Analysis of Information Management and Scheduling Technology in Hadoop

Ma Weihua, Zhang Hong, Li Qianmu, Xia Bin
School of Computer Science and Technology
Nanjing University of Science and Engineering
Nanjing 210094, China
kryolith.xiabin@gmail.com

Journal of Digital Information Management

ABSTRACT: Development of big data computing has brought many changes to society and social life is constantly digitized. 'How to handle vast amounts of data' has become a more and more fashionable topic. Hadoop is a distributed computing software framework, which includes HDFS and MapReduce distributed computing method, make distributed processing huge amounts of data possible. Then job scheduler determines the efficiency of Hadoop clusters and user experience. Under the premise of familiar with the mechanism of Hadoop's running tasks, make a full analysis of the existing Hadoop task scheduling algorithm, such as FIFO-Scheduler, Capacity –Scheduler, FairShare-Scheduler and LATE-Scheduler, found that the existing scheduling algorithms do not perceive the performance of the computing node, so that it cannot assign different tasks depending on the machine performance in heterogeneous Hadoop cluster.

Subject Categories and Descriptors
H.2.8 [Database Applications]: Data Mining; **H.3.3 [Information Storage and Retrieval]:** Information Search and Retrieval

General Terms: Data Mining, Data Clusters, Hadoop

Keywords: Cloud Computing, Hadoop, Task Scheduling, Map Reduce

Received: 11 November 2013, Revised 13 February 2014, Accepted 26 February 2014

1. Introduction

Hadoop is a distributed computing software framework sponsored by a famous open source software foundation Apache. This software is developed using Java program language. Users can develop distributed computing software, fully using the power of clusters to do high-speed calculation and storage, without knowing the details of the technology [1].

Data need to be read in the process that compute nodes execute the Map task. However, a non-local task must be localized firstly, which need consume some network resources and local disk bandwidth. Thus, network bandwidth of Data node and operations of disk IO can be reduced effectively if only migration of data of task MapReduce is reduced. While for the task itself, data localization would not be needed if they are exactly local. Obviously, this step for a task with a lot of data, still need a certain amount of time. So it's wise to improve the rata of Map task's input data localization to reduce single execution time of task Map.

The first paragraph introduces the framework of Hadoop and its job scheduler. The second paragraph analyzes the Hadoop task scheduling algorithms like IFO-Scheduler, Capacity–Scheduler, FairShare-Scheduler and LATE-Scheduler, concluding that there exist some problems in these scheduling algorithms. At last, we generalize the problems that exist in the scheduling algorithms assigning tasks in heterogeneous Hadoop cluster.

2. The Framework of Hadoop and Work Distributing

Inspired by Google's Mapreduce and GFS, Hadoop

succeeds in developing a distributing file system HDFS (Hadoop Distributed File System) and the realization of the open source of Mapreduce. With the wide use of Hadoop, open source community has developed the distributing database HBase, big data analysis tool Pig, Hive and so on [2]. In order to analyze the whole clusters' data, they develop the Chukwa and Zookeeper, which are the coordination components of the whole Hadoop. These components make up the Hadoop Cloud Platform Ecosystem. The composition structure is shown as Figure 1.

