MapReduce Performance Scaling Using Data Prefetching

Jung Lee*, Kyung Tae Kim*, Tso Youn-Chen*
*Sungkyunkwan University College of Information and Communication

--************************--------------------------------

Abstract:

Recently, due to the advent of social networks, bio-computing, and the Internet of Things, more data is being generated than in the existing IT environment, and as a result, research on efficient large-capacity data processing techniques is being conducted. MapReduce is an effective programming model for data-intensive computational applications. A typical MapReduce application includes Hadoop, which is being developed and supported by the Apache Software Foundation. This paper proposes a data prefetching technique and a streaming technique to improve the performance of Hadoop MapReduce. One of the performance issues of Hadoop MapReduce is work delay due to input data transmission in the MapReduce process. In order to minimize this data transfer time, a prefetching thread in charge of data transfer was created separately, unlike the existing MapReduce. As a result, data transmission became possible even during the MapReduce operation of data, reducing the overall data processing time. Even with this prefetching technique, the job waits for the first data transmission due to the characteristics of Hadoop MapReduce. To reduce this waiting time, the streaming technique was used to further reduce the waiting time due to data transmission. Mathematical modeling was performed to measure the performance of the proposed method, and as a result of the performance measurement, it was confirmed that the performance of MapReduce to which the streaming method was additionally applied was improved compared to MapReduce to which only the existing Hadoop MapReduce and prefetching methods were applied.

Keywords — MapReduce, Prefetching, Streaming**.**

--************************--------------------------------

I. INTRODUCTION

Recently, due to the advent of social networks such as bio-computing and the Internet of Things, more data is being generated than in the existing IT environment. As a result, research on efficient large-capacity data processing techniques is being conducted. MapReduce is an effective programming model for data-intensive computational applications. A representative MapReduce application, there is Hadoop, which is being developed and supported by the Apache Software Foundation [1-3]. Due to the nature of Hadoop MapReduce, when processing external data rather than local data in order to utilize the computational resources of all nodes constituting the cluster to the maximum, data transmission should be considered for the efficiency of

MapReduce. A study on the data prefetch technique [5] was conducted to solve the data processing delay during the data transmission period. However, even if the prefetching technique is applied, data processing is impossible during the initial data transmission time, so there is a waiting time for work.

In this paper, we try to improve the job processing performance of Hadoop MapReduce by proposing a prefetching technique that enables data processing and transmission at the same time and a streaming technique that can shorten the data transmission time.

II. PREVIOUS WORKS

A. MapReduce

Hadoop MapReduce consists of two phases: Map and Reduce. As shown in Figure 1, in the map phase, a plurality of map phases divide the input data into splits and process them, and when all map tasks are completed, the map phase ends. Afterward, the results of the map tasks are mixed and sorted, then processed in parallel by the reduced tasks, and the results of the reduced tasks are combined into one output file, ending the entire MapReduce task [4]. In this process, if the node processing the map task does not have the input data split of the map task it is executing, it receives the node's data with the corresponding input file and executes the map task. Since there is no data, data processing waits until data transmission is completed.

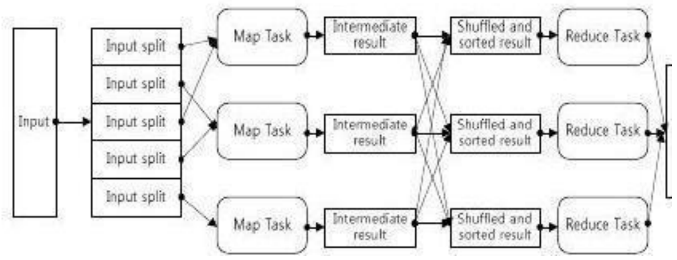

Figure 1. MapReduce job processing flow chart

Recently Manakkadu et al. [8] proposed a scalable MapReduce-based method for collaborative filtering primarily used in e-commerce, online retail, and streaming service providers such as Amazon, and Netflix, etc. In this paper, we proposed data prefetching and streaming methods to improve the scalability of MapReduce jobs.

