International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 2, February 2015, PP 13-17
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Performance Optimization for Short Job in Hadoop

Parmeshwari Sabnis, Prof Chaitali Laulkar

Computer Department, Sinhgad College of Engineering
Pune University, Pune, India
sabnis.parmeshwari@gmail.com, calaulkar.scoe@sinhgad.edu

Abstract: *For solving large data-intensive problem, Hadoop Map Reduce, parallel computing framework is a widely used. To be able to process large-scale datasets, Hadoop focuses on high throughput of data than on job execution performance,. So when there is use of Hadoop Map Reduce to execute short jobs that requires quick responses, causes performance limitation. Short Map Reduce job are usually expected for short execution or quick response times. We optimized the standard Hadoop for greater performance. By optimizing the job initialization and termination stages, changing the task assignment from heartbeat-based pull-model to a push model, and providing an instant message communication mechanism instead of heartbeats, increase in execution speed is achieved.*

Keywords: *Hadoop, Map Reduce, Job execution, Performance optimization, parallel computing.*

1. INTRODUCTION

Map Reduce is a simple solution towards large-scale data processing and analysis. Apache Hadoop is an open-source implementation of GFS and Map Reduce. Hadoop's Map Reduce framework consists of a job scheduler (Job Tracker) running on the master node and a task manager (Task Tracker) is running on each slave node. Job Tracker is the only master control, which can run on any computer in the cluster for scheduling and managing other Task Trackers, allocating Map task and Reduce task to free Task Trackers for parallel running and monitoring the condition of the tasks. There can be more than one Task Tracker. Task Tracker is in charge of the implementation of the tasks. It must run on Data Node, which means that Data Node is not only a data storage node, but also a computing node. If a Task Tracker's task fails, Job Tracker will allocate the task to one of other free Task Trackers, and rerunning.

Map Reduce model the computing process into two functions, Map function and Reduce function. Map function accepts a key-value pairs set as input, and outputs one or more intermediate state key-value pairs set. When a job is submitted to the Map Reduce framework, Map Reduce will divide it into several Map tasks and assign them to different nodes for running. Every Map task only deals with a part of the input data.

For improving the performance of Map Reduce due to above reason, many performance optimization methods are introduced. Optimization can be performed considering any parameter of Map Reduce. Such as Map Reduce parses the data file iteratively and line by line, programming application programs with high efficiency under this circumstance is a way to optimization. Performance of Map Reduce can be improved considering parameters such as avoid unnecessary Reduce tasks pull in external file, add a Combiner to Job.[6][15] There are over 190 configuration parameters in current Hadoop system. How to adjust these parameters so that jobs can run as fast as possible is also a kind of optimization idea. The fact that Hadoop configuration based on cluster hardware information and the number of nodes can greatly improve the performance of Hadoop cluster has been proved. For this optimizing the job scheduling algorithm can be done. The scheduler is a pluggable module in Hadoop, and users can design their own dispatchers according to their actual application requirements.

Big challenge is to how to minimize the cost of data transmission for cloud user. Map-Reduce-Merge is a new model that adds a Merge phase after Reduce phase that combines two reduced outputs from two different Map Reduce jobs into one, Map-Join-Reduce[7] adds Join stage before

Reduce stage. T. Nykiel proposed MRShare[14] is a sharing framework that transforms a batch of queries into a new batch that can be executed more efficiently by merging jobs into groups. Further it evaluates each group as a single query. For iterative problems Map Reduce need lots of input-outputs and unnecessary computations while solving it. Twister proposed by J. Ekanayake is an enhanced Map Reduce runtime that supports iterative Map Reduce computations efficiently, which adds an extra Combine stage after Reduce stage, which results in data output from combine stage which results into next iteration's Map stage. It avoids instantiating workers repeatedly and previously instantiated workers are reused for the next iteration with different inputs. HOP is a modified version of Map Reduce framework that helps users to get returns from a job as it is being computed. D. Jiang et al [11] found that the merge sort in Map Reduce costs lots of I/Os and seriously affects the performance of Map Reduce.

Users usually expect short execution or quick response time from a short Map Reduce job. To be able to process large-scale datasets, the fundamental design of the standard Hadoop places more emphasis on high throughput of data than on job execution performance. This causes performance limitation when we use Hadoop Map Reduce to execute short jobs that requires quick responses. To provide SQL like queries or analysis, some query systems are available, such as Google's Sawzall [10], Facebook's Hive and Yahoo!'s Pig. Obviously, these systems are very sensitive to the execution time of underlying Map Reduce jobs. Therefore, reducing the executing time of short jobs is very important to these types of applications. In order to speed up the execution of short jobs, optimization methods are required to improve the execution performance of Map Reduce jobs. For comparison of previous working of Map Reduce and this one, this system need to be tested on an application. For this K-means clustering algorithm would be considered. K-Means Clustering is one technique used to provide a structure to unstructured data so that valuable information can be extracted. This document discusses the implementation of the K-Means Clustering Algorithm over a distributed environment using ApacheTM Hadoop.

