SFSAN Approach for Solving the Problem of Small Files in Hadoop

Tharwat EL-SAYED, Mohammed Badawy, and Ayman EL-SAYED
Computer Science and Engineering. Department., Faculty of Electronic Engineering, Menoufia University
Menouf 32952, Egypt
tharwat_uss89@hotmail.com, mohamed.badawi@el-eng.menofia.edu.eg, ayman.elsayed@el-eng.menofia.edu.eg

***Abstract*— Hadoop is a distributed computing framework written in Java and used to deal with big data; it is designed to handle large files. Handling the small files leads to some problems in Hadoop performance. Several approaches are used to overcome the housing and the access efficiency problems of the small files in Hadoop. These approaches are Hadoop Archive Files, Federated NameNodes, Batch file consolidation, Sequence files, HBase, and S3DistCp. All these approaches have some advantages, but they have some limitations as the slow in converting all the HDFS Files to Sequence Files and the lack of correlations between the files stored in the Sequence Files. In this paper, we proposed an enhancement of the Sequence files approach called Small Files Search and Aggregation Node (SFSAN) approach. Our proposed approach improves the Hadoop performance by overcoming some of the limitations of the Sequence Files approach and keeping up its advantages.**

Keywords*— *HDFS; DataNode; NameNode; Sequence file; Small Files; Hadoop; MapReduce; and SFSAN.

I. INTRODUCTION

In the International Digital Corporation (IDC) it is estimated that the digital data in the world will reach 3,500,000 Exabyte by 2020. The traditional computing technologies become inadequate for storing, managing and processing massive datasets. So, the need for a new platform becomes unavoidable, this platform should be a distributed computing framework to handle the massive data sets [1-5]. Many companies like Yahoo, Amazon, Facebook, and NewYorkTimes uses Hadoop framework [6]. The HDFS is a distributed file system and used as the storage of Hadoop. The HDFS consists mainly of several DataNodes which are used for storing the data and only one NameNode used for organizing client access to the DataNodes and managing the metadata of the stored files. Fig. 1 represents the HDFS architecture [7, 8].

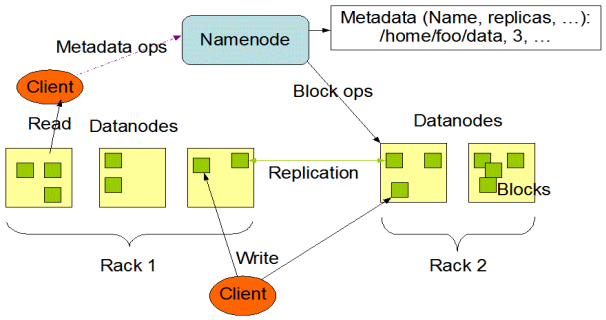

Figure 1: The HDFS Architecture

Each file will be defined as an object in the HDFS and each of which requires 150 bytes of metadata. A Large number of small files stored in the system metadata occupies a large portion of the namespace, so the time consumed by the MapReduce operations will be increased. The DataNode block size is 128 Mbytes long and if this length is duplicated, small files problem becomes unavoidable and solving this problem will be a good achievement. Several approaches have been used for solving the NameNode memory consumption problem and the MapReduce performance problem. Examples of those approaches are Hadoop Archive Files, Federated NameNodes, Change the ingestion process/interval, Batch file consolidation, Sequence files, HBase, and S3DistCp. Sometimes choosing among these approaches becomes elusory but can be evaluated according to some metrics like the number of DataNode blocks consumed for storing each file, the consumed NameNode memory, or the CPU time to perform operations on the data (MapReduce performance) [9, 10]. Reading and processing the small files causes a lot of hopping from one node to another to obtain each small file. This operation leads to a big burden over the Hadoop [11-13]. Storing the small files inside the HDFS and running MapReduce operations on these small files leads to low efficiency in Hadoop performance. The problem of the small files is as a result of three subproblems: (1) The existence of the small files will consume the NameNode memory (metadata), (2) HDFS file access mechanism is not proper for accessing the small files, and (3) HDFS lacks Input / Output optimization mechanism like file prefetching and caching [13,14]. This research will improve the Hadoop performance based on searching for the HDFS small files and aggregating them in larger sequence files. Our proposed approach will work on a sequence file trying to overcome its drawbacks. The novelty of our work results from installing the SFSAN node in the Hadoop cluster. This node is the main coordinator of our work. It helps to overcome the correlation and delay limitations that face the Sequence Files approach. In our approach, we will run the HDFS Find Tool periodically for aggregating the small files depending on three factors determined by the organization like (1)The defined size of the files to be considered as small files. (2)Some words in the file name or the extension of the files. (3)The period after which the HDFS Find Tool will run. By using these three factors, we can overcome the limitations of the Sequence Files Approach and still keeping its advantages. Our technique reduces the consumed memory from the NameNode and the consumed DataNodes memory when storing the data blocks. Our

