A recovery mechanism of MapReduce model based on transferring dependency

Li Zhuyu[1,a], Li Fucong[1,b], Li Xiangxin[1,c]

[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China

[a]xiaoyu3696@126.com, [b] li274860615@163.com, [c]lixiangxinustb@163.com

Keywords: recovery; JobTracker; dependency; MapReduce; Zombie task; Synchronization mechanism; Backup mechanism

Abstract. MapReduce is a popular big data processing tool, but there exists a main node failure problem. In order to make the restarted JobTracker recover jobs site quickly, there are mainly two recovery mechanisms: synchronization mechanism and backup mechanism. In synchronization mechanism, TaskTracker sends synchronous message to JobTracker within required time in order to recover jobs without considering dependency between Map and Reduce tasks, which leads to congestion**.** Using backup mechanism, JobTracker recovers jobs based on backup records ignoring assigned tasks between backup intervals (which is called zombie tasks). That will result in repeated task assignment. We come up with a new backup mechanism based on transferring dependency, with the addition of dependency list mechanism and zombie task recovery mechanism. The new mechanism effectively solves problems of congestion and duplicated distribution of zombie tasks, making the jobs continue running from the breakpoint.

Introduction

Hadoop is a software platform for the developing and operating big data, MapReduce is computing framework of Hadoop platform, responsible for distributed computing[1]. In the processing of TB and PB level data, MapReduce has become one of the most widely used parallel programming model[2].There is a main node, called JobTracker[3,4], and several worker nodes, called TaskTracker, in the MapReduce model.

Currently synchronization mechanism[5] and backup mechanism[6] are mainly adopted by Hadoop system to deal with the main node failure. In synchronization mechanism, TaskTracker sends synchronous message to JobTracker within required time in order to recover jobs without considering dependency between Map and Reduce tasks. That may lead to congestion[7]. The Hadoop that uses a backup mechanism regularly backups job information normally. Although JobTracker restores the job site according to backup records after restart, it ignores the existence of zombie tasks in the backup intervals, thus resulting in duplicated distribution of zombie tasks, waste of system resources, and abnormal operation.

Thus we come up with a new backup mechanism based on transferring dependency, adding in dependency list mechanism and zombie tasks recovery mechanism. The new mechanism solves congestion and duplicated distribution of zombie tasks problems as mentioned before effectively, making the jobs continue working from the breakpoint.

Congestion problems in synchronization mechanism

Specifically speaking, in synchronization mechanism, JobTracker sends a synchronization request (Ask Sync) to TaskTracker after restart, and TaskTrackers send their tasks' status information to JobTracker for synchronization.

This mechanism may cause task congestion problems, resulting in a lockup of the whole job. Congestion appears in the synchronization mechanism when there is an incomplete update. Incomplete Update means the occurrence of exception in some worker nodes that did not send the synchronization information to JobTracker, which renders the JobTracker unknown of the existence

of the abnormal node after it restarts. This incomplete update is closely related to the dependencies between Map and Reduce tasks in MapReduce.

Exception in backup mechanism

In the backup mechanism, Hadoop regularly stores the job's overall status information in HDFS[8] (Hadoop Distributed File System), and HDFS will store the job information orderly in a directory way based on completion time, including job ID, job status, job description, job completion events etc,. The keep time of job information can be set up in HDFS, which can be automatically cleaned up when expired [6].

Since there is a time interval between the last backup and the restart of JobTracker after failure, that is, the backup interval, the task (zombie task) executed by TaskTracker during the backup interval has no backup record in HDFS. Unknown of the existence of these tasks after restart, JobTracker therefore would redistribute these tasks, causing jobs exception.

The new backup mechanism based on transferring dependency

Taking into consideration the problems currently existed in the two recovery mechanisms, a new round backup mechanism based on transferring dependency is put forward. The new mechanism to which the dependency list mechanism and zombie task recovery mechanism were added on basis of the original backup mechanism aims for solving the congestion problems during the recovery process and eliminating exception.

