IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

In-Memory Processing of Big Data to Improve Processing Rate

[1]Hyup Gu Lee, [1]Yun Kim, [2]Narendra Bhanukiran

[1]Computer Science, Seoul Gangseo Campus of Korea, South Korea,
[2]Information Technology, Amaravathi Institute of Technology, Bangalore, India

Abstract : The amount of data generated due to the development of IT technology is increasing exponentially every year. As an alternative to this, research on distributed systems and in-memory-based big data processing techniques is being actively conducted. The processing performance of existing big data processing techniques processes big data faster as the number of nodes and memory capacity increases. However, the increase in the number of nodes increases the frequency of failures in the big data infrastructure environment, and increases infrastructure management points and infrastructure operation costs. In addition, the increase in memory capacity increases the infrastructure cost for node configuration. Therefore, in this paper, we propose an in-memory-based hybrid big data processing technique to improve big data processing rate. The proposed method adds a combiner step to the distributed system processing method, and at that stage, by applying in-memory-based processing technology, the big data processing time is reduced by approximately compared to the existing distributed system-based big data processing method. In the future, for practical verification of the proposed technique, realistic performance evaluation in a big data infrastructure environment composed of more nodes is required.

***IndexTerms* - In-memory computing, MapReduce, Big Data.**

I. INTRODUCTION

Big data processing techniques must process massive amounts of data faster and more accurately. The initial plan to reflect these requirements was operated by selecting the Scale-Up method to increase the performance of the configured system. however

The distributed system-based big data processing technique and the in-memory-based big data processing performance are closely related to the number of nodes and the performance of the nodes. As the number of nodes and memory capacity applied to the big data processing technique increases, the big data processing performance increases [1, 2].

However, as the number of nodes increases, the frequency of possible failures in the big data infrastructure environment increases, which also increases the cost of infrastructure management points and infrastructure operation. In addition, the increase in memory capacity increases the infrastructure cost for the node configuration required for big data processing [3].

Therefore, in this paper, we propose an in-memory-based hybrid big data processing technique to improve big data processing rate. The proposed technique adds a combiner stage to the main stage of MapReduce, and applies in-memory-based processing technology at that stage to reduce the number of nodes compared to the existing distributed system-based big data processing method, and to reduce the big data processing rate by approximately improve

The structure of this paper is as follows. Chapter 2 examines the existing distributed system-based big data processing techniques and in-memory-based big data processing techniques, and analyzes the requirements. Chapter 3 presents an in-memory-based hybrid big data processing technique to improve the big data processing rate proposed in this paper. In Chapter 4, we analyze the performance of the proposed technique, and finally, in Chapter 5, we present the conclusion.

II. PREVIOUS WORKS

This chapter examines the most commonly used distributed system-based big data processing techniques and in-memory-based big data processing techniques by classifying them according to big data processing methods. In addition, the requirements for the proposed big data processing technique are derived based on the research results.

Distributed system-based big data processing technique

The distributed system-based big data processing technique [4, 5] utilizes the existing distributed system to process a huge amount of data. As a distributed system widely used in big data processing techniques, the big data processing technique using the Hadoop distributed file system is widely used in the big data processing technique using GlusterFS. The big data processing technique using the Hadoop distributed file system processes big data by applying the MapReduce framework. This MapReduce framework plays a role in refining, processing, and analyzing vast amounts of data stored in a distributed file system. [Figure 1] shows the big data processing technique based on the MapReduce framework.

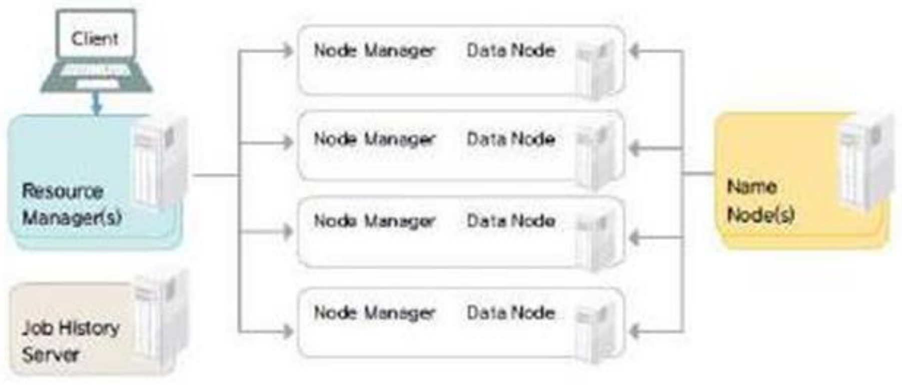


Figure 1. Big data processing technique based on MapReduce framework 1. Mapreduce frameworks based big data processing scheme

