International Journal of Innovative Research in Engineering
Volume 3, Issue 4 (July-August 2022), PP: 96-100.
www.theijire.com **ISSN No: 2582-8746**

An Efficient Replication Method for Map Reduce Scaling

Fahri Kumar
Department of Computer Science, Sri Sairam Engineering College, Tamilnadu, India.

How to cite this paper:
Fahri Kumar, "An Efficient Replication Method for Map Reduce Scaling", IJIRE-V3I04-96-100.

Abstract: *Recently, research and utilization of distributed storage and processing systems for large-scale data processing and management are becoming important. Hadoop is widely used as a representative distributed storage and processing framework. Task assignment in Map-Reduce, which is performed based on the Hadoop distributed file system, is assigned as close as possible considering the locality of data. However, in the data analysis work in Map-Reduce, there is data that is frequently requested depending on the type of work. In this case, due to the low locality of the data, it causes problems of an increase in execution time and delay of data transmission. In this paper, we propose a Least Recently Used LRU-data replication technique according to the data access pattern to improve the processing speed of Map-Reduce. In the proposed method, data locality is improved by using a replica optimization algorithm for LRU-data showing high access frequency according to the data access pattern, and consequently the operation time is reduced. As a result of the performance evaluation, it was confirmed that the load of the access frequency was reduced compared to the existing technique.*

Key Word: *Distributed Computing, Map-Reduce, Hadoop, Locality, LRU-Data*

I. INTRODUCTION

With the recent growth of social media and the deployment of digital devices such as mobile devices in all areas of life, various types of large-scale data are rapidly being created, distributed, and stored in our daily lives. At the same time, efforts to accumulate and analyze vast amounts of data are continuing.

This is because new values that have not been seen before are systematically derived through the analysis of various digitally accumulated data. It seems that research to utilize big data, which has been an issue steadily for several years, is on track in earnest. To cope with this exponentially growing data with commercial solutions, the resulting cost burden must also increase exponentially [1][2]. Therefore, research and utilization of distributed storage and processing systems for large-scale data processing and management are becoming important, and the answer is recently being obtained from open source solutions. Hadoop [3] [4][5]vis a representative open source software framework that supports distributed applications for storing and processing large-scale data.

Hadoop is a Hadoop Distributed File System (HDFS) for storing large-scale data of more than petabytes in a cluster environment [6] and MapReduce [5][7][8] to support parallel processing based on it. It consists of a framework. The Hadoop distributed file system and Map-Reduce are receiving a lot of attention due to their high usability as confirmed in real cases, and studies for further performance improvement are being actively conducted [6][7][9].The Map-Reduce framework is composed of Map and Reduce functions that are commonly used in functional programming. In the map stage, according to the definition of the function, chunks (data blocks) are read and the processed data is changed into a key-value form, and in the reduce stage, the result of the map stage is merged and output. The Hadoop distributed file system basically stores three chunk copies in node, rack, and off-rack in consideration of data loss or failure. Task assignment in Map-Reduce, which is performed based on the Hadoop distributed file system, is assigned as close as possible considering the locality of data[10].

In other words, if a task is assigned to the node where the chunk required to perform the task is located, or if the node is performing another task, it is another node in the rack where the node is located, and all other nodes in the rack where the node is located also perform different tasks[11][12][13]. If it is in the middle, it allocates the task of another rack to the node that is not performing. At this time, if the data to be processed is located in another node, additional operations such as data search, access, and transmission are required to call the corresponding chunk to the processing node, which increases the execution time of the corresponding operation.