Figure 1. The schematic diagram of the composition structure of Hadoop

Firstly, HDFS is a distributing file system, which is erected

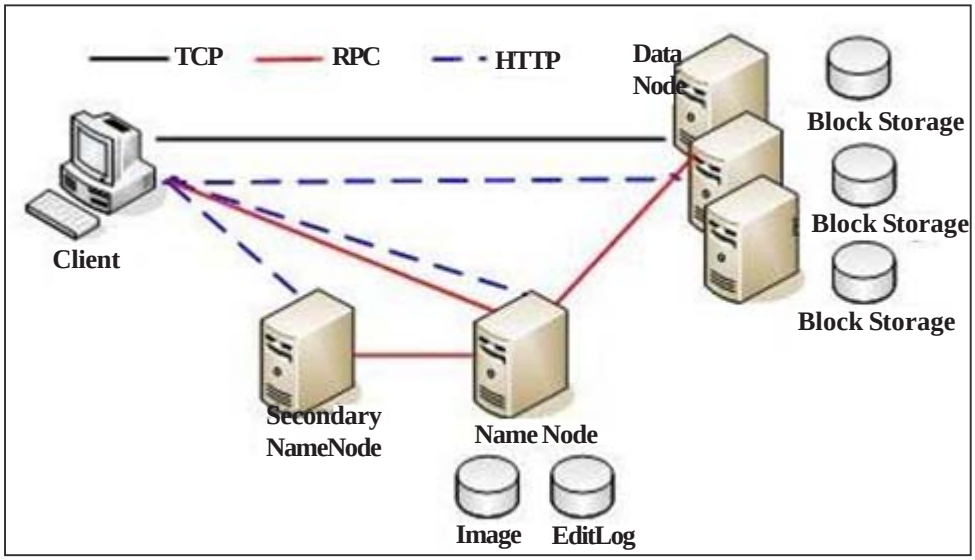

Figure 2. The structure of HDFS system

over the local computer's file system. It can provide distributing file storage, especially suitable for huge data storage and the condition when you save once and read for many times. HDFS saves files by providing backup, so its reliability is high. HDFS use Master-slave structure; the managing node is called NameNode. The controlled node is called DataNode. The node that monitors NameNode is called Secondary NameNode.AS is shown in Figure 2, DataNode is responsible for storage. It can save, modify or delete the files locally based on the instructions of the NameNode. As a back end, Secondary NameNode tests the status of the NameNode regularly.

Secondly, Hadoop realize the calculation structure of MapReduce. Users can write their own MapReduce programs to realize distributing calculation. In the structure of MapReduce Google has published, the main controlled nod Master is called JobTracker Node, which is responsible for the managing work of the computing nodes in the whole clusters. Working nodes worker is called TaskTracker Node. Every TaskTracker is in charge of several slots. Every computing slot runs a task.

Assuming that in Hadoop clusters, there are altogether n computing nodes named TaskTracker. Define them as $node[] = \{node_1, node_2, ..., node_n\}$. Every computing node is equipped with several Map computing slots and Reduce computing slots. These computing slots are equipped with a series of resources, such as CPU, memory and disk. That is, $Slot = \{cpu, mem, disk, ...\}$.

After the work users have submitted enters into the queue waiting to be distributed, JobTracker will initialize it and divide it into several Map tasks and several Reduce tasks, that is, $Job = \{t_m^1, t_m^2, ---, t_m^i, t_m^1, t_m^2, ---, t_m^k\}$,in which t_m^i represents Map's subtasks and t_r^k represents Reduce's subtasks.

TaskTracker reports its current conditions to JobTracker in a fixed cycle. If there is a task running on the Node presently, TaskTracker will report the progress rate and then JobTracker updates the state of the task. If there is an available slot on TaskTracker, it will ask for tasks from JobTracker through heartbeat packet. JobTracker assigns tasks using task scheduler. In fact, task scheduler is choosing a suitable job from the work queue based on some rules and then choosing some tasks from the job to give them computing resources. The mathematical model of task assignment is shown as formula (1).