The MapReduce framework-based big data processing technique uses the Resource Manager, Job History Server, and Node Manager provided by Hadoop to create data nodes that store vast amounts of data and name nodes that store meta information of those data nodes. The main characteristic of this MapReduce framework is that the big data processing time decreases as the number of data nodes increases. However, as the number of data nodes increases, the infrastructure environment should be properly configured because the probability of failure increases.

The big data processing technique using GlusterFS, consists of a distributed file system similar to the big data processing technique using the Hadoop distributed file system described above. Unlike the Hadoop distributed file system, GlusterFS does not separately create metadata for nodes where data is stored, and it is recommended to build it through a separate system or technology. Due to this, problems such as data loss due to loss of node information due to name node failure are likely to occur in the Hadoop distributed file system, but GlusterFS does not occur frequently. [Figure 2] shows the GlusterFS-based big data processing technique.

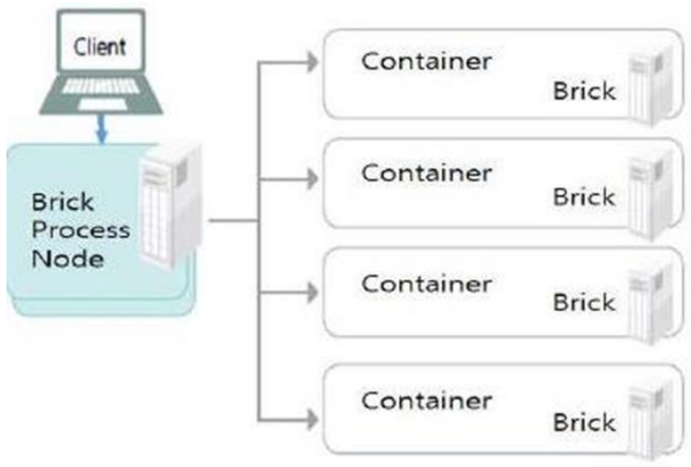


Figure 2. GlusterFS-based big data processing technique

Fig. 2. GlusterFS based big data processing scheme

In [Figure 2], the GlusterFS-based big data processing technique manages the data stored through the separately configured Brick Process Node through containers that play the same role as the data nodes of Hadoop. The saved data is saved in units of bricks defined by GlusterFS.

In-memory-based big data processing

The in-memory-based big data processing technique [2, 7] is a technique for processing a large amount of data by utilizing the memory of distributed and stored nodes. A commonly used method in in-memory-based big data processing techniques is to process vast amounts of data using spark.

Spark is a technology developed as the next step in MapReduce and provides a higher level API than MapReduce. By providing API, Spark enables logic-focused big data programming. Spark's big data processing performance operates based on the memory of the nodes where data is stored. Because of this, it is about 100 times faster than the distributed system-based big data processing method described above. Due to this processing performance, Spark has a lower waiting time for big data processing compared to the existing distributed system-based big data processing technique, so it performs near real-time processing of vast amounts of data. [Figure 3] shows the in-memory-based big data processing technique.