In addition, by occupying a limited bandwidth on a network composed of multiple nodes, there is a problem of delaying the transmission of other data as well.In the data analysis work in Map-Reduce[10][12], there is data that is frequently requested according to the type of work. For example, when performing comparison with specific dictionary data, the dictionary data as a reference is required by most nodes. When data comparison is performed, it causes problems of increase in execution time and delay of data transmission when the locality of prior data is low [15].Considering these problems, in this paper, we propose a LRU-data replication technique according to the data access pattern to improve the processing speed of Map-Reduce[16][17]. The proposed method utilizes the replica optimization algorithm for LRU data with high access frequency according to the data access pattern. To solve As a result, it is possible to reduce the task execution time by improving data locality.The structure of this paper is as follows. Section 2 describes the problems and research objectives of the previously proposed Hadoop framework. Section 3 describes the LRU-data replication technique according to the data access pattern to improve the processing speed of the proposed Map-Reduce. Section 4 shows the superiority of the proposed technique through performance evaluation with existing techniques, and the final section 5 presents conclusions and future research directions.

II.PREVIOUS WORK

Techniques for processing large amounts of data are being actively studied. Hadoop, a representative framework for processing such large amounts of data, supports distributed applications that can process large amounts of data. Hadoop is a technology that implements the Hadoop distributed file system using the Google File System (Google File System) [9][16][18], the underlying system used in Google , and Map-Reduce, a data distributed processing framework based on it. Map-Reduce processes data by combining two functions, Map and Reduce. Map is a process that accepts a set of data and creates new data through predetermined processing[19].

As shown in [Figure 1], the Hadoop framework is divided into the MapReduce and Hadoop file systems described above. The job tracker of the master node allocates the job to the task tracker of each slave node in consideration of the job execution time and the location of the data node. And since Namenode manages metadata including the location of each chunk of data when initial data is input, it is used when job trackers use a shovel. Then, the initial batch of data is randomly discharged, and basically 3 chunk copies are maintained in consideration of data loss or failure.

However, in the data analysis work in Map-Reduce, there is data that is frequently requested depending on the type of work. For example, when performing comparison with specific dictionary data, the dictionary data that serves as a reference is required by most nodes[20]. When data non-table is performed, it causes problems of delay in data transmission and increase of execution time in a state where the locality of the dictionary is low. And when arranging data in the chunk size in data node, when arranging each chunk data in chunk size in 3 nodes, 3 copies of each chunk size are randomly placed without considering the location so that the same data skewed phenomenon appears in a similar position[21][22].

Considering this problem, it is necessary to extract LRU-data according to the access frequency and then create a chunk replica to reduce the load on the node and to solve the data clustering phenomenon by replicating locality in consideration.

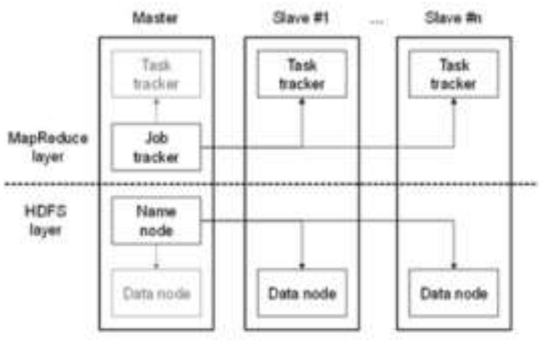


Figure 1: Hadoop Framework

III.PROPOSED METHOD

a. Structure of the proposed system

In this section, we propose a LRU-data replication algorithm that considers data access patterns to improve map-reduce processing speed. In the proposed method, in order to solve the node load problem according to the data analysis work type in the existing Map-Reduce and the clustering of data arrangement, the data access management table is created in the namenode and then LRU-data according to the access frequency. After extraction, the number of copies is calculated. In addition, in order to solve the phenomenon of data aggregation, replicas are placed in consideration of locality as much as possible. The system structure of the proposed method is similar to that of the existing method, but the name node of the Hadoop framework has been modified. Namenode records the location of each chunk of data through file metadata and records the access frequency through the access frequency table to use it for LRU-data extraction. In addition, replicas that consider locality as much as possible through data node location information are additionally placed.