technique operation is to search for the HDFS small file and save them in sequence files to reduce the number of DataNode blocks consumed for storing these files, which in turn reduces the size of NameNode metadata, reduces the MapReduce tasks overhead, and also reduces the total CPU processing time[15-17]. The rest of this paper is organized as follows: Section II discusses the Sequence Files approach. In Section III, we introduce the proposed approach. Section IV describes the HDFS Find Tool. In Section V, the performance evaluation is introduced. Finally, Section VI concludes the paper.

II. SEQUENCE FILES APPROACH

The file format is the layout of the file, which determines the data organization in the file. The file format should be known to the programs that deal with the file in order to enable these programs from accessing the data. Sequence file is one of these file data formats. Sequence files depend mainly on key-value pairs, so, it is used with programs that prefer the key-value pair data format. Sequence files use only Java for data manipulation and there are well known Java API classes were written in order to enable us to write, read, and sort the sequence files. The Hadoop small files problem can be solved by using the sequence file format. Also, the custom MapReduce programs use the sequence file format depending on the key-value structure for enhancing the parallel processing methodology [18]. The main reason for using the Sequence files approach is to save the file name of small files when storing these small files in the HDFS. The sequence file consists of two components, the key which used to store the file name, and the value which used to store the file content. Table 1, shows how the small files would be stored in a sequence file:

TABLE I: STORING THE SMALL FILES IN A SEQUENCE FILE

Key	Value	Key	Value	Key	value
file1.txt	file1 contents	file2.txt	file2 contents	fileN.txt	fileN contents

If we have 3,000,000 small files, the created sequence file also will contain 3,000,000 keys and 3,000,000 values, each key in the sequence file corresponds to one of the small files. The Sequence files approach supports the block compression attribute. The sequence files can be divided into smaller parts at MapReduce operations, this means that each map task will run for each block (128 MBs) rather than running a map task for each small file. This approach enables us to save the name of the small file and to create multiple sequence file at the same time. For creating one sequence file, a large time will be consumed, so the sequence file is not practical in the case of a plenty of small or large files. It can work slowly in converting the existing data into sequence files, but it is possible to gather multiple sequence files in parallel for manipulating the delay problem. The small files are collected and organized as a group, but it is difficult to work on this sequence files as accessing or storing one by one[14]. For this reason and to improve the efficiency in Hadoop, we should use a parallel mechanism in order to access these sequence files at the same time. On the other hand, the Sequence File isn't suitable for dealing with large files. The Limitations of the sequence File Approach can be summarized in the following two points:

A. The slow in converting all the HDFS Files to Sequence Files [19].

B. The lack of consideration of files correlations when storing the files in the HDFS [20, 21].

III. PROPOSED APPROACH

In this section, we introduce an enhancement approach for manipulating the Sequence Files problem. Our proposed approach is called Small Files Search and Aggregation Node (SFSAN) and it is shown in Fig. 2. The SFSAN approach is used to perform two functions. The first one is to run the HDFS Find Tool periodically to search for the small files stored in the HDFS according to a well-defined methodology based on one or more of the attributes introduced by the HDFS Find Tool like (file size, file type, and the time of update). The purpose of this process is to avoid converting the large files to sequence files. The other function of the SFSAN approach is to convert the obtained small files into sequence files and deleting the original small ones. The flowchart in Fig. 3 explains these operations. The small files problem can be handled more effectively by using the SFSAN approach; it can overcome the limitations of the Sequence Files approach and still have its advantages. The first limitation is slow in converting all the HDFS Files to Sequence Files, this limitation can be solved by searching for files with size less than a definite size; this size is a dynamic value and can be determined by the organization. In our experiments, we preferred to search for the files with size less than 25% of the DataNode block size (which is 128 Mbytes). The second limitation is the lack of correlations between the HDFS files and it can be solved by storing the files based on the files extensions like .txt, .jpg, .mp4, .ppt, etc. and also storing the files based on part of the file name. By using our methodology we can build the correlation between the stored files.