Each Reduce task maintains the data acquisition list which contains the acquisition address of intermediate results from the Map task needed when executing the Reduce task, that is, the dependent Map task ID and the task execution node. We call this list Data Dependence Node List (referred to as DDNL) [7]. Set it as D set. Online Node List (referred to as ONL) refers to the node list that successfully responds to JobTracker's control message within heartbeat cycle T, that is, the normal operation node list, which is set as O set.

Let's consider ways to recover the job under different circumstances respectively. First, take the simplest case1 into consideration. That is, no zombie task and no abnormal node in the backup interval, job site can be restored according to the backup information only.

The second case is that there is zombie task in backup interval and there is no abnormal node during recovery. JobTracker sends control messages to all worker to inquire whether or not there is a zombie task after restart within heartbeat period T. After the worker nodes receive the message, they will make the judgment according to their own condition: Workers that have zombie task will send to JobTracker the heartbeat that contains zombie task status information; the remaining other Workers will send heartbeat only reporting the node status to JobTracker. JobTracker can then restore the job site in accordance with all heartbeat messages received and backup information.

There is also another case3: that is, no zombie task in the backup interval but having abnormal nodes during the recovery process. If there were nodes that belong to Set D but do not belong to set O, these nodes would be called set R, namely a set of abnormal nodes. The nodes set that is dependent on nodes Set R is referred to as Set W. JobTracker sends a control message to all the worker nodes after restart and each node will respond after receiving the message, and JobTracker will get ONL after the end of the T time. If Set R is not empty, then JobTracker will send control messages to ONL requesting their DDNL. If Set W is not empty, then nodes in the Set W will make response respectively after receiving the control message and send their DDNL to JobTracker. JobTracker will send control message to the Set W after comparison between all DDNL and set R, which requires the ignorance of the corresponding nodes of the Set R, and distributes the Map tasks of nodes in Set R to new nodes, and notifies that the nodes in Set W depend on the new nodes. As shown in Fig. 1:

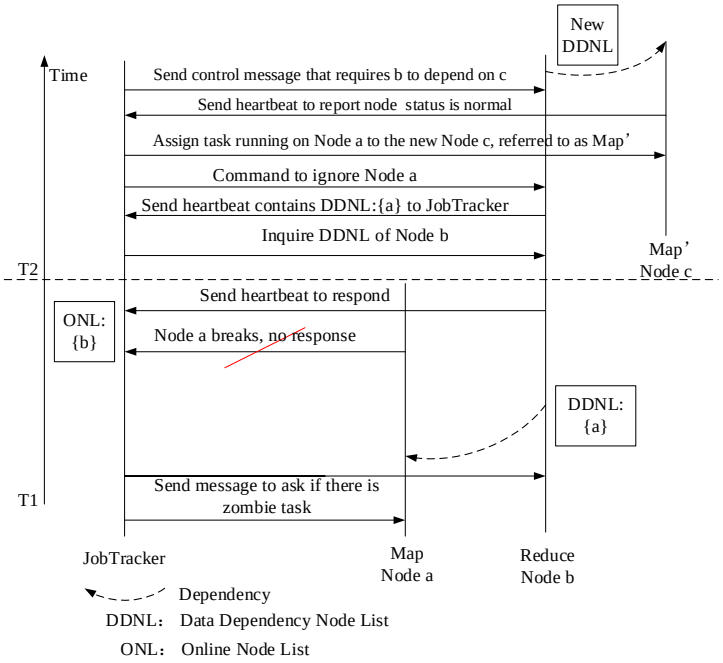


Fig. 1 Recovery process of Hadoop in case 3

The last as well as the most complicated case is that there are both zombie task and abnormal node. The job recovery process at this time is shown as Fig. 2:

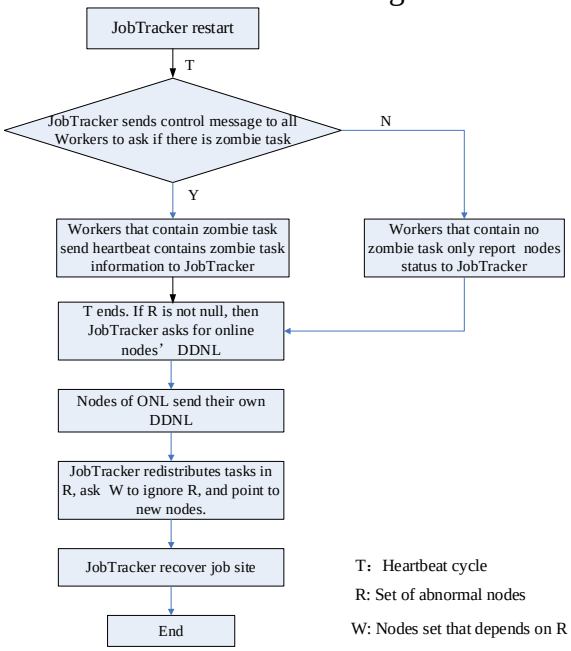


Fig. 2 Flow diagram of job recovery in case 4

Experimental environment

Set the experimental environment as follows: Configure 16 severs of Dual Quad Core Xeon E5335 2.0GHz CPU, 4G memory, within a cluster and connect servers with 100M Ethernet. Install XenServer virtualization platform on all servers [9]. Install on each server a virtual machine system configured as shown in Table 1. Class A node, in Table 1, contains one server on which a 4-core CPU, 1G memory VM(virtual machine) was installed. Class B node contains 12 servers, with each server installed two 4-core CPU, 1G memory VM; class C node contains three servers, with each sever installed four 2-core CPU, 512M memory VM. All nodes are installed with Debian Linux Ecth 4.0 operating system. Class A node stands for JobTracker; class B and class C nodes contain a total of 36 VM and are used as TaskTracker, forming a heterogeneous environment. As we can see from Table 1.

Table 1 Experimental environment configuration

Configuration \ Node	Class A	Class B	Class C
Server number	1	12	3
Number of VM in each server	1	2	4
Sum of VM	1	24	12
Number of CPU core in each VM	4	4	2
Memory	1G	1G	512M
OS	Linux Debian Etch 4.0		

Experimental results and performance analysis

Example run by the experiment is Hadoop system's build-in word counting program - WordCount. It can calculate the number of times appeared in the specified data congestion. In order to obtain better statistical results, we started 500, 1000 and 1500 same jobs respectively in the experiment, which runs as follows: divide them into 100,200 and 300 groups, each group with five jobs; each group runs in a serial way, while jobs inside a group run in a concurrent way.

Within the running time of jobs in each group, run a manual shutdown and restart of JobTracker at a randomly selected time point in order to simulate the failure of main node. The operating system's average restart time serves as the pause interval. Reference group runs normally and main node won't be restarted; the experimental groups which use the synchronization mechanism, the backup mechanism and the improved mechanism respectively obtain the comparing results shown in Table 2 and Table 3 below.

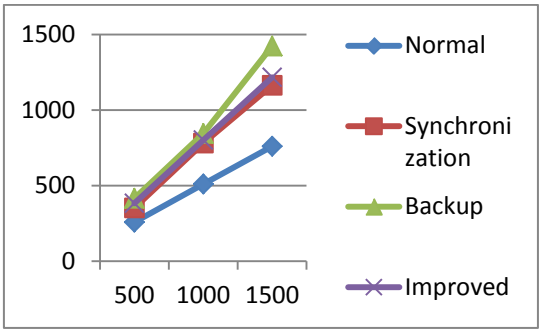
Table 2 Recovery time comparison

Recovery time \ Workload	500	1000	1500
Normal	258.12	510.22	760.24
Synchronization	352.25	780.66	1163.19
Backup	415.63	844.53	1422.96
Improved	382.54	801.39	1216.62

Table 3 Congestion times comparison

Congestion times \ Workload	500	1000	1500
Normal	0	0	0
Synchronization	9	16	25
Backup	30	92	145
Improved	4	7	15

As can be seen from Table 2, in the case of not restarting JobTracker, the average time for each group to perform the job is 258.12s, which represents the average performance of the system. And yet after a JobTracker failure occurs in the job, since it takes time for JobTracker to restart and restore global information, the three recovery mechanisms therefore have varying degrees of slowing down in Table 2. We can also see that the using improved mechanism, Hadoop has the minimal congestion times. At the same time, job sit can be recovered as soon as possible compared to other recovery mechanisms. From the tables above, we can get Fig. 3 and Fig. 4 below which are more clear.