Objective is to reduce the time cost during the initialization and termination stages of a job by removing the constant time cost of 4 heartbeats for its setup and cleanup tasks. For the second optimization, instead of using the heartbeat-based pull-model task assignment, design and implement a push-model task assignment mechanism. For the third optimization, design and implement an instant message communication mechanism for events notification between the Job Tracker and Task Trackers to separate the message communication from heartbeats. That are discussed in next section.

2. METHODS

2.1 Removal of Job Setup and Clean Setup

As presented in the state transition of a job, a setup task should be scheduled first. In brief, the task is processed as follows:

2.1.1 Launch Job Setup Task

Through a heartbeat, the Job Tracker discovers a Task Tracker with free map/reduce slot which can accept a new task, and then the Job Tracker schedules a task to this Task Tracker.

2.1.2 Job Setup Task Completed

The Task Tracker processes the task, and then reports information of the task to the Job Tracker. The two steps described above will take two heartbeats (at least 6 seconds as a heartbeat interval is at least 3 seconds). Similarly, a cleanup task must be scheduled after all map/reduce tasks are completed and will take another 2 heartbeats with at least 6 seconds. As a result, the setup and cleanup tasks will take at least 12 seconds total. For a short job which runs only in a couple of minutes, these two special tasks may take around 10% or more of the total execution time. If one can cut down the fixed time cost of 4 heartbeats for a short job then that will be a noticeable performance improvement for a job execution. By taking a closer look at the implementation of the setup and cleanup tasks, it is found that modification to these two tasks to remove the time cost of the 4 heartbeats can be done.

As discussed above the job setup task simply makes a temp directory for data output, and the job cleanup task deletes the directory. The actual time cost of these two tasks is very small. Thus,

instead of sending message to the Task Trackers to launch the job setup/cleanup task by a heartbeat, immediately execute the job setup/cleanup task in the Job Tracker. That means, when the Job Tracker initializes a job, the setup task of the job will be immediately executed in the Job Tracker. After all map or reduce tasks of the job are completed, a cleanup task of the job will be immediately executed in the Job Tracker as well. So a new version of Hadoop Map Reduce from the standard Hadoop framework is proposed as follows:

- Add the methods setupJob() and cleanupJob() to JobInProgress , the method setupJob() implements what the method runJobSetupTask() in the Task class does and similarly the method cleanupJob() implements what the method runJobCleanup() does in the Task class.
- Calls the method setupJob() from "JobInProgress.initTask()" and then alters the state of job to the RUNNING state.

Calls the method cleanupJob() from "JobInProgress.completedTask()" when all map/reduce tasks have been completed.

2.2 Push Task Assignment

Each Task Tracker periodically sends information to the Job Tracker and performs the pull-model task requests, and the Job Tracker responds as shown in fig. Usually it is refer to the pull-model heartbeat communication mechanism. With the heartbeat commun ication, the Task Trackers report node information to the Job Tracker and then the Job Tracker issues control commands to the Task Trackers. To some extent, the pull-model heartbeat communication mechanism can help prevent the Job Tracker from being overwhelmed. But it comes with a heavy time cost. The Job Tracker has to wait for the Task Trackers to request tasks passively, and as a result, there will be a delay between submitting a job and scheduling the job due to the heartbeat interval. Important information cannot be immediately reported from theTask Trackers to the Job Tracker and this delays the task schedule, further increasing the time cost of job execution and decreasing the utilization efficiency of computing resources.

2.3 Separate Heartbeat and Job/Task Control Message

The Job Tracker and Task Trackers perform the message communication with each other by heartbeats. The content of a heartbeat includes information of the Task Tracker, and task state, etc. To improve the communication performance, separate the job/tasks control message communication from heartbeats and provide an instant message communication mechanism as shown in Figure 4. In this new mechanism, when important events such as task completion happen, the information will be send to the Job Tracker immediately. For all job/tasks scheduling events, the instant message communication is used, but for those cluster management events that are not that performance sensitive still the heartbeat communication mechanism is used. [1]

To test the performance an application based on K-means algorithm is developed [5]. The first step in designing the Map Reduce routines for K-means is to define and handle the input and output of the implementation. The input is given as a <key,value> pair , where 'key' is the cluster center and 'value' is the serializable implementation of vector in the data set. The prerequisite to implement the Map and Reduce routines is to have two file one that houses the clusters with their centroids and the other that houses the vectors to be clustered. Data set for K-means clustering can be according to user's need.

3. RESULTS

The performance for optimized version of the Hadoop Map Reduce framework is compared to the standard Hadoop. During the execution of a job, the state of the slot of each Task Tracker at every second is been recorded. After applying the optimization of dismissing setup/cleanup tasks, the setup and cleanup time costs are noticeably reduced. It is obvious that, while cluster executing a job, the slots are used at a higher level of utilization, in both map phase and reduce phase. In Figure 1, the horizontal ordinate represents queries with different lengths of DNA sequences and the vertical one the time costs. Comparison to the standard Hadoop, our optimized Hadoop reduces the time cost on an average.

