Design of Elastic Hadoop Supporting Dynamic Scaling of the Cluster

Wooseok Ryu
Dept. of Healthcare Management
Catholic University of Pusan
Busan, Republic of Korea
e-mail: wsryu@cup.ac.kr

***Abstract*—This paper discusses the problem of node management in the Hadoop cluster and presents a mechanism for managing the Hadoop cluster more elastically. The proposed mechanism supports instant removal of a slave node from the cluster and reconnection to the cluster. Using this, the cluster can be managed more elastically because slave nodes no longer need to be dedicated to the cluster. The experimental results show that the proposed mechanism can process 5 times faster than when a slave node is arbitrarily stopped.**

Keywords-Hadoop; cluster management; scalability.

I. INTRODUCTION

Big data computing is one of key issues in many areas of business domains that wish to discover breaking knowledge from massive data [1]. Apache Hadoop is the most popular open source platform that contributes to broadening the scope of big data analysis. Its distributed processing capability can be extended to thousands of nodes because it supports real-time up-scaling of the cluster [2].

However, we found that Hadoop lacks real-time down-scaling of the cluster [3]. This causes serious problems for small business domains that want to configure a Hadoop cluster with limited resources. The reason is that it would be a financial burden for small businesses to construct a cluster using a large number of dedicated systems. If dynamic up/down scaling of Hadoop is provided, the cluster can be configured more economically by using existing business computers only when necessary. Although cloud computing can be considered as an alternative, some domains, such as small-and-medium sized hospitals do not agree to it due to security reasons.

This paper presents an implementation-level mechanism to manage the Hadoop cluster more elastically by removing nodes from the cluster and adding them again to the cluster in an instant manner. This makes it possible to maintain an elastic Hadoop cluster including existing computers, which means that these computers can be used for analysis or at work in turn, depending on the circumstances at that time. The main idea of this work is initiated and partly implemented by our previous works [3][4]. This paper improves the previous studies through detailed implementation and experiments.

This paper first analyzes the Hadoop architecture with problem statements in Section 2. In Section 3, this paper presents a design and implementation of the elastic node management in Hadoop. Experimental studies are discussed in Section 4, followed by the conclusion.

II. PROBLEM STATEMENT

A Hadoop cluster consists of one master node and a set of slave nodes. The main components of the Hadoop are the Hadoop Distributed File System (HDFS) and the MapReduce framework. The former is a filesystem to store big data in a distributed manner. The latter is to process user requests on big data in parallel. Currently, the MapReduce framework is controlled by Yarn, which is a new framework for job scheduling and resource management [5]. The software architecture of Hadoop is depicted in Figure 1.

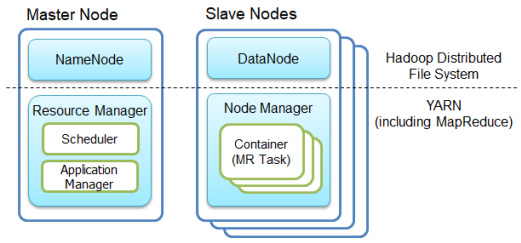

Figure 1. Software architecture of Hadoop.

If a certain slave node needs to be stopped, all the processes inside the node should of course be closed. The first problem is that the processes are handled separately even though they are tightly related to each other. The second problem is that there is no way to stop the *DataNode* process instantly. The existing decommissioning mechanism of HDFS includes moving of data blocks to other live nodes, which cannot be done in a short period of time.

III. SYSTEM DESIGN AND IMPLEMENTATION

In our implementation, we designed a single, unified interface for the Hadoop cluster, which immediately pauses or resumes all the server processes of a slave node. When a pause of a certain slave node is requested, the interface lets *NameNode* and *Resource Manager* of the master node finish the execution of related server processes running on the slave node and delegate task executions to other slave nodes. If a resumption of the node is requested, the master node automatically initiates server processes on the slave node without additional user control.

We implemented the mechanism in the Apache Hadoop version 2.7.4. We modified some source codes to support

 ISBN: 978-1-61208-631-6

new properties and pausing/resuming procedures of HDFS and Yarn in addition to the new interface. Figure 2 shows the detailed mechanism when the cluster needs to be down-scaled. In step 1 in the figure, user requests for a pause by calling the shell command "*NodeManage.sh pause slavenode1*". This adds the node descriptor in a configuration file specified by a new server property named *dfs.host.pause* specified by *yarn.resourcemanager.nodes.pause-path*, followed by sending a *refreshNode* command to the *Resource Manager* and *NameNode*. We consider the state of the node in the configuration file as *Paused* [3][4]. The *Resource Manager* reads the configuration file and decommissions the *Node Manager* process on the specified slave node by terminating the related daemon, marks the node as unusable, and reschedules tasks to other nodes as described in steps 2 and 3. In steps 4 and 5, the *NameNode* reads the file and terminates the *DataNode* daemon when receiving a heartbeat message. This does not decommission the *DataNode*, which demands extra time for moving data blocks, which cannot be done immediately [4].