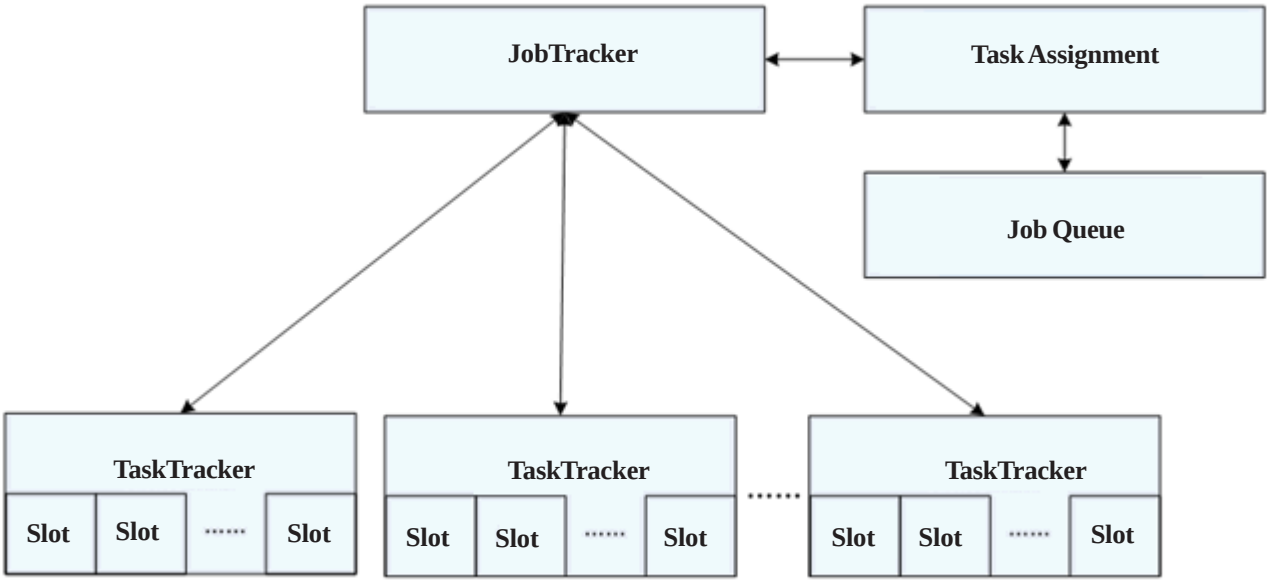

Figure 3. The schematic diagram of Hadoop's task assignment

$$Assign\ (taskTracker) = \{t, slot_1, slot_2, ...slot_x \mid x < availableSlot\} \quad (1)$$

The schematic diagram of Hadoop's task assignment is shown as Figure 3. When TaskTracker receives the task messages that JobTracker sends, it will add these tasks into execution queue and then submit to task execution object to do them.

3. The Leading Scheduling Algorithms and Analysis of Their Features

3.1 FIFO-Scheduler

In Hadoop, '*First in first out scheduling algorithm*' is used as the acquiescent scheduling algorithm. When Hadoop clusters are not equipped with any scheduling module, JobTracker acquiescently calls First in first out scheduling module to assign tasks.

'*First in first out scheduling algorithm*' sorts the tasks in the ready queue according to the deadline of the tasks. JobTracker gives priority to the scheduling of the task in the head of the queue until the end of the task before the next mission in turn. In subsequent versions of Hadoop, to be able to consider the priority of the task, it considers the task priority in the FIFO scheduling algorithm. When sorting the tasks, it gives the priority to the comparison of the priority level of the tasks; task of higher priority level came in front of the queue.

The biggest advantage of the FIFO scheduling algorithm is that it realizes very simple and computational load on JobTracker is relatively small. When the number of clusters is very large, its requirement to the computer power of JobTracker is not high and it doesn't need to take up too much memory resources. And if the number of Hadoop cluster machine reaches a certain magnitude, JobTracker is the bottleneck of the entire cluster. So for large clusters is that FIFO would be a good scheduling algorithm.

It is precisely because it realizes very simple, so its shortcomings is very much. For example: no analysis of the compute nodes, leading to not be able to assign tasks according to the compute nodes; no analysis of the nature of the job, in total disregard of the nature of the job in the process of assigning tasks, likely to cause unreasonable job assignment. If you have a large job, you need to perform a long time. However, these jobs do not urgent. FIFO scheduling algorithms assign this type of job, the job may lead to long-term occupation of the cluster system but some urgent short job cannot be able to get the computing resources. Encountered such a problem may easily affect the user experience.