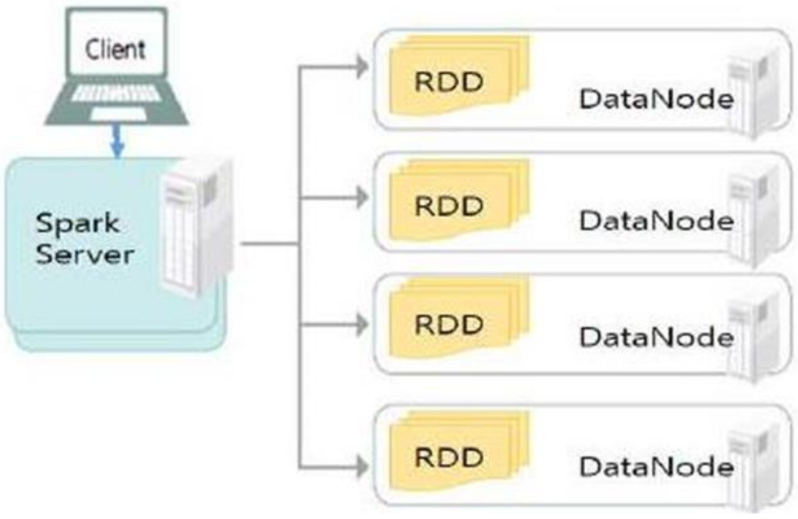


Figure 3. In-memory-based big data processing technique

In [Figure 3], the in-memory-based big data processing technique processes data through the memory of data nodes. The memory processing method is defined by Spark whenever stored data is transformed and processed (Resilient) .Distributed Data is stored in memory through data structure and data is processed.

III. Mathematical Model

In this section, the problems of the distributed system-based big data processing technique and the in-memory-based big data processing technique described above are derived and the requirements are analyzed.

First, the big data processing performance of the distributed system-based big data processing technique decreases if there are not many nodes. The number of nodes in the distributed system-based big data processing technique is closely related to the big data processing speed, and as the number of nodes increases, the processing performance increases. However, the increase in the number of nodes increases the frequency of possible failures in the big data infrastructure environment and increases the infrastructure management points, increasing the cost required for infrastructure operation.

probability of occurrence of a failure , the total number of nodes is distributed and stored in 40s, and triplicate. If it is defined as the number of nodes with failure (ErrNode), the probability of failure is as Equation (1).

$$P = ErrNode\left(\frac{e_1}{n}\right)\left(\frac{e_1-1}{n-1}\right)\left(\frac{e_1-2}{n-2}\right) \quad (1)$$

If the number of failed nodes in Equation (1) is set to 10, it can be predicted that failure may occur with a weak probability.

Equation (2) represents the number of blocks to be lost (ErrBlock).

$$ErrBlock = P\frac{DefaultBlockSize}{DataStroreSize} \qquad (2)$$

If the size of the data stored in Equation (2) is defined as , and the basic block size stored in the data node is defined as , the number of lost blocks is about 3,906. The loss of these stored blocks is very fatal in a system where data reliability is important.

Therefore, the big data processing technique should reduce the frequency of failures and enable faster big data processing.

Second, the infrastructure construction cost of the in-memory-based big data processing technique is high. The in-memory-based big data processing technique processes data using the memory of the nodes where data is stored. For this characteristic, normal big data processing is possible only when the node is configured with a memory capacity larger than the data capacity to be processed. Big data processing for large files is expensive to apply in-memory-based big data processing techniques.

Therefore, the big data processing technique requires a hybrid big data processing technique in which the in-memory based big data processing technique and the distributed system based big data processing techniques are fused.

IV. Proposed method

The proposed in-memory-based hybrid big data processing technique processes big data in a hybrid method in which the in-memory-based big data processing technique and the distributed system-based big data processing technique are fused according to the requirements analyzed above. [Figure 4] shows the processing process of the proposed big data processing technique.