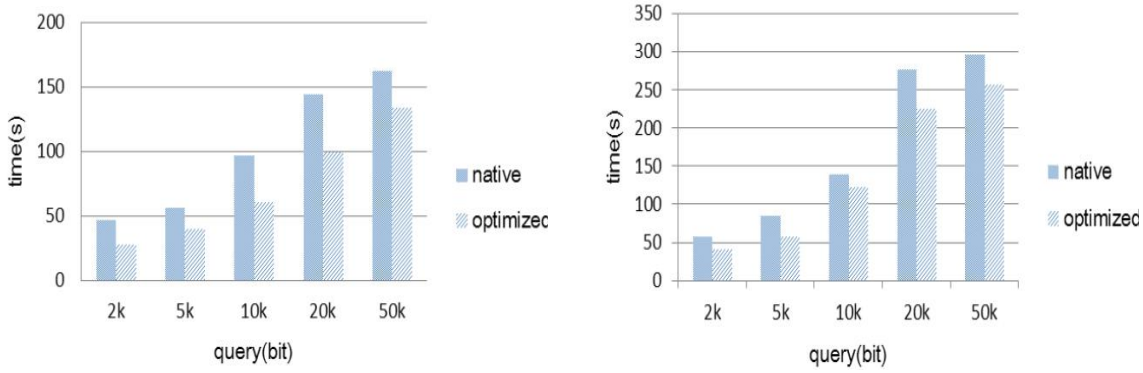

Fig1. *Time cost comparison of the standard and optimized Hadoop*

4. CONCLUSION

Hadoop and its application are more and more widespread. Though Hadoop shows good performance in dealing with large data sets concurrently, there are still some shortcomings. This paper describes the working of Map Reduce and analyzes existing problems of Hadoop data processing platform, and gives suggestions of Hadoop cluster optimization. Future work can include stability test of this optimized version of Hadoop and make more test on variety of benchmark application and dataset for further improvement.

ACKNOWLEDGMENT

We take this opportunity to express my deep sense of gratitude towards those, who have helped us in various ways, for preparing this paper. We would like to thank the reviewers of this paper for their constructive comments.

REFERENCES

[1] Jinshuang Yan, Xiaoliang Yang, Rong Gu, Chunfeng Yuan, and Yihua Huang," Performance Optimization for Short Map Reduce Job Execution in Hadoop", *2012 Second International Conference on Cloud and Green Computing*

[2] Changqing Ji, Yu Li, Wenming Qiu, Uchechukwu Awada, Keqiu Li," Big Data Processing in Cloud Computing Environments", *2012 International Symposium on Pervasive Systems, Algorithms and Networks.*

[3] Benjamin Heintz, Chenyu Wang, Abhishek Chandra, and Jon Weissman,"Cross-Phase Optimization in Map Reduce ", *2013 IEEE International Conference on Cloud Engineering*.

[4] XiaohongZhang, GuoweiWang, ZijingYang, YangDing," A Two-phase Execution Engine of Reduce Tasks In Hadoop Map Reduce ", *2012 International Conference on Systems and Informatics (ICSAI 2012).*

[5] Prajesh P Anchalia, Anjan K Koundinya and Srinath N K, "Map Reduce Design of K-Means Clustering Algorithm ",*2013 IEEE*

[6] Huang Lu, Hu Ting-ting and Chen Hai-shan, "Research on Hadoop Cloud Computing Model and its Applications", *2012 Third International Conference on Networking and Distributed Computing*

[7] H. Yang, A. Dasdan, R. Hsiao, and D. Parker, "Map-reduce-merge: simplified relational data processing on large clusters," *in Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007, pp. 1029–1040.*

[8] K. Shvachko, K. Hairong, S. Radia et al., "The Hadoop Distributed File System."pp.1-10.

[9] Songchang Jin,Shuqiang Yang,Yan Jia, "Optimization of Task Assignment Strategy for Map-Reduce", *2012 2nd International Conference on Computer Science and Network Technology.*

[10] R. Pike, S. Dorward, R. Griesemer, S. Quinlan. "Interpreting the Data: Parallel Analysis with Sawzall," *Scientific Programming Journal, vol. 13, no. 4, Oct. 2005, pp. 227-298.*

[11] D. Jiang, B. Ooi, L. Shi, and S. Wu, "The performance of Map Reduce: An in-depth study," *Proceedings* of the VLDB Endowment, vol. 3, no. 1-2, pp. 472–483, 2010.

[12] Shubin Zhang, Jizhong Han, Zhiyong Liu, Kai Wang, Shengzhong Feng. Accelerating Map Reduce with Distributed Memory Cache. *IEEE, 2009 15th International Conference on Parallel and Distributed Systems. 2009.*

[13] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas, "Mrshare: Sharing across multiple queries in Map Reduce," *Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 494–505, 2010.*

[14] Xin Daxin, Liu Fei. Research on optimization techniques for Hadoop cluster performance [J]. *Computer Knowledge and Technology, 2011,8(7):5484~5486*