e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

EFFICIENT PROCESSING OF MAPREDUCE JOBS USING SCALABLE HADOOP CLUSTERS

Woo S. Ryu[*1], **Tso H. Chen**[*2]

[*1]Department Of Management, Catholic University Of Pusan, Pusan, South Korea.

[*2]Deapartment Of Computer Science, Shanghai Jian Qiao University, Shanghai, China.

ABSTRACT

This paper studies a method for estimating the scale of a Hadoop cluster to process big data as a cost-effective manner. In the case of medical institutions, demands for cloud-based big data analysis are increasing as medical records can be stored outside the hospital. This paper first analyze the Amazon EMR framework, which is one of the popular cloud-based big data framework. Then, this paper presents a efficiency model for scaling the Hadoop cluster to execute a Mapreduce application more cost-effectively. This paper also analyzes the factors that influence the execution of the Mapreduce application by performing several experiments under various conditions. The cost efficiency of the analysis of the big data can be increased by setting the scale of cluster with the most efficient processing time compared to the operational cost.

Keywords: Hadoop, Cluster, Scalability, Mapreduce.

I. INTRODUCTION

In the case of small and medium-sized hospitals under secondary hospitals, which occupy most of Korea, although the need for big data analysis using medical data is constantly being raised, it is difficult to directly build a big data analysis platform due to various problems such as cost. In August 2016, as the enforcement regulations of the Medical Act were amended and standards for facilities and equipment necessary for management and preservation of electronic medical records were implemented, electronic medical records are stored not only inside medical institutions but also outside hospitals such as the cloud[4][5] This became possible. Accordingly, small and medium-sized hospitals became able to analyze big data at low cost in a cloud environment instead of building a big data analysis platform on their own [6].

Major domestic and overseas cloud platforms include Naver Cloud of Naver Business Platform, Amazon Web Services (AWS), Microsoft Azure, and Google Cloud [7][8][9]. Among them, Amazon Web Services, the most widely used cloud platform worldwide, EMR provides a HaaS (Hadoop-as-a-Service) big data analysis framework through the Amazon framework. It has the advantage of conveniently performing big data analysis without construction [9].

In the case of cloud services, the cost of use is determined according to the performance and number of virtual nodes used, and the duration of use. In order to use cloud services cost-effectively, it is necessary to reduce the number of computing resources to be used and the time to use them. When this is applied to the Hadoop platform, it can be defined as estimating the optimal Hadoop cluster size for cost-effectively performing a given MapReduce application. In this paper, the cost-effectiveness model of the MapReduce framework is presented by extending previous studies [10], and the factors affecting the performance are analyzed through experiments.

The structure of this paper is as follows. First, in section 2, EMR we analyze the Amazon framework as a related study, and in section 3, we present an efficiency model for executing MapReduce in Hadoop. section 4 compares and analyzes MapReduce execution performance according to cluster size on the Hadoop platform, and section 5 describes the conclusion and future research.

II. AMAZON EMR ARCHITECTURE

Amazon (Elastic MapReduce) is a Hadoop framework for cost-effectively processing large amounts of data in the Amazon cloud environment [9]. In addition to Hadoop, Amazon supports various open source distributed frameworks such as Apache Spark, HBase, and Hive, so that open source-based big data analysis can be realized as it is in the cloud environment.

Amazon handles MapReduce. The first is how to utilize Amazon (Elastic Compute Cloud). is a re-sizable distributed computing resource, and after allocating multiple instances to the cluster (Hadoop Distributed File System) and installing the MapReduce framework, Hadoop can be operated. However, this method has the characteristic that charging occurs continuously while the is being operated, and the stored data is deleted when the cluster is terminated.

The second method is to use EMRFS (Elastic Map Reduce File System) as data storage instead. It features a wrap-around review of Amazon, a scalable, low-cost, cloud-based data store provided by Apache. Amazon is a storage service that has excellent durability and scalability, but is relatively inexpensive. In other words , the Amazon cluster can be selectively started when analysis is needed while data is stored in the long-term storage and the cluster can be shut down immediately when the analysis is finished. In cases where a large amount of data is continuously stored, such as in small and medium-sized hospitals, but data analysis needs to be performed intermittently as needed, it is better to store data in the node rather than maintaining it in the nodes for a long period of time and transfer it to the Amazon node only when data analysis is required. It is more cost-effective to form a cluster to end the analysis in a short time.

Amazon 's pricing policy is calculated on a per-second basis based on usage time per node. In other words, the fee is set in proportion to the size of the Hadoop cluster and the usage time. For example, the cost of using 10 nodes for 1 hour is the same as the cost of using a cluster with 2 nodes for 5 hours.