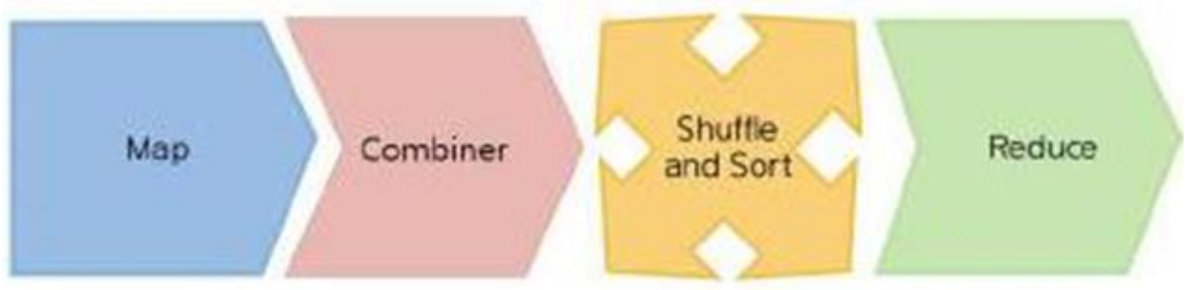


Figure 4. Processing process of the proposed big data processing technique

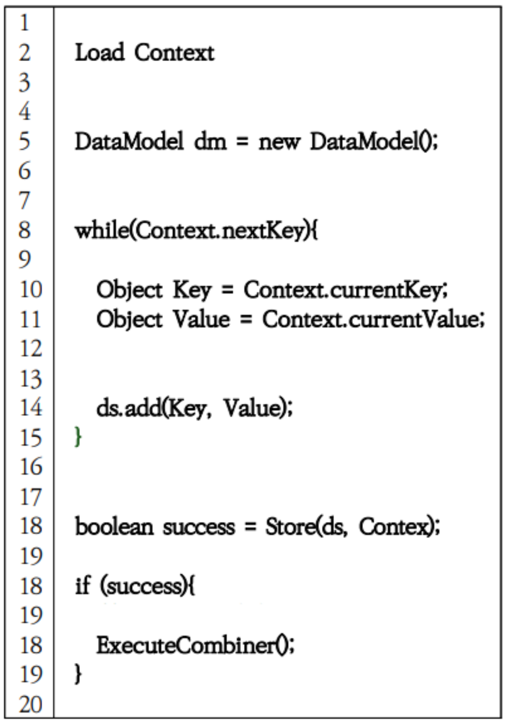
Fig. 4. The process of big data processing scheme The proposed big data processing scheme utilizes Hadoop, and the distributed system uses the Hadoop distributed file system and processes big data through MapReduce. The process of the proposed big data processing technique proceeds with the main steps of the MapReduce framework. The main stages of MapReduce are Map, Combiner, Shuffle and Sort, and Reduce. Map stage performs the role of primary purification and processing of data to be analyzed and processed according to the purpose. The combiner stage is the stage proposed in this paper, and it processes the data similar to the reduce stage before sending the massive amount of data processed in the Map stage of MapReduce, which is executed earlier, to the Shuffle and Sort stage. The Shuffle and Sort step is automatically performed by the MapReduce framework, and the data transmitted from the Map step or Combiner step is collected, grouped, and then sorted. The Reduce stage performs processing and analysis based on the data received from the previously executed Shuffle and Sort stage.

The proposed big data processing technique applies in-memory-based processing technology to the combiner stage among the main stages to perform faster big data processing. [Figure 5] shows the pseudo code for data processing in the combiner stage.

Combiner stage loads the Context object that plays the same role as MapReduce's public data storage first in order to increase the memory of the data received from the Map stage. The Context object is an object in which all the data processed in the Map stage are stored, and the data model for storing in the memory is composed of key and value types. The reason why the data model is configured in the form of key and value is that the basic data model that delivers data in MapReduce is stored in the form of key and value. [Figure 6] shows the data model in the form of keys and values stored in memory.

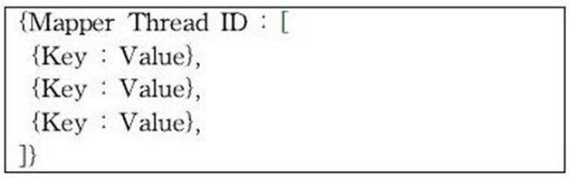
In [Figure 6], the data model is stored as an array in the form of keys and values for each thread of the Mapper object where the Map function is executed, and the data structure uses JSON suitable for data processing speed.