The input data of the Hadoop MapReduce job is divided into lines and executed in the map task [4], so that the map task can be performed only with partial data rather than the entire input file. This means that the streaming technique that processes data during data transmission can be applied, and if the streaming technique is additionally applied to the existing prefetching technique, the map task can shorten the overall job processing time as shown in Figure 4.

B. Data prefetching

The map task of Hadoop MapReduce requires input data to process the assigned map process. If the input data is in a node other than the local node, the input data is secured through data transmission.

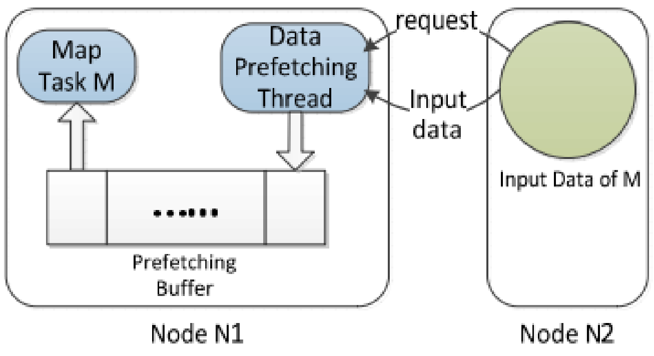

Fig. 2. Map Task with Prefetch Architecture

Since data cannot be processed during this transmission process, the overall processing time is delayed [5].

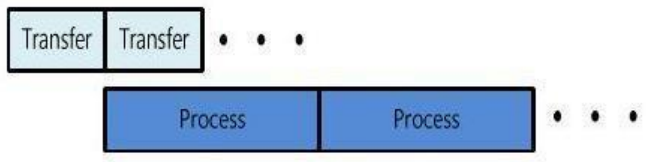

Figure 3. Execution of map task with prefetching applied

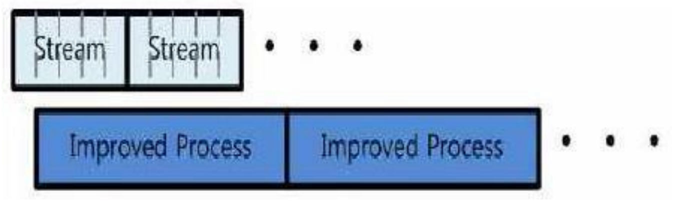

Figure 4: Execution of map task with prefetching and streaming

The core of the prefetching technique is to reduce the overall map task processing time by enabling data transmission and processing simultaneously by processing serially processed data transmission and processing in parallel using a prefetching thread. However, even if such a prefetching technique is applied, during the initial data transmission time, data processing cannot be executed until one data is completely transmitted, so a waiting time occurs.

III. MODEL

In this paper, to improve the performance of Hadoop MapReduce, data transmission, and processing are parallelized based on the prefetching technique, and a streaming technique [6] is applied to minimize the waiting time that occurs during the initial data transmission. suggest The existing prefetching technique uses a prefetching thread to process and transmit data at the same time as shown in Figure 3, but since data processing cannot be performed during the first data transmission, there is a waiting time until the transmission is completed. Figure 4. Execution of map task with prefetch and streaming applied 4. Map task with Prefetch and Streaming

For mathematical modeling for the performance measurement of the proposed system and the existing Hadoop.

Through the above factor values, the existing MapReduce performance of Hadoop can be modeled as follows.

$$t_{hraloop} = \left(\frac{S_{buf}}{V_{tran}} + \frac{S_{buf}}{V_{map}}\right) * \frac{S_{map}}{S_{buf}}$$

Also, the performance of MapReduce with data prefetching is as follows.

$$t_{prefectch} = \frac{S_{iuf}}{V_{tran}} + \frac{S_{iuf}}{V''_{map}} + \left(\frac{S_{map} - S_{iuf}}{m \quad (V'_{map}, V_{tran})}\right)$$

The reason for choosing a small value between the two values in the above modeling is that the data transmission time is faster than the data processing time. V'_{map}, V_{tran}