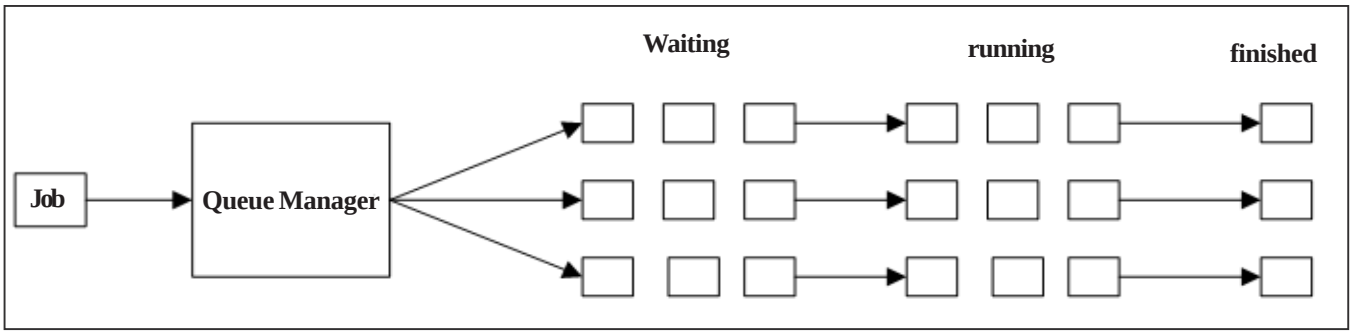

Figure 4 . Capacity Scheduler

3.2 Capacity-Scheduler

Capacity-Scheduler is a scheduling algorithm launched by yahoo. To be able to make full use of the cluster computing resources, Capacity Task Scheduler divide the whole Hadoop cluster into several job waiting queues.

When each Hadoop user submits the job, JobTracker will submit it to the specified queue according to the job properties set up by users. If it has no specified queue, Job Tracker will store the job in the default queue. In each job queue, Capacity-Scheduler ranks the jobs by FIFO scheduling algorithm, that is to say, the order of job execution in the queue is based on the submission time. While among queues, the amount of computing resources that each queue can own is determined by configuration information in configuration profiles.

Capacity Task Scheduler supports multiple queues sharing computing resources of the whole Hadoop cluster. All the queues protect the jobs submitted by users. Resources settings to these queues start according to configuration information when computing power scheduler switches on. When jobs are submitted, Capacity Task Scheduler puts them in the default queue according to configuration information. The running process of the scheduling algorithm is shown in Figure 4.

To make full use of computing resources of the whole cluster, when resources that the queues assigned are not used, Capacity Scheduler will assign them to other queues. When resources are occupied by other queues, scheduler will assign these resources that are occupied back if the queue takes on heavier loads.

To reflect the priority of the job, Capacity Task Scheduler provides configuration items. If Hadoop cluster administrator deploys mapred.capacity - scheduler. queue. queue-Name.supports - priority, priority of the job can be set when submitting the job, which will be an important basis during job dispatching.

Capacity-Scheduler divides the whole Hadoop clusters into several sub-cluster systems. Every line logically corresponds to a sub-cluster system. In every sub-cluster system computing resources can be distributed dynamically. What arouses the most attention in capacity-Scheduler is that this scheduling model provides support for large memory work. When some work need large memory, there is almost no way to distribute computing resources for common scheduling algorithms. Capacity-Scheduler is able to integrate the memory resources of several computer slots and distribute the task, based on users' need. Capacity-Scheduler has many advantages that FIFO doesn't have. For example, it supports several lines, which can distribute the works to different lines according to whoever submits the works and wait to enforce them. What's more, it can guarantee every line obtaining clusters' resource, in which every job is able to obtain these computing resources. Last but not least, it supports dynamic resource distribution, the priority of jobs and memory-intensive jobs.

However, Capacity Scheduler takes data localization into final consideration. The tasks that allocated in this way have low data localization, requiring downloading data in other HDFS nodes when executing Map tasks. In large-scale data center, broadband network has reached its limits. Downloading data fragmentation from other nodes puts great pressure on network undoubtedly as well as disks of local computing nodes, which extends executing time of the jobs.

Capacity Scheduler puts forward a motion that supports large memory work. However, its supporting way is just an elementary attempt. If users need to submit large memory work, they need to surmise how much memory the job needs before they submit it. However, these users are not sensitive to the memory the job needs in usual circumstances. Therefore, it is not a feasible way to let users surmise memory. As to the support for large memory jobs, a feasible way is to make the machine surmise the needed memory. In open-source community, there are already some contributors putting forward a feasible scheduling module to surmise the needed memory in a task.