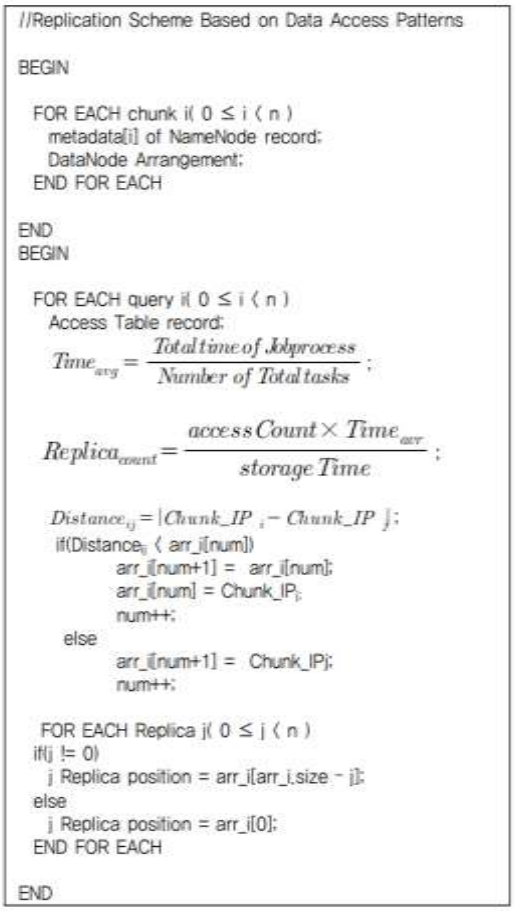
b. Data Extraction

In the proposed technique, data access patterns are recorded in NameNode to extract LRU-data. In the Map-Reduce framework, a job submitted from a client is divided into tasks by the job tracker and assigned to the task tracker. In order to find the data required to perform the job, the task tracker accesses the Hadoop distributed file system and utilizes the metadata. To implement the proposed technique, a module for identifying the data access frequency of a task is loaded on the NameNode that manages data, and an access pattern storage structure is created. The file name is the name including the location of the original file, the replica list is a list of names including the location of the replica, the block ID indicates the location of the block where the replica is stored in the data node, and the access number indicates the number of accesses to each replica indicates.The access frequency per average task processing time is used to derive the optimal number of replicas according to the access frequency. At this time, the average task processing time is calculated based on Equation (1).

$$Time_{avg} = \frac{Total\,time\,of\,Jobprocess}{Number\,of\,Total\,tasks} \qquad (1)$$

Equation (1) means the total task assigned to all nodes per job processing time by the Hadoop framework from the client. Through this, it is possible to calculate the optimal number of replicas based on the average processing time per task. Equation (2) shows the operation to calculate the number of copies of each data using the average task processing time, number of accesses, and data storage time.

$$Replica_{count} = \frac{access\,Count \times Time_{avr}}{storage\,Time} \qquad (2)$$



```
//Replication Scheme Based on Data Access Patterns

BEGIN

  FOR EACH chunk i( 0 ≤ i ( n )
    metadata[i] of NameNode record;
    DataNode Arrangement;
  END FOR EACH

END
BEGIN

  FOR EACH query i( 0 ≤ i ( n )
    Access Table record;
    $Time_{avg} = \frac{Total\,time\,of\,Jobprocess}{Number\,of\,Total\,tasks}$;

    $Replica_{count} = \frac{access\,Count \times Time_{avr}}{storage\,Time}$;

    $Distance_{ij} = |Chunk\_IP_i - Chunk\_IP_j|$;
    if(Distance_ij ( arr_i[num])
            arr_i[num+1] =  arr_i[num];
            arr_i[num] = Chunk_IPj;
            num++;
      else
            arr_i[num+1] =  Chunk_IPj;
            num++;

   FOR EACH Replica j( 0 ≤ j ( n )
  if(j != 0)
    j Replica position = arr_i[arr_i.size - j];
  else
    j Replica position = arr_i[0];
  END FOR EACH

END
```

Figure 2. Data replication algorithm.

c. Replica replacement algorithm

In the proposed method, after calculating the optimal number of replicas, a placement strategy is taken that considers locality as much as possible, rather than randomly placing replicas. As shown in [Figure 4], the Hadoop framework uses an IP address allocation policy based on distance internally. In detail, the closer the address matches, the closer it can be said to a node. Replicas are placed using address assignments according to these distances . To evenly distribute replicas, placement of replicas checks the addresses of existing data placement nodes and places them where data does not exist. For example, if data exists only in node 10.0.1.1, the replica is placed in 10.1.YY first. With this replica placement policy, it is possible to maximize the locality of data.