However, in most cases, the data transmission time is more of a problem than the data processing time, so only the case with a short transmission interval is considered in this paper. Also $V'\ _{map}$, V_{map}

there is no difference in resources except for a small memory space used for the transmission buffer, so the performance difference is negligible. The modeling considering this is as follows.

$$t_{preftecth} = \frac{S_{buf}}{V_{tran}} + \frac{S_{buf}}{V_{mup}} + \left(\frac{S_{map} - S_{buf}}{V_{tran}}\right)$$

the prefetching MapReduce modeling, the $\frac{S_{buf}}{V_{tran}}$

map task is the first data transmission to process the input data. If this part is expressed as transmission to which streaming technique is applied, the waiting time until data processing is the time at which the data that can be processed with the least amount of processing is completed $\frac{S_{streanbuf}}{V_{tran}}$

In addition since the entire input file size does not change even if the streaming technique is applied, the time required to transmit the remaining files other than the first file transmission is modeled by the existing prefetching technique, $\frac{S_{map} - S_{buf}}{V_{tran}}$

, and the file processing speed $\frac{S_{buf}}{V_{map}}$

can also be expressed as . . Therefore, the modeling of the processing time of Hadoop Map Reduce to which the prefetching and streaming techniques proposed in this paper are applied is as follows.

$$t_{inquroeed} = \frac{S_{streambuf}}{V_{tran}} + \frac{S_{buf}}{V_{map}} + \left(\frac{S_{map} - S_{buf}}{V_{tran}}\right)$$

The formula to measure how much performance improved Map Reduce to which prefetching and streaming techniques are applied compared to the existing Hadoop Map Reduce is as follows.

$$\Delta t = \left(\frac{S_{uif}}{V_{tran}} + \frac{S_{uff}}{V_{map}}\right) * \frac{S_{map}}{S_{buf}} - \left(\frac{S_{streambuf}}{V_{tran}} + \frac{S_{suf}}{V_{map}} + \left(\frac{S_{mapp} - S_{suf}}{V_{tran}}\right)\right)$$

value	Case 1	Case 2	Case 3
V_{tran}	1,024	1,024	1,024
V_{mapp}	1,024	2,048	2,048
$S_{streambu}$	1,024	512	10
S_{buf}	1,024	1,024	1,024
S_{map}	102,400	102,400	102,400

Table 2. Values for Compare Original Mapreduce and improved Mapredcue

Using this formula, the factor values for performance comparison with the existing Hadoop MapReduce are set as shown in Table 2. Factor value settings for performance comparison of existing MapReduce and Extended MapReduce.

IV. EXPERIMENTAL RESULTS

value	Case 1	Case 2	Case 3
V_{tran}	1,024	1,024	1,024
V_{map}	2,048	2,048	2,048
$S_{streamb\,uf}$	1,024	512	10
S_{buf}	1,024	1,024	1,024
S_{mapp}	102,400	102,400	102,400

Table 3. Factor value settings for performance comparison of MapReduce with prefetching and Enhanced MapReduce

Figure 5 shows that Hadoop, to which prefetching and streaming techniques are applied, shows better performance compared to the existing Hadoop MapReduce. In addition, it can be seen that as the difference between the map task processing performance and the data transmission performance is not large, the performance improvement by prefetching becomes larger.

Table 3 below shows the settings for comparing the performance of the system to which only the existing prefetching is applied and the system to which the streaming technique is additionally applied.