3.3 FairShare-Scheduler

FairShare-Scheduler is a combination of the company's data processing services Facebook to develop. In Facebook company requires frequent data mining, due to the different processing logic, algorithms for task scheduling requirements is not the same. For example, Hive and other tasks required response time is short, you must promptly return result; while some background data processing need not timely return the results of the business, such as cluster idle period can be processed. So, FairShare-Scheduler will be in the division of the entire computing cluster into multiple resource pools (Pool), each holding a certain resource pool of computing resources, resource pools to accommodate some of these jobs, and take different depending on the configuration information scheduling policy, while also being able to share various resources dynamically pool resources.

FairShare-Scheduler according to the configuration information will divide the entire cluster resources into multiple resource pools, jobs submitted by the user will get these resource pools of computing resources, and then dispatch scheduling policy based on these resource pools, and perform these operations [3].

FairShare-Scheduler compared to other scheduling algorithms has many advantages [4]:

1. It can be very fast even in the shared cluster running short jobs. Unlike the FIFO by certain operating exclusively cluster.

2. In a shared cluster production environment to give a job to provide adequate assurance that the experimental nature of the job is run separately.

3. The Support for resource preemption can make the task of high priority precedence; support delay scheduling can improve the localization rate data.

4. The fair scheduler provides management interface that can display real-time scheduling information on the web.

The new version still does not support large memory scheduler job scheduling and capacity as early as the types of jobs have support. Due attention is fair scheduling algorithm fair share of slots, and slots in a heterogeneous cluster ability differences, the surface is fair, but in terms of computing power slot is not fair to see, which led to fair scheduling algorithm performance on heterogeneous clusters is very poor [5].

3.4 LATE-Scheduler

Since Hadoop was considered to run in isomorphic clusters when designing, it's simpler to forecast progress in the implementation of tasks belonging to the same job in isomorphic cluster [6]. For example, if a task is running behind the same type of task by 20%, JobTracker will sign the task as a backward task and open a backup task [7]. While, progress in the implementation of tasks varies due to different hardware configuration in heterogeneous clusters. So that lots of backup tasks are generated in system and system resources are wasted.

LATE (Longest Approximate Time to End) is a scheduling algorithm for heterogeneous clusters proposed by University of California, Berkeley, which can solve speculation and execution of backup tasks effectively in heterogeneous clusters. LATE sorts tasks based on response time of tasks in the job and response speed of computing nodes then picks up backup tasks needing speculative execution from them. There are also some thresholds to limit backup task growth, avoiding too many speculative tasks.

LATE-Scheduler sets the following three parameters:

SpeculativeCap: The largest quantity of parallel running backup tasks in Hadoop system; (The recommended value is 10% of the total calculated slot.)

SlowNodeThreshold: Mark node speed threshold. Slow nodes, whose scheduling score is below the threshold, will not start backup tasks;

SlowTaskThreshold: Mark whether the threshold for the task starting speculative tasks automatically. When task progress is below average progress of the same type tasks, out of that threshold, scheduler will start backup tasks automatically.

TimeLeft: Tasks' remaining time. Calculation method, see equation (2).

$$ProgressRate = ProgressScore / t \quad (2)$$

$$TimeLeft = (1 - ProgressScore) / ProgressRate$$

In equation (2), *ProgressScore* is the value of the progress of the task. *ProgressRate* is task execution rate.

LATE scheduling policy is performing the following operations when a node asks JobTracker for assignment, while quantity of all speculative tasks in system is smaller than *SpeculativeCap*:

If attribute of the node is great and it performs tasks very quickly, scilicet node score is higher than, request made by the node will be ignored;

Sort currently running tasks (not including running backup tasks) according to estimated remaining completion time.

Choose a task that cost most time, its current progress below *SlowTaskThreshold*, then execute speculative task for it.

LATE-Scheduler considers heterogeneity of nodes when choosing nodes to execute backup tasks. *SlowNodeThreshold* is set in LATE-Scheduler to distinguish nodes' speed. So that resources of Cluster System can be used effectively when choosing nodes to execute backup tasks. Whereas, general scheduling algorithm in Hadoop, not considering heterogeneity of nodes, may execute backup tasks in very slow execution nodes, and then tasks execution speed of the machine is slower. Such an allocation strategy cannot only fail to improve performance, but also reduce the throughput of the system and waste computing resources of the system.

