main task of setup is to create a temporary output directory, and the cleanup task is to remove the directory. These two tasks itself is not long, the real time-consuming is JobTracker and TaskTracker waiting the heartbeat each other, so we consider the following from the perspective of optimization strategies to reduce the time of executing.

Merge the setup and clean tasks into the mapreduce task, That is to say, when a job completes initialization, we start the MapReduce task directly. The first thing is executing Setup task at the beginning of Map task to create the environment of operating jobs and the last thing in the Reduce is Cleanup task to remove the temporary directory. To do this, we can add a setupJob () method to the MapTask class and add a cleanJob () method to the ReduceTask class. At the very beginning of the Map task, call the runJobSetupTask () method of he Task class, call the runJobCleanupTask () method of the Task class at the end of the reduce task, and then job ends.

After optimization, there is no PREP.SETUP state in the flow chart, and it is merged into the RUNNING.WAIT state and the CLEANUP state is merged into the RUNNING.SUC.WAIT state. Since the heartbeat communication time is omitted at least four times, the execution efficiency of the frame can be remarkably improved.

CONCLUSION

Hadoop MapReduce is one of the most successful and widely used parallel framework. This paper propose optimization plan to optimize the environment preparation and cleaning in the process of execution by analyzing inner of the framework of job execution process. As a result, it effectively saves at least four heartbeat cycle time and realize the job execution performance optimization. Now, Big data processing has been applied to all aspects of life, therefore, the research on Hadoop MapReduce has great significance, and future research will be focused on the following direction. Making optimization from task scheduling algorithm and change the default scheduling algorithm to let system assign tasks to slave nodes flexibly according to the load situation. From an overall perspective, we can use multi-core, multi-GPU, Network Attached Storage and so on to further strengthen the function of Hadoop, combined with the characters of big data.

REFERENCES