IV.EXPERIMENTAL RESULTS

In this section, to prove the superiority of the proposed method, performance evaluation was performed through simulation with the existing Hadoop framework [3]. This simulation was performed by implementing its own Java simulator. [Table 1] shows the performance evaluation environment. After fixing the total number of nodes in the cluster to 100 and the total chunk data to 1000, the comparison data required for task execution was changed to 50 and 200, and the standard deviation of the access frequency of all nodes was compared and evaluated. Here, comparison data refers to data that requires frequent data access because it requires comparison with other data during query processing. In addition, query processing time according to changes in network bandwidth was compared and evaluated.

Table 1: System configuration

parameter	value
Total Nodes (EA)	100
Whole Chunks of Data (pcs)	1000
Comparison data (pieces)	
Bandwidth (Mb/s)	

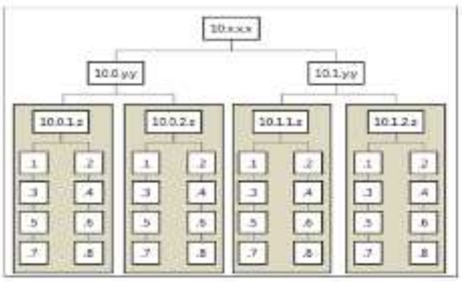


Figure 3: Replica replacement policy

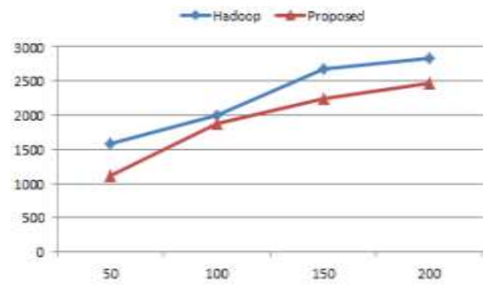


Figure 4: Hadoop vs Proposed performance

Figure 4 and 5 compares and evaluates query processing time according to bandwidth. In this performance evaluation, the total number of nodes is 100, and the total number of chunks of data is 1000 and 200 data for comparison. By law, the node load is reduced by increasing the number of replicas to reduce the query processing time and consequently to minimize. It is possible As a result of performance evaluation, compared to the existing Hadoop the average query processing time in the proposed method is about 55% decreased. In this paper, comparison data with the existing Hadoop framework the standard deviation and bandwidth of node access frequency according to the number of the query processing time according to the change was compared. As a result the access load concentrated on the node that stores the comparison dataIt was confirmed that it is distributed to other nodes, and as a resultIt was confirmed that the query processing speed was improved.

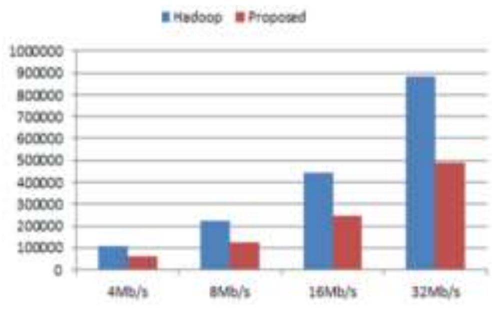


Figure 5: Bandwidth comparison between Hadoop and proposed method

V.CONCLUSION

In this paper, we analyze the problems of existing techniques and analyze data access patterns to improve the processing speed of MapReduce. A replication technique was proposed to suggest according to the data access pattern, the proposed algorithm requires a high frequency of access. A replica optimization algorithm for visible is proposed to improve data locality and reduce processing time. As a result of the performance evaluation, the proposed method is compared to the existing technique, the load on the frequency of access is reduced by about 8% on average. The experimental results confirmed that the average query processing time was reduced by 55%.

References

[1]. J. Dittrich and J. Quiané-Ruiz, "Efficient Big Data Processing in Hadoop MapReduce," Proc. of the VLDB Endowment, Vol.5, No.12, pp.2014-2015, 2012.

[2]. J. Cohen, J. Dolan, M. Dunlap, J. Hellerstein, and C. Welton, "MAD Skills: New Analysis Practices for Big Data," Proc. of the VLDB Endowment, Vol.2, No.2, pp.1481-1492, 2009 .

[3]. Shvachko, H. Huang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," Proc. of the IEEE Symposium on Massive Storage Systems, pp.1-10, 2010.

[4]. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communication of the ACM, Vol.81, No.1, pp.107-113, 2008.

[5]. Lu, Peng, Gang Chen, Beng Chin Ooi, Hoang Tam Vo, and Sai Wu. "Scalagist: Scalable generalized search trees for mapreduce systems [innovative systems paper]." Proceedings of the VLDB Endowment 7, no. 14 (2014): 1797-1808.

[6]. 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.

[7]. I. Hwang, K. Jung, .Im, and J. Lee, "Improving the Map/Reduce Model through Data Distribution and Task Progress Scheduling," Journal of the Korea Contents Association, Vol.10, No.10, pp.78-85,

[8]. Lin, Yuting, Divyakant Agrawal, Chun Chen, Beng Chin Ooi, and Sai Wu. "Llama: leveraging columnar storage for scalable join processing in the mapreduce framework." In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 961-972. 2011.

[9]. H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters," Proc. of the ACM SIGMOD International Conference on Management of Data, pp.

[10]. S. Ghemawat, H. Gobioff, and S. Leung. "The Google File System," Proc. of ACM Symposium on Operating Systems Principles, pp.29-43,

[11]. H. Zhao, S. Yang, Z. Chen, S. Jin, H. Yin, and L. Li, "MapReduce Model-Based Optimization of Range Queries," Proc. of the International Conference on Fuzzy Systems and Knowledge Discovery(FSKD '12), pp.2487-2492,

[12]. Lars George, "HBase: The Definitive Guide", O'REILLY, 2011

[13]. Kristina Chodorow, "MongoDB: The Definitive Guide Edition", O'REILLY, 2013

[14]. B. Pang, ,L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval: Vol.2, No.1-2,pp.1-135,

[15]. Manakkadu, Sheheeda, and Sourav Dutta. "On efficient resource allocation in the Internet of Things environment." In Proceedings of the 8th International Conference on the Internet of Things, pp. 1-5. 2018.

[16]. S. Mukherjee, P. Bhattacharyya, "Sentiment Analysis in Twitter with Lightweight Discourse Analysis", Proc. of COLING 2012, pp.1847-1864, 2012

[17]. N. Godbole, S. Skiena, "Large-Scale Sentiment Analysis for News and Blogs", Proc. of the ICWSM'2007, 2007

[18]. Li, Ren, Haibo Hu, Heng Li, Yunsong Wu, and Jianxi Yang. "MapReduce parallel programming model: a state-of-the-art survey." International Journal of Parallel Programming 44, no. 4 (2016): 832-866.

[19]. A. Pak, P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", Proc. of the LREC'2010, 2010

[20]. H. Tang, S. Tan, X. Cheng, " A survey on sentiment detection of reviews," Expert Systems with Applications, Vol.36, pp.10760-10773,

[21]. Seth Gilbert, Nancy Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT New 33(2), pp. 51-59, 2002 .

[22]. J. Dean, S. Ghemawat, "MapReduce; Simplified Data Processing on Large Clusters", Communications of the ACM, Vol.51, No.1, pp.107-113, 2008