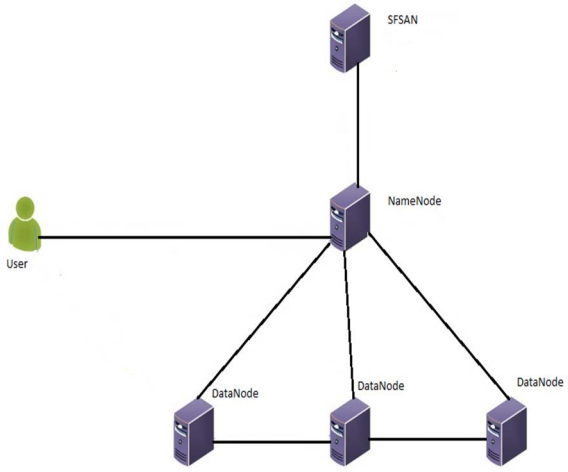

FIGURE 2: THE CLUSTER ARCHITECTURE

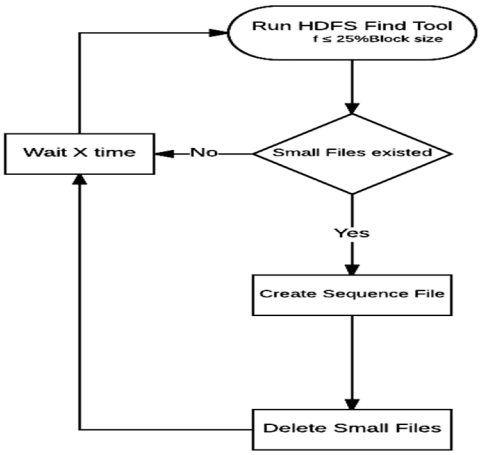

Figure 3: The SFSAN Approach Flow Chart

IV. HDFS FIND TOOL

HDFS Find Tool is similar to the find command used in Linux file system but it is updated in the HDFS version. One advantage of the find command is it can traverse one or more HDFS directory trees, finds all HDFS files that match the used expression and applies selected actions on them. The result will be moved to stdout, and its format will be one path per line. For example, to find all files that are contained in the directory tree hdfs:///usr/$USER/InputData, have a name matching sample *.txt, were modified less than 30 minutes ago, and their size is ≤ 25% of the DataNode block size we used the following command [22]:

```
$ hadoop jar /search-mr-*-job.jar
org.apache.solr.hadoop.HdfsFindTool -find
hdfs:///usr/$USER/InputData -type f -name 'sample*.txt' -
mmin -30 -size -32000000c
```

V. PERFORMANCE EVALUATION

In this Section, we evaluate the Hadoop performance by using the SFSAN approach by measuring the following metrics:

- The consumed DataNode memory.
- The consumed NameNode memory.
- The time is taken by MapReduce operations.

We also compared it with the native HDFS and the SequenceFiles approach.

A. Environment Settings

The experimental platform is established on one node, which is Dell Inspiron 1545 with Dual-core Intel CPU (2.00 GHz), 4.00GB memory acts as NameNode, as DataNode with replication factor equals 1 and also plays the SFSAN role.

B. Evaluation Metrics

1) *The consumed DataNode Blocks:* The number of blocks consumed by each file reflects the optimum use of the DataNode memory. Files of a size smaller than the default block size will waste a lot of DataNode memory.

2) *The consumed NameNode Memory:* Defines the size of the metadata for the stored data in the HDFS. The capacity of the namespace of the system is limited by its physical memory. Files, directories, and blocks are named objects in HDFS and each of which requires 150 bytes for indexing and saving them.

3) *The MapReduce applications processing time:* Defines the time taken by the MapReduce application. The less CPU processing time the highest Hadoop performance.

C. Methodology

Experiments include two steps: searching for the small files located in the HDFS and collecting them into sequence files, then, deleting the original files from the HDFS. To compare our SFSAN proposed approach, the sequence files approach, and the native HDFS, we evaluate them in terms of memory overhead and time cost when performing the MapReduce word count application. These metrics are evaluated with a system store 400, 800 and 1000 small files. The HDFS Find Tool used to find these small files and by passing them to the next stage that will collect these small files in larger sequence files. In large organizations, running the HDFS Find Tool will be used according to the needed methodology that will be determined by the file type, the file size, and the update time in the HDFS. All of the three attributes can be reached in the Hadoop find command extensions.