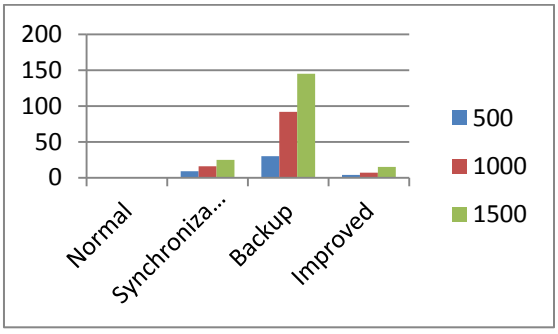


Fig. 3 Recovery time comparison Fig. 4 Congestion times comparison

As can be seen from Table 3 and Figure 5, with the increased workload, the recovery time of systems using synchronization mechanism and backup mechanism does not have a surge in trend,

but there is a significant increase in congestion times. This is because the probability of the backup mechanism and synchronization mechanism encountering the previous congestion problem is greatly increased. In actual situation, synchronization mechanism produces more abnormal conditions, because the stability level of a worker node in the laboratory equipment goes far beyond the reach of that in real life situations.

Compared to the synchronization mechanism and backup mechanism, the improved recovery mechanism has the minimum results, whether in recovery time or in congestion times, and there is no significant increase, which indicates that the improved one enhances the fault tolerance of the Hadoop cluster.

Conclusion

MapReduce model is widely used in big data processing field for its high degree of parallelism and scalability advantages. The industry’s typical representatives like Yahoo, Amazon and IBM take it as a basic computing model for cloud computing platform, and apply it to Internet computing services, enterprise computing services and scientific computing services [10]. Under such large-scale demand, higher requirements for the stability of MapReduce model are also put forward. Hence, the failure recovery issues of the main node attract the industry's attention. We present a new, more comprehensive recovery mechanism based on transferring dependency to address the congestion and abnormal problems that exist in the current recovery mechanism, allowing the main node to resume jobs quickly and accurately after restart, and making them proceed at the breakpoint.

References

[1] Apache Hadoop [Z]. http://lucene.apache.org/hadoop/,2010-10-15/2010-12-28.

[2] LI Jian-jiang, CUI Jian, WANG Dan, YAN Lin, HUANG Yi-shuang: Survey of MapReduce Parallel Programming Model[J].ACTA ELECTRONICA SINICA,2011,39(11):2636-2641.

[3] Dean J, Ghemawat S. MapReduce: Simplied Data Processing on Large Clusters[C].Proc of OSDI.04,2004:137-150.

[4] Zaharia M，Konwinski A, Joseph A D, et al. Improving MapReduce Performance in Heterogeneous Environments[C].Proc of OSDI.08,2008:29-42.

[5] Hadoop-3245, Provide ability to persist running jobs[Z]. [2009-07-03].

https://issues.apache.org/jira/browse/HADOOP-3245.

[6] Hadoop-1876, Persisting completed jobs status [Z]. [2009-07-03].

https://issues.apache.org/jira/browse/HADOOP-1876.

[7] ZHANG Zhao-ning, PENG Yu-xing: A Method for Solving the Congestion Issue During the Single Node Recovering Based on the MapReduce Model[J].Computer engineering and science, 2011,33(3):146-148.

[8] http://hadoop.apache.org/hdfs/.

[9] http://www.citrix.com/xenserver.

[10]LI Cheng-hua, ZHANG Xin-fang, JIN Hai, XIANG Wen: MapReduce: a New Programming Model for Distributed Parallel Computing[J]. COMPUTER ENGINEERING & SCIENCE, 2011,33(3),130-132.