III. EFFICIENCY MODEL OF MAPREDUCE EXECUTION

In this study, we propose an efficiency model for calculating the size of the cluster that requires the least cost when executing MapReduce in Hadoop on an arbitrary dataset. In this case, the execution cost is EMR calculated as the product of the number of nodes included in the cluster and the total execution time, similar to Amazon's charging model.

Assuming that the execution times of one map task and one Mreduce task are , respectively, the execution time of t_m, t_rone $T_N$job including Nmap tasks and reduce tasks Rcan be briefly expressed as Equation (1) in Equation (1).

$$T_N = M/N \times t_m + R/N \times t_r$$

However, some assumptions were made in the above formula to simplify the formula. The above formula does not take into account the network load caused by shuffling and sorting map task results, and it is assumed that one task can be executed at the same time for each node. In addition, it is assumed that the reduce task is started after the map tasks of all nodes are finished, but in reality, the reduce task can be started after some of the map tasks are finished, so there is a slight difference.

When executing MapReduce, a task that is mainly processed in parallel is a map task. In the above equation, Mthe value is the number of blocks split according to the size of the input data. At this time, a map task created one per block is automatically created. On the other hand, the value is a value specified by the user, Rand after partitioning the temporary file resulting from the map task according to the value, it is passed as an input to the reduce task. RIf Rthe value is set to 1, Equation (1) can be simplified as Equation (2).

$$T_N = M/N \times t_m + t_r$$

If we C(t)define the tcost of use per second for one computing node as , and the cost of cuse per second as C(t), $c^\times$ ×then satisfies . The total running cost of the cluster using the cluster $C_N$consisting of N nodes for a total of seconds tis calculated as Equation (3) according to the above assumption.

$$C_N = N \times C(T_N)$$

$$C_N = C(N \times T_N)$$

$$C_N = C(N \times M/N \times t_m + N \times t_r)$$

the number of nodes in the cluster increases , α increases Nand parallel execution occurs, so α $T_N$decreases, but the execution cost according to the size of the cluster $C_N$increases Nas α increases. The execution time $T_N$of a job Nbecomes maximum when this is 1, and Nbecomes Mminimum when it exceeds this. On the other hand, the total execution cost $C_N$is Nminimized when is 1 and Nincreases as m increases. In this case, the efficiency of the cluster scale can be said to be the most efficient when the benefit of time reduction versus cost increase is the greatest. NExecution efficiency when running in nodes $E_N$can be defined as Equation (4).

$$E_N = (T_1/T_N)^2/N$$

Combining this with Equation (2), it can be expressed as Equation (5) below.

$$E_N = ((M \times t_m + t_r)/(M/N \times t_m + t_r))^2/N$$

According to Equation (5), theoretically, when the efficiency is high in MapReduce execution, it Mis set to Nbe a multiple of Nthis. That is, Mthe efficiency can be increased when it is set to , etc. when it 2,4,8is 8 . NHowever, in actual MapReduce, it may not be evenly distributed according to the scheduler, and especially if HDFS is used instead of EMRFS, the creation of a map task is affected by the data locality, and the map task is distributed to distributed nodes. are not evenly generated or the cost of transmitting data blocks to the network is added.

IV. EXPERIMENTAL RESULTS

1. Experimental environment setup

In this section, we are going to analyze the change in execution performance of MapReduce that occurs when the number of nodes in a cluster is changed. As an experimental environment, a cluster in which 10 units equipped with a 2-core Pentium processor and 4GB memory were PCconnected through a 1 gigabit switching hub was used. Ubuntu 14 and Hadoop 2.7.4 were used as software for each system, and the default settings of the distribution were used for Hadoop settings such as scheduler and data block size.

To analyze the MapReduce performance according to the cluster size, a Hadoop cluster consisting of various number of slave nodes was constructed. One node out of a total of 10 nodes was set as a master node, and a total of 9 clusters were set up from 1 to 9 slave nodes.

For Hadoop Job to compare performance according to cluster size, the wordcount example program included in the Hadoop distribution was used. For performance measurement, the total time when executing the wordcount example program with the yarn command was measured through the system time. After 5 runs under the same conditions, the average running time was calculated. At this time, the number of map tasks is automatically set as the number of blocks obtained by dividing the data size by the block size (128MB), and the number of reduce tasks is set to the default value of 1.