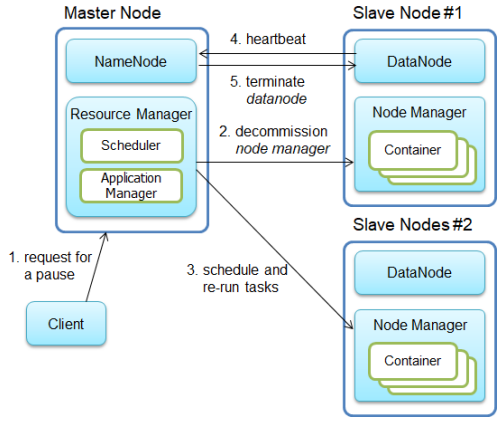

Figure 2. Processing flow for pausing a slave node.

We implemented that the resumption of the paused node can be done by calling a command "*NodeManage.sh resume slavenode1*", which initiates the *DataNode* and *Node Manager* processes in consecutive order. Once the node is resumed, it can store data blocks and execute job tasks immediately without any further work.

IV. Experimental Results

To verify feasibility of the proposed mechanism, we built a small Hadoop cluster consisting of one master node and four slave nodes. Each node is equipped with a 2-core Pentium processor and 4GB main memory. Ubuntu 14 is installed as an operating system. All the nodes are connected to each other with a 1 Giga-bit Ethernet switch.

Figure 3 shows the comparison of processing times among three evaluation cases when executing a *wordcount* program with two text datasets of which sizes are 1Gbyte and 2Gbyte, respectively. When one slave node is paused while the program is running (marked as *Pause*), its processing time was 10~20% slower than when all slave nodes are running (marked as *Normal*). If server processes of one slave node are killed during the execution (marked as *Kill*), its processing time was more than 6 times slower than that of *Normal*. The reason is that the *Resource Manager* has to wait for a timeout, 10 minutes by default, when a slave node is not reachable.

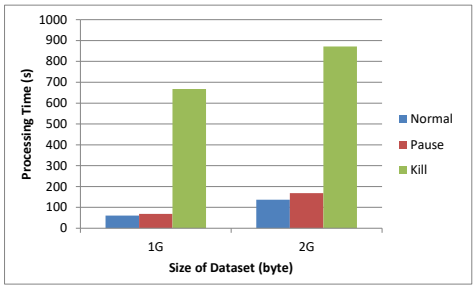

Figure 3. Performance comparison of the proposed implementation

The result shows that the proposed implementation can speed up more than 5 times faster than when the processes are killed arbitrarily in the Apache Hadoop with the default configuration.

V. Conclusion

This paper discussed problems of dynamic node management in the Hadoop cluster and proposed a new mechanism to manage slave nodes more elastically. Implementation of the proposed mechanism is also discussed, along with the experiment. The experimental results show that when a slave node is being stopped, the proposed mechanism can speed up more than 5 times compared with the Apache Hadoop. The main contribution of this paper is to provide the empirical implementation of the proposed mechanism. More comprehensive experimental studies need to be performed as future works.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2016R1C1B1012364).

References

[1] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, "The anatomy of big data computing," Software: Practice and Experience, vol. 46, no. 1, pp.79–105, 2016.

[2] W. K. Lai, Y. U. Chen, and T. Y. Wu, "Towards a framework for large-scale multimedia data storage and processing on Hadoop platform," The Journal of Supercomputing, vol. 68, no. 1, pp. 488–507, 2014.

[3] W. Ryu, "Flexible management of data nodes for Hadoop Distributed File System," The Third International Conference on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2017) IARIA, Apr. 2017, pp. 1–2, ISBN: 978-1-61208-070-3.

[4] W. Ryu, "Implementation of dynamic node management in Hadoop cluster," International Conference on Electronics, Information, and Communication (ICEIC 2018), Jan. 2018, pp. 814–815.

[5] V. K. Vavilapalli et al. "Apache Hadoop Yarn: Yet another Resource Negotiator," Proc. Symp. Cloud Computing, ACM, Oct. 2013, doi:10.1145/2523616.2523633.

 ISBN: 978-1-61208-631-6