The combiner stage creates a data model object to store the data received from the Mapper object, and then starts reading the data stored in the Context object. The read data is saved in the created data model. The created data model is saved in the memory when all the data received from the Mapper object are saved. When the memory storage is completed, the combiner stage runs the combiner that plays the same role as a mini reducer and ends.



```
Load Context


DataModel dm = new DataModel();


while(Context.nextKey){

   Object Key = Context.currentKey;
   Object Value = Context.currentValue;


   ds.add(Key, Value);
}


boolean success = Store(ds, Contex);

if (success){

   ExecuteCombiner();
}
```

Figure 5. Pseudocode for data processing in Combiner stage



```
{Mapper Thread ID : [
 {Key : Value},
 {Key : Value},
 {Key : Value},
]}
```

Figure 6. Key value type data model stored in memory

V. Experimental results

In this chapter, performance evaluation of the proposed in-memory-based hybrid big data processing technique is performed. The number of nodes for the distributed system-based big data processing scheme (Distributed Big Data Processing Scheme, DBPS) and the proposed big data processing scheme (Hybrid Big Data Processing Scheme, HBPS) is based on the requirements analyzed in the previous related research. We compare and analyze the size of data lost according to the increase in big data processing time and the probability of failure.

The big data infrastructure environment for performance evaluation is verified by establishing a fully distributed mode based on Hadoop. As the version of Hadoop used for performance evaluation, the officially distributed version is used. Nodes configured in Hadoop fully distributed mode consist of at least 5 to 12 nodes. A total of two namenodes and one secondary namenode for storing meta-information of analyzed data are configured one by one. For performance evaluation, data nodes increase from a minimum of 3 data nodes to a maximum of 10 data nodes for performance evaluation. The analysis target for big data processing is Apache web server logs of about capacity, and data from January to December 2018. During the analysis period for big data processing, extraction and analysis are performed on the last 31 days of data among 12 months of data, and the MapReduce job is a job that analyzes the number of abnormal URL accesses.

Big Data Analysis Processing time analysis increases the number of data nodes from at least 3 to 10, measures the execution time of MapReduce jobs, and compares and analyzes the results. The analysis target is defined as data of the last 31 days among Apache web logs. <Figure 7> and <Table 2> show the results of comparison analysis of big data processing time between the proposed technique and the comparison target.

According to the results, as the number of data nodes increased compared to , processed data faster. Big data processing speed is higher in HBPS than DBPS in 3 data nodes. Data was processed quickly. In particular, the big data processing time of HBPS decreased by approximately as the number of nodes increased. The reason for this result is that the combiner step applied to , similarly to reduce, performs the reduce process based on memory after the Map step. Therefore , as the size of the big data infrastructure environment increases, it becomes possible to operate with a smaller number of data nodes compared to that, and the infrastructure operation and construction cost decreases.

node	3 – 4	5 – 6	7 – 8	9 – 10	Sum
HBPS	33,290	26,544	19,404	8,832	88,070
DPBS	33,864	27,858	20,865	10,736	93,323
increase	−574	−1314	−1461	−1904	−5,253

Table 2. Result of comparison analysis of big data processing time by number of data nodes

Analysis of data loss according to the number of nodes analyzes the amount of data loss according to the number of nodes based on the probability of occurrence of failures derived from the requirements analysis. The number of data nodes is defined as at least 3 to 100, the data size is defined as , and the basic block size is defined as . <Figure 8> and <Table 3> show the data loss analysis according to the number of nodes. According to the results, as the number of data nodes increased, the amount of data loss due to failure increased rapidly. In the big data infrastructure environment with the number of data nodes from 21 to 40, the amount of data loss increased slightly compared to the big data infrastructure environment with the number of nodes from 3 to 20 , but the number of data nodes increased from 81 to 100. In the big data infrastructure environment, the amount of data loss increased rapidly. At this time, the amount of data loss is n blocks .

The reason for this result is that the probability of unexpected errors in the network and each data node increases as the number of data nodes increases due to the nature of the distributed system. Therefore, research and development are needed to reduce the number of data nodes and reduce the processing time of big data.