There are two types of data in this experiment. One is text data whose capacity is increased by using only 300 words to minimize the input of the reduce task. Dataset 1 in Table 1 has 256 MB, 512 MB, 1GB, 2GB five capacities 256MB , and 4GB in the case of, the result data after performing the map task is only 8,916 bytes, so the load of the reduce task is very low. Dataset 2, like Dataset 1, consists of five different sizes of data, but it is text data created with random words to increase the load on the reduce task. In the case of Dataset 2, 256 MB the result data after performing the map task has a larger capacity than the original data, and the reduce 2.5M task load is large enough that the number of input records of the reduce task is large.

Table 1. Dataset for the experiment

Data set	Data size	Map output materialized bytes	Reduce input records	Reduce output records
1	256MB	8,916	608	304
	512MB	17,832	1,216	304
	1 GB	35,664	2,432	304
	2 GB	71,328	4,864	304
	4 GB	147,108	10,032	304
2	256MB	294.8M	2.5M	2.5M

	512MB	589.8M	5.0M	2.5M
	1 GB	1,179.5M	10.0M	2.5M
	2 GB	2,358.8M	20.0M	2.5M
	4 GB	4717.6M	40.1M	2.5M

2. Results

Figure 1 shows the results showing the execution time according to the number of slave nodes when the program is executed using dataset 1. In the case of dataset 1, it can be seen that the execution performance is inversely proportional as the number of slave nodes increases as the load of the reduce task is little. If the data size is 256MB , there are two data blocks and two map tasks are created. There is almost no performance difference between when there is one slave node and when there are 9 slave nodes. Since one node has two cores and two map tasks are processed in parallel, it means that only one slave node can provide sufficient performance. data size is 1GB, there are 8 data blocks and a total of 8 map tasks are created. At this time, in theory, the best map task execution performance should be shown when the number of nodes is 4, but as a result of the experiment, when the number of slave nodes is increased from 1 to 2, the performance is doubled and the performance improvement is insignificant when the number of nodes is continuously increased. As shown in Figure 2, the 1/4 lowest cost was confirmed when the number of slave nodes was equal to the number of map tasks, and when the data size was 2GB, 4GB, the number of slave nodes was 4 and 8, respectively. low can be seen. The difference between the theoretical model and practice is determined to occur because the map task is not evenly distributed among nodes due to data locality [11] [12] according to the replication coefficient of the data block.

