International Journal of Pure and Applied Mathematics
Volume 119 No. 15 2018, 2833-2841
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

Small files problem in Hadoop -A Survey

Rathidevi R[1], Sujatha Srinivasan[2]
[1]*Research Scholar,*
School of Computing Sciences, VISTAS,
Chennai, India, rathirajsuresh@gmail.com
[2]*Asso. Prof & Head, Dept. of Information Technology*
School of Computing Sciences, VISTAS
Chennai, India, ashoksuja08@gmail.com

***Abstract* —** Hadoop and MapReduce are open source software and enables distributed computing. It efficiently deals with Big data and can handle both structured and unstructured data in a better manner. It contains two essential components HDFS and MapReduce. HDFS (Hadoop Distributed File System) handles a small number of large files efficiently, but not with a large number of small files. A huge number of small files consume a large quantity of memory. Each small file creates a map task and a large number of map tasks slow down the performance. Various methods are identified to handle a set of small files in an efficient manner. This paper provides a comparative study of various methods for handling small files in the Hadoop system.

Keywords—Big data, MapReduce, Hadoop, Clusters, Small Files, Merging

1. Introduction

Internet service growth produces a large amount of data which are stored in data clouds. HDFS is a distributed data storage system with master-slave Architecture. Hadoop Architecture has two components, one NameNode and some DataNodes. NameNode holds all the information about files. NameNode is accessed frequently to get the information about stored data for access. When a large number of small files are stored, the performance of HDFS is reduced because there is a bottleneck created for handling requests for Meta data about a massive number of small files. HDFS provides unlimited storage for storing large files. But storing small files in large number is a big problem. Meteorology, social media, e-business, healthcare, and other areas produce a massive quantity of data in small size, which is comparably smaller than the block size. Moreover, each file occupies a separate location. So it consumes a large quantity of memory in storing Meta data in NameNode. NameNode is frequently called to know the information about files in DataNode, it slows down the process. Various methodologies combined to improve HDFS have been proposed. Two such methods are found widely in the literature. The first is by increasing the number of name nodes instead of having single NameNode to improve the efficiency and another way is by merging small size files up to the block size to utilize memory efficiently. Section 2 gives an overview of Hadoop-MapReduce system. Section 3 reviews the literature on handling a large number of small files by Hadoop and MapReduce. Section 4 concludes the article.

2

2. Overview of small files problem in Hadoop

Hadoop's design is for batch processing of voluminous data. The default Data Block size of HDFS is 128 MB. When the file size is remarkably smaller than the block size, then the efficiency is significantly degraded. The important parts of Hadoop Framework are HDFS and MapReduce which are explained in the following sections.

2.1 HDFS role in Hadoop

HDFS deals with huge amount of file in an easily accessible manner. Figure 1 gives the architecture of HDFS. Files are stored on many machines in replicated manner. Every file in HDFS is broken into small blocks and each block is duplicated in the cluster of Hadoop. Hadoop Distributed File System (HDFS) stores Metadata on dedicated servers. The HDFS function of the file systems enables files to be distributed to different machines. Application data on servers refer to Data Node and System data on servers referred to as Name Node. The NameNode and DataNodes are used to check the status of the cluster.

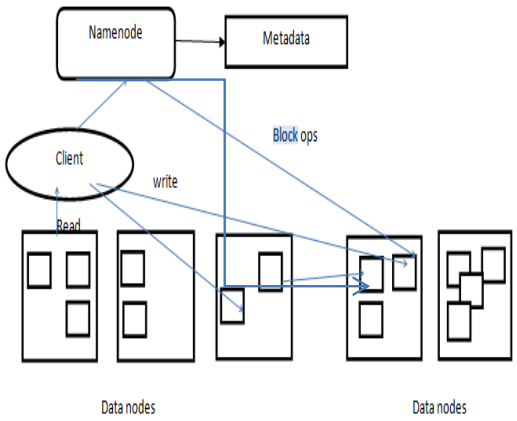

Fig. 1 Architecture of HDFS

The Name Node is commodity hardware with GNU/LINUX Hardware and Name node software. It is used to open, close and rename the file operations in the Namespace file system and directories. It also finds the matching blocks of Data Nodes. The file system client requests like read and write are performed by Data Nodes. According to NameNode instruction, The Data Nodes also do operations like creating and removing a block. HDFS is used to detect faults and recovery. It has large data sets to manage applications. It decreases network congestion and increases the result.

2.2. MapReduce role in Hadoop

Map Reduce contains a framework for creating applications to execute large quantity of information simultaneously, on a large group of commodity hardware with high performance. A Map-Reduce task generally breaks the input data-set into individual blocks. Map tasks perform actions on individual blocks at the same time. The framework arranges the output of the maps and then they are given as input to minimize the tasks. Both the input and the output from the system are stored in a file-system. The framework is used for scheduling problems, observing them and re-executes the problems which are not processed. Major Data processing techniques mappers and reducers are coming under MapReduce concept. Converting job to the mapper and reducer is not a simple task. If the application is placed in the Map-Reduce form once, measuring the application to run on a number of systems in a cluster is simply a change of layout. The Map-Reduce algorithm has two tasks Mapper and Reducer.

2.2.1. Mapper

Maps are the separate jobs which are used to convert input files into intermediate files. Different types of input files can be generated while converting intermediate files. Zero or number of output pairs can be mapped by the given input pairs. Each element present here is split into key-value pairs. The input files blocks count decides the map's quantity.

2.2.2 Reducer

Reducer gathers the output from the Map and combines those large sequences of data into a reduced set of sequenced data. Multi computing nodes data processing is measured easily with Map Reduce. A huge number of small files which are extremely smaller than the block size is not easy to store and handle efficiently by HDFS. Accessing small files involve lots of seeks and hopping between data nodes, which in turn, affects data processing. In name node's memory, all files, directory, and the blocks in HDFS is considered as an object. These objects are about 150 bytes in size each. If 10 Million small files are to be stored, then every individual file use individual block. This, in turn, uses 3 gigabytes of memory, with the huge amount of files, more storage space is needed to store the Metadata and its limit cannot be measured. Accessing small files through Name Node also slows down the process because storing each and every file in individual block results in more blocks and accessing these take more time. In MapReduce, A block of data is processed in Map at a time. A large number of small files occupy many blocks – which mean lots of tasks, and lots of records by Application Master. This will reduce the overall cluster performance compared to large file processing.

2.2.3 Optimized MapFile based Storage for Small files Approach

Small files problem being a challenge in Hadoop, a most frequently used approach is to merge the small files to form bigger ones. This reduces the number of files and for each file; an index is created while WebGIS technique is used to compress data. The small files are evenly combined with large files with Opti-

mized MapFile Storage for Small files (OMSS) to solve small files problem and to use memory efficiently and reduce the processing time. Small files are combined to form large files up to the block size (cut off point). While merging large files if it crosses block size (cut off point), then that file is removed from the large file and an internal fragmentation will occur in the merged file.

The caching methodology is used to cache local index file and correlated files. When a file is requested which is not in the cache, then the client sends a request to NameNode