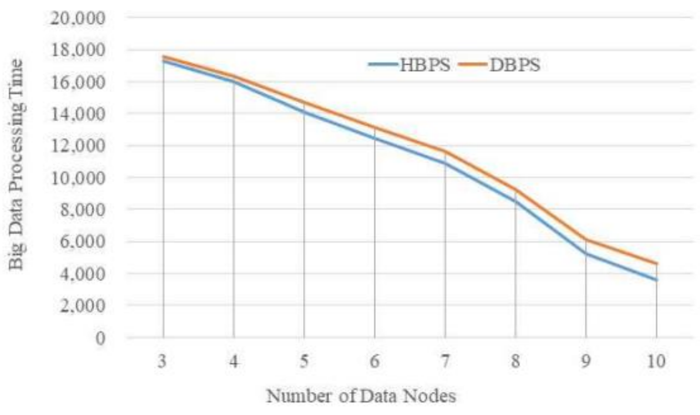


Figure 7. Big data processing time comparison analysis

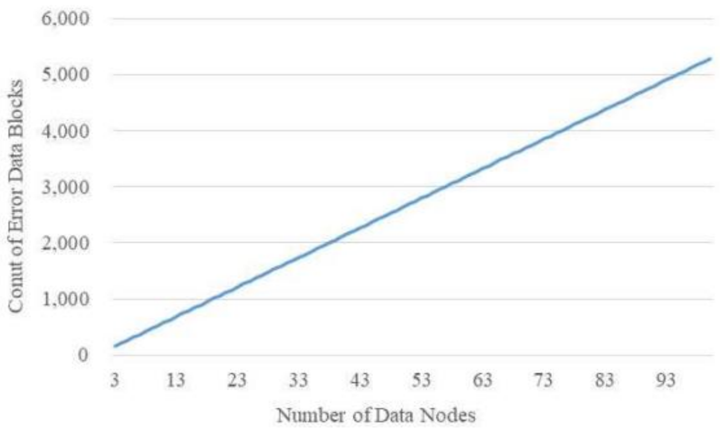


Figure 8. Analysis of data loss according to the number of data nodes

node	3 – 20	21 – 40	41 – 60	60 – 80	81 – 100
disappearance	11,074	32,168	53,262	74,355	95,449
increase	–	21,094	32,168	42,188	53,262
ratio	–	10%	47%	61%	74%

Table 3. Result of comparison analysis of error data blocks by number of data nodes

V. CONCLUSION

Due to the development of IT technology, the amount of newly created data every year is increasing exponentially. As an alternative to this, the existing big data processing technique processes a large amount of data by increasing the number of nodes and the amount of memory.

Therefore, in this paper, an in-memory-based hybrid big data processing technique was proposed to improve the big data processing rate. The proposed method adds a combiner step to the distributed system processing method, and in-memory-based processing technology is applied in that step. According to the performance evaluation results of the proposed method, the big data processing time was faster at 10 data nodes compared to the existing method, and the amount of data loss according to the number of nodes was 81 to 100 compared to the method proposed by the existing method. 53,262 blocks were lost in data nodes.

REFERENCES

[1] H. Lee, Y. Kim, J. Park and J. Lee, "Map Reduce-Based Partitioner Big Data Analysis Scheme for Processing Rate of Log Analysis," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 5, pp. 593 600, 2018

[2] H. Lee, Y. Kim, K. Kim and J. Choi, "Design of GlusterFS Based Big Data Distributed Processing System in Smart Factory," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 1, pp. 70 75, 2018

[3] D. Hwang, K. Ko, S. Park and W. Kim, "Development for establishing Big Data-based alley commercial area," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 11, No. 6, pp. 784 792, 2018

[4] H. G. Lee, Y. W. Kim and K. Y. Kim, "Implementation of an Efficient Big Data Collect ion Platform for Smart Manufacturing," Journal of Engineering and Applied Sciences, 12(2Si), pp. 6304-6307, 2018

[5] Y. Kwon and I. Kim, "A Study on Anomaly Signal Detection and Management Model using Big Data," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 16, No. 6, pp. 287 294, 2016

[6] J. Kim, J. Park and S. Chung, "Analysis of Network Log based on Hadoop," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 17, No. 5, pp. 125 130, 2017

[7] E. Jeong and B. Lee, "A Design of Hadoop Security Protocol using One Time Key based on Hash-chain," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 10, No. 4, pp. 340 349, 2017

[8] 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)

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

[10] 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.

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

[12] 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.