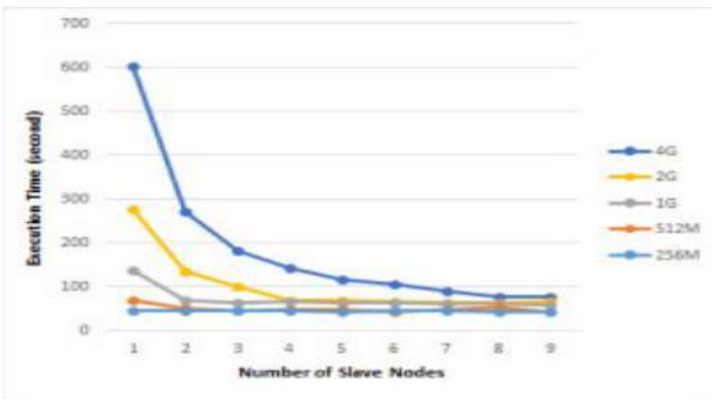

Fig 1: Performance of Mapreduce using dataset1

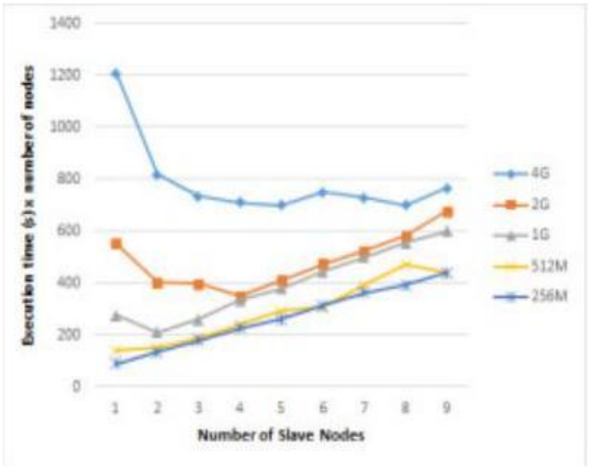

Fig 2: Cost of running MapReduce on Dataset1

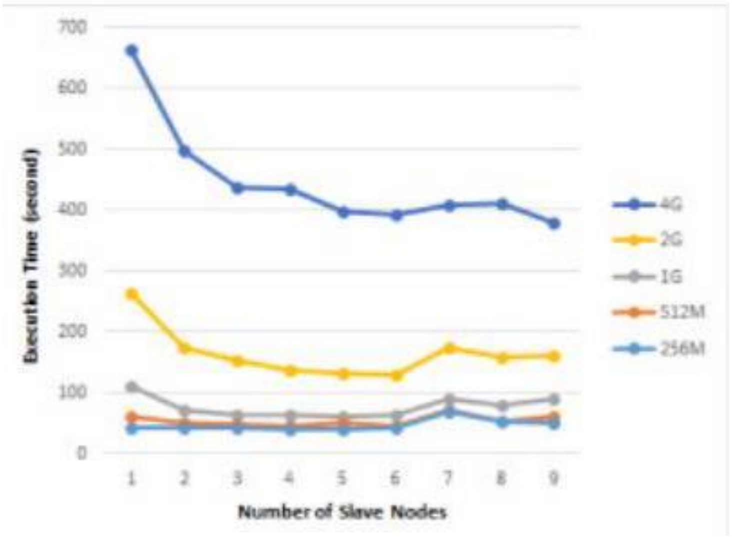

Fig 3: Map reduce execution performance of dataset 2

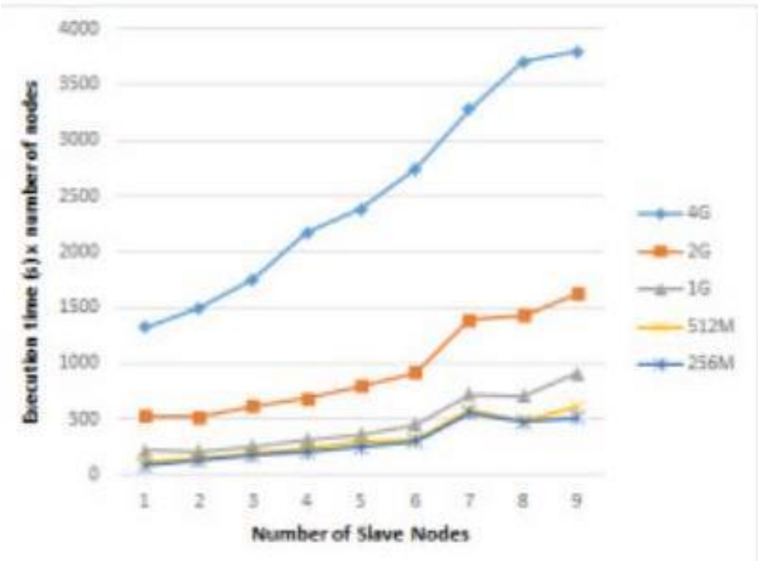

Fig 4: Cost of running MapReduce on dataset2

Figure 3 shows the results of performing the same experiment using dataset 2. When the data size is 256MB, 512MB, 1GB no significant difference is found compared to Dataset 1, but when the number of slave nodes is increased to 6 or more, it can be seen that the execution performance is rather deteriorated. It is judged that the network transmission load has increased due to the increase in the number of nodes. In the case of 2GB data, compared to Dataset 1, the execution time decreases more gradually, and it can be seen that the performance is the highest when the number of nodes is 6. However, according to Figure 4, the cost continues to increase when the number of nodes is 2 or more. In this case, the most efficient cluster size can be limited to when the number of slave nodes is 2. If the data size is 4GB, it can be seen that the benefit of execution time according to parallel execution is very limited, and thus the cost continues to rise. Considering the experimental results, the execution time and cost of MapReduce according to the cluster size are greatly affected by the reduce task. In order to estimate the effective cluster size, the map task to maximize parallelism is focused on the map stage. Development of MapReduce program of When the load of the reduce stage is small, in this experiment, it was confirmed that the cost could be minimized when slave nodes were configured as many as the number of 1/ 4 map tasks.

V. CONCLUSION

In this paper, a cluster efficiency model to minimize the execution cost when executing MapReduce in Hadoop is presented and verified through experiments. The optimal cluster size is an important issue, especially in a cloud environment, and the number of nodes to be used is directly related to the cost of use. In order to estimate the optimal cluster size, it is first proven through experiments that it is necessary to optimize parallel execution in the map task stage to reduce the load in the reduce task stage, which is relatively difficult to perform in parallel. A way to reduce the execution time and cost of MapReduce, which can simultaneously reduce the execution time and cost of MapReduce, was proposed. Through this, as big data analysis is possible more cost-effectively, it is possible to analyze big data such as small and medium-sized hospitals, so the division of redemption data such as small and medium hospitals will be utilized. Big data analysis will expand with future research could it be as a future study, it is to calculate a more precise cost model in the cloud environment by actually applying this experiment to Amazon to evaluate the performance and cost of EMR.