Whereas, there are also some limitations of LATE. LATE-Scheduler divides tasks into three stages with equal ratio when classifying Reduce tasks. For many tasks, the three sub-process consume different time. In that case, LATE judging tasks execution speed goes wrong.

Furthermore, LATE-Scheduler remains to be improved about methods to distinguish fast nodes and slow nodes. Considering heterogeneity of system and that each computing node has different resources (like CPU, Memory), it is possible that some nodes execute Map tasks fast but execute Reduce tasks very slowly.

4. Questions about Scheduling Algorithm Assigning tasks in Heterogeneous Hadoop Clusters

Following are two questions in heterogeneous Hadoop clusters:

1. Due to the different hardware configurations and different computing power, existing task scheduling algorithms are built under the premise of homogeneous clusters and they take quantity of computing slots in computing nodes when assigning tasks. LATE is the only one to consider heterogeneous clusters, but it is only used to detect backward tasks in heterogeneous clusters and allocation results obtained are not very satisfactory. While, Hardware Configuration determines computing ability. Sorts of factors

affecting tasks operation progress, like speed of IO disk, are not taken into consideration.

2. Configuration information differentiation of clusters causes a change in data storage strategy. Due to the differentiation of clusters, configuration information of each data node in HDFS is different. Data block each node can store changes according to total data capacity, so it has great impact on localization ratio of the data.

5. Conclusion

Hadoop job scheduling determines the efficiency of Hadoop clusters and user experience. However, the current scheduling algorithms of Hadoop do not consider cluster heterogeneity. Job scheduling for heterogeneous Hadoop clusters efficiently has been a challenge. Under the premise of familiar with the mechanism of Hadoop's running tasks, make a full analysis of the existing Hadoop task scheduling algorithm, such like FIFO-Scheduler, Capacity –Scheduler, FairShare-Scheduler and LATE-Scheduler: is a little simple and cannot complete allocation that grows more complicated; Capacity –Scheduler divides jobs into several queues though, it also use FIFO-Scheduler internally. For jobs in the same queue, problems in FIFO-Scheduler still exist in Capacity –Scheduler; FairShare-Scheduler considers sharing cluster computing capability equally, while the equity is just limited to sharing computing slots fairly. Different slots own different computing capability, which is not fair at all;

LATE-Scheduler starts with solving backward tasks, to a certain extent probing backward tasks, but not that ideally in actual use.

We found that the existing scheduling algorithms do not perceive the performance of the computing node, so that it cannot assign different tasks depending on the machine performance in heterogeneous Hadoop cluster.

Acknowledgments

This work has been funded by College Natural Science Foundation of Jiangsu Province (11KJB520002), Jiangsu 973 Scientific Project (BK2011023, BK2011022), the National Natural Science Foundation of China (61272419, 60903027), China postdoctoral Foundation(2012M521089), Jiangsu Postdoctoral Foundation(1201044C), Jiangsu Natural Science Foundation(BK2011370), Research Union Innovation Fund of Jiangsu Province (BY2012022).

References

[1] Rasooli, A., Down, D. (2011). An adaptive scheduling algorithm for dynamic heterogeneous Hadoop systems. Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research. IBM Corp., p. 30-44.

[2] Li Qianmu, Li Jia. (2012). Rough Outlier Detection Based Security Risk Analysis Methodology. *China Communications*. 5 (7) 14-21.

[3] Fair Scheduler Guide [OL] http://hadoop.apache.org/common/docs/current/fair_scheduler.html.

[4] Moseley, B., Dasgupta, A., Kumar, R, et al. (2011). On scheduling in map-reduce and flow-shops. *In*: Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures. ACM, p. 289-298.

[5] Xia, Y., Wang, L., Zhao, Q, et al. (2011). Research on Job Scheduling Algorithm in Hadoop. *Journal of Computational Information Systems*, 7 (16) 5769-5775.

[6] Li Qianmu, Zhang Hong. (2012). Information Security Risk Assessment Technology of Cyberspace: a Review. *Information- an International Interdisciplinary Journal*. 15 (11) 4677-4684.

[7] N. Goranin, A. Cenys. (2009). Genetic algorithm based Internet worm propagation strategy modeling under pressure of countermeasures. *Journal of Engineering Science and Technology Review*. 2, p. 43-47.