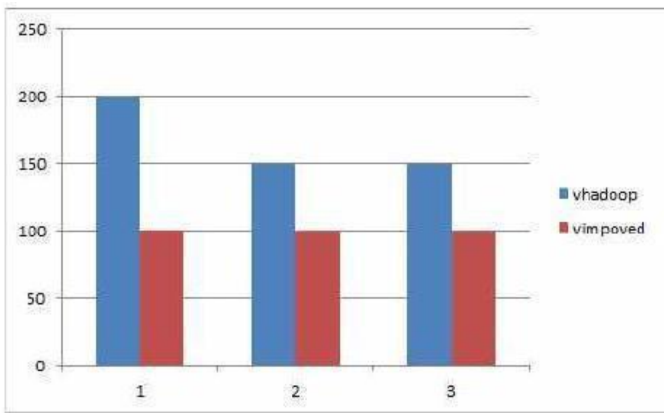

Figure 5. Comparison of performance between the existing MapReduce and the improved MapReduce

When the streaming technique is applied, the smaller the size of the streaming buffer, the shorter the transmission waiting time. The results in Figure 6 show that the performance of MapReduce to which the streaming technique is additionally applied is improved compared to MapReduce to which only prefetching is applied.

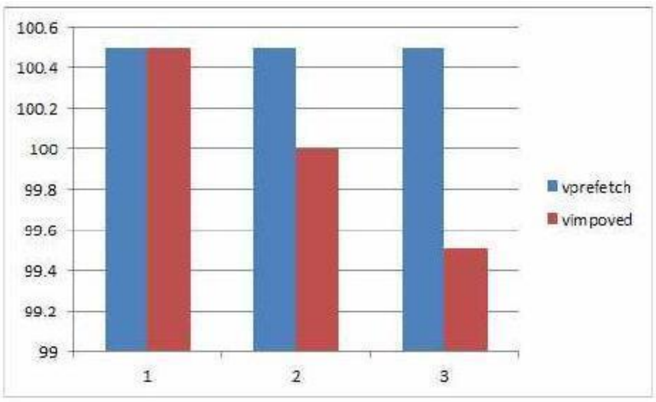

Figure 6. Comparison of performance between MapReduce with prefetching and MapReduce with constant

V. CONCLUSION

In this paper, to solve the performance degradation problem caused by data transmission of Hadoop MapReduce, the prefetching technique and the streaming technique were proposed. By applying the prefetching technique in the map stage, data processing and transmission can be processed in parallel to reduce the data transmission latency and the streaming technique is added to reduce the job processing latency due to the initial data transmission where the prefetching technique cannot be applied. applied. In addition, mathematical modeling was performed to measure the performance of the result, and as a result of the performance measurement, it was confirmed that the performance of MapReduce to which the streaming technique was additionally applied was improved compared to the existing MapReduce to which only Hadoop MapReduce and prefetching techniques were applied.

In order to solve the packet overhead problem caused by the small data transmission size when streaming techniques are applied in the future, we plan to conduct a study on the policy for determining the size of transmission data per packet.

REFERENCES

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Communications of the ACM, vol. 51, no. 1, (2008)

[2] D. Jiang, B. Chin, L. Shi, and S. Wu, "The performance of MapReduce: an in-depth study", Proceedings of the VLDB Endowment. vol. 3, no. 1, (2010).

[3] J. Dean and S. Ghemawat, "Mapreduce: a flexible data processing tool", Communications of the ACM, vol. 53, no. 1, (2010)

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System", IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), (2010) May 3-7: Incline Village, USA.

[5] Tao Gu, Chuang Zuo, Qun Liao, Yulu Yang and Tao Li "Improving MapReduce Performance by Data Prefetching in Heterogeneous or Shared Environments " International Journal of Grid and Distributed Computing Vol.6, No.5 (2013), pp.71-82

[6] Jiadong Wu and Bo Hong, "Improving MapReduce Performance by Streaming Input Data from Multiple Replicas" Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference Vol 1, (2013)

[7] Mitra, Arnab. "On Type D Fuzzy Cellular Automata-Based MapReduce Model in Industry 4.0." In *Industrial Transformation*, pp. 209-222. CRC Press, 2022.

[8] 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.

[9] Daghighi, Amirali, and Jim Q. Chen. "Robustness Comparison of Scheduling Algorithms in MapReduce Framework." In *Intelligent Computing*, pp. 494-508. Springer, Cham, 2022.

[10] Li, Wei. "Link Mining and Topology Fusion of Social Network Nodes Based On MapReduce." In *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, pp. 1066-1069. IEEE, 2022.