VI. REFERENCES

[1] Y. Ding and K. Kim, "A Customized Tourism System Using Log Data on Hadoop," J. of the Korea Institute of Electronic Communication Sciences, vol. 13, no. 2, Apr. 2018, pp. 397-404.
[2] E. Nazari, M. H. Shahriari, and H. Tabesh, "Big Data Analysis in Healthcare: Apache Hadoop, Apache spark and Apache Flink," Frontiers in Health Informatics, vol. 8, no. 1, 2019, pp. 92-101.
[3] J. Choi, "Utilization value of medical Big Data created in operation of medical information system," J. of the Korea Institute of Electronic Communication Sciences, vol. 10, no. 12, Dec. 2015, pp. 1403-1410.
[4] Y. Ahn and H. Cho, "Hospital System Model for Personalized Medical Service," J. of the Korea Convergence Society, vol. 8, no. 12, Dec. 2017, pp. 77-84.
[5] S. Kim and D. Kim, "The Design and Implementation of the Fire Spot Display System Using s Smart Device," J. of the Korea Institute of Electronic Communication Sciences, vol. 13, no. 6, Dec. 2018, pp. 1287-1292.
[6] M. Lee, "Considerations for the Migration of Electronic Medical Records to Cloud Based Storage," J. of Korean Library and Information Science, vol. 47, no. 1, Mar. 2016, pp. 149-173.
[7] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, Microsoft Azure. Berkeley: Apress, 2015
[8] T. Gunarathne, T. Wu, J. Qiu, and G. Fox, "MapReduce in the Clouds for Science," In Proc. the IEEE Cloud Computing Technology and Science, Indianapolis, USA, 2010, pp. 565-572.
[9] S. Mathew, "Overview of Amazon Web Services," Amazon Whitepapers, Nov. 2014.[10] W. Ryu, "Cost-Effective MapReduce Processing in the Cloud," In Proc. the Conf. on Korea Information and Communication Engineering, vol. 22, no. 2, Oct. 2018, pp.114-115.
[10] A. Sharma and G. Singh, "A Review on Data locality in Hadoop MapReduce," In 2018 Fifth Int. Conf. on Parallel, Distributed and Grid Computing, Solan Himachal Pradesh, India, Dec. 2018, pp. 723-728.
[11] S. Kim, Y. Kim, and W. Kim, "The Design of Method for Efficient Processing of Small Files in the Distributed System based on Hadoop Framework," J. of the Korea Institute of Electronic Communication Sciences, vol. 10, no. 10, Oct. 2015, pp. 1115-1122.
[12] Laines-Hidalgo, José Ignacio, José Armando Muñoz-Sánchez, Lloyd Loza-Müller, and Felipe Vázquez-Flota. "An Update of the Sanguinarine and Benzophenanthridine Alkaloids' Biosynthesis and Their Applications." Molecules 27, no. 4 (2022): 1378.
[13] Maruyama-Inoue, Maiko, Tatsuya Inoue, Shaheeda Mohamed, Yoko Kitajima, Shoko Ikeda, Arisa Ito, and Kazuaki Kadonosono. "Incidence of elevated intraocular pressure after intravitreal injection in Japanese patients with age-related macular degeneration." Scientific Reports11, no. 1 (2021): 1-5.
[14] Sheheeda Manakkadu, Srijan Prasad Joshi, Tom Halverson, and Sourav Dutta. "Top-k User-Based Collaborative Recommendation System Using MapReduce." In 2021 IEEE International Conference on Big Data (Big Data), pp. 4021-4025. IEEE, 2021.
[15] Singh, Balraj, and Harsh K. Verma. "IMSM: An interval migration based approach for skew mitigation in mapreduce." Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science) 14, no. 1 (2021): 71-81.
[16] Kalia, Khushboo, and Neeraj Gupta. "Analysis of hadoop MapReduce scheduling in heterogeneous environment." Ain Shams Engineering Journal12, no. 1 (2021): 1101-1110.
[17] Marzuni, Saeed Mirpour, Abdorreza Savadi, Adel N. Toosi, and Mahmoud Naghibzadeh. "Cross-

MapReduce: Data transfer reduction in geo-distributed MapReduce." Future Generation Computer Systems 115 (2021): 188-200.

[18] Angles, Renzo, Fernanda López-Gallegos, and Rodrigo Paredes. "Power-Law Distributed Graph Generation With MapReduce." IEEE Access 9 (2021): 94405-94415.

[19] Ghattas, Badih, Antoine Pinto, and Sambou Diao. "MapReduce Clustering for Big Data." In 2021 IEEE International Conference on Big Data (Big Data), pp. 5116-5124. IEEE, 2021.