D. Results

The occupied memories of the DataNode and the NameNode are measured when storing a lot of small files using the native HDFS approach, the sequence files approach and SFSAN approach. Figures 4-6 describe our results. Due to the merging facilities, the SFSAN and the sequence file approaches consume fewer memories of NameNode than the native HDFS that leads to increasing the storage efficiency. If we run the MapReduce wordcount program, it will result in a big reduction in the CPU processing time when using the SFSAN approach and the SequenceFiles approach. Fig. 4 shows that saving files in the HDFS by using the SequenceFiles approach and the SFSAN approach is more practical than using the Native HDFS. Fig. 5 shows that using the SequenceFiles approach and the SFSAN approach is more practical than using the Native HDFS for solving the problem of NameNode memory consumption. Fig. 6 shows that the MapReduce word count program takes less time in case of the SequenceFiles approach and the SFSAN approach, but it takes a lot of time in the case of the Native HDFS. Finally, we can conclude that the SFSAN approach still saving the advantages of the Sequence Files approach and in addition, it overcomes its limitations (converting large files to sequence files and the lack of consideration of files correlations).

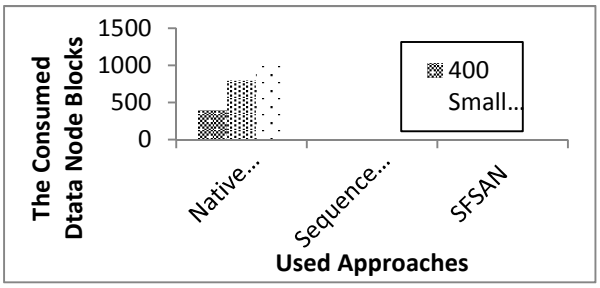

Figure 4: The Storage Consumption in the DataNode

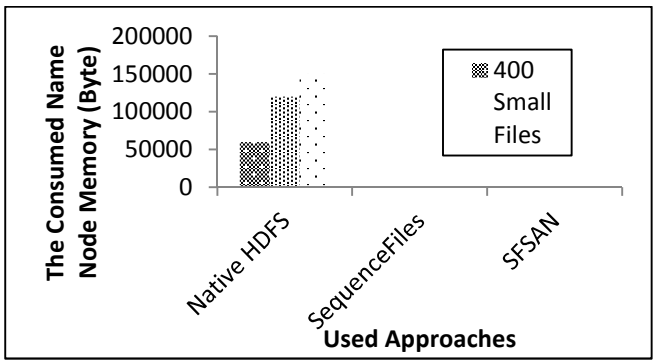

Figure 5: The NameNode Memory Consumption

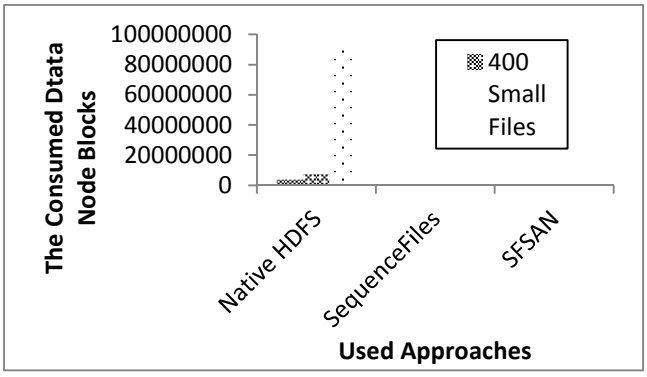

Figure 6: The MapReduce Word Count Program Processing Time

CONCLUSION

The process of storing and processing of the small files in the HDFS leads to low efficiency in the Hadoop performance. Several approaches are introduced to solve the small files problem in Hadoop including Sequence File Approach, but still some shortcomings are existed. Our proposal; SFSAN approach is an enhancement for the Sequence Files approach. It overcame the two limitations of the Sequence File Approach. Firstly, it solved the problem of the slow in converting the HDFS Files to Sequence Files by searching for the files with size less than a predefined value; this size is a dynamic and can be determined by the organization. Secondly, it solved the lack of correlations between the HDFS files by storing the files basing on their extensions or basing on part of the file name. Finally, the experimental results proved that the SFSAN approach will overcome the shortcomings of the Sequence File Approach maintaining its advantages.

REFERENCES

[1] Essa, Y. M., Attiya, G., & El-Sayed, A. (2013, December). Mobile agent based new framework for improving big data analysis. In 2013 International Conference on Cloud Computing and Big Data (CloudCom-Asia), pp. 381-386, IEEE.

[2] Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In 2013 International Conference on Collaboration Technologies and Systems (CTS), pp. 42-47, IEEE.

[3] Zhao, X., Yang, Y., Sun, L. L., & Huang, H. (2012). Metadata-Aware small files storage architecture on hadoop. In International Conference on Web Information Systems and Mining (WISM 2012), pp 136-143.

[4] Guru Prasad, M. S., & Nagesh, H. R. (2017). Swathi Prabhu,". High Performance Computation of Big Data: Performance Optimization Approach towards a Parallel Frequent Item Set Mining Algorithm for Transaction Data based on Hadoop MapReduce Framework. In International Journal of Intelligent Systems and Applications 9(1): 75-84 · January 2017.

[5] Khan, Imran, et al (2017). An efficient framework for real-time tweet classification. International Journal of Information Technology, 9.2 (2017): pp. 215-221.

[6] Meng, B., Guo, W. B., Fan, G. S., and Qian, N. W. (2016). A novel approach for efficient accessing of small files in HDFS: TLB-MapFile. In proceeding of Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 17th IEEE/ACIS International Conference, pp. 681-686, Shanghai, China, 2016.

[7] White, Tom. Hadoop: The definitive guide. O'Reilly Media, Inc., 2012.

[8] Wang, Feng, et al (2009). Hadoop high availability through metadata replication. Proceedings of the first international workshop on Cloud data management, ACM, 2009.

[9] Mall, Nupur N., and Sheetal, Rana. Overview of Big Data and Hadoop. Imperial Journal of Interdisciplinary Research 2.5, 2016.

[10] Manjunath, R., R. K. Channabasava, and S. Balaji (2016), A Big Data MapReduce Hadoop distribution architecture for processing input splits to solve the small data problem. 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, IEEE.

[11] Idrees, Sheikh Mohammad, M. Afshar Alam, and Parul Agarwal (2018). A study of big data and its challenges. International Journal of Information Technology (2018), pp. 1-6.

[12] Chandra, Pravin, and Manoj K. Gupta (2018). Comprehensive survey on data warehousing research. International Journal of Information Technology 10.2 (2018), pp. 217-224.

[13] Gao, S., Li, L., Li, W., Janowicz, K., and Zhang, Y. (2017). Constructing gazetteers from volunteered big geo-data based on Hadoop. Journal of Computers, Environment and Urban Systems, Vol 61, pp. 172-186, 2017.

[14] Gupta, B., Nath, R., and Gopal, G. (2016). An Efficient Approach for Storing and Accessing Small Files with Big Data Technology. International Journal of Computer Applications, International Journal of Computer Applications, Vol 146 – No.1, pp. 36-39, July 2016

[15] Guo, Y., Rao, J., Cheng, D., and Zhou, X. (2017). iShuffle: Improving Hadoop Performance with Shuffle-on-Write. IEEE Transactions on Parallel and Distributed Systems, 28(6), pp. 1649-1662, 2017.

[16] Yong, Keh Kok, Hong Hoe Ong, and Vooi Voon Yap. GPU SQL Query Accelerator. International Journal of Information Technology, 22.1 (2016).

[17] Nagesh, H. R., and Prabhu, S. (2017). High Performance Computation of Big Data: Performance Optimization Approach towards a Parallel Frequent Item Set Mining Algorithm for Transaction Data based on Hadoop MapReduce Framework. International Journal of Intelligent Systems and Applications, 9(1), pp. 75-84, 2017.

[18] Lu, Xiaoyi, et al (2013). High-performance design of Hadoop RPC with RDMA over InfiniBand. Parallel Processing (ICPP), 2013 42nd International Conference on. IEEE, 2013.

[19] Gohil, P. and Panchal, B. (2014). Efficient ways to improve the performance of HDFS for small files. Journal of Computer Engineering and Intelligent Systems, Vol. 5, No.1, 2014, pp. 45-49.

[20] Dong, Bo, et al (2012). An optimized approach for storing and accessing small files on cloud storage. Journal of Network and Computer Applications, Vol. 35, Issue 6, pp. 1847-1862, November 2012.

[21] Tong Zheng, Weibin Guo, and Guisheng Fan. (2017). A Method to Improve the Performance for Storing Massive Small Files in Hadoop. The 7th International Conference on Computer Engineering and Networks (CENet2017) Shanghai, China, 2017.

[22] Wang, M., Grindrod, R., O'Meara, J., Zuzuarregui, M., Martinez, E., and Fallon, E. (2015). Enterprise search with development for network management system. In 2015 IEEE International Congress on Big Data, pp. 430-437, New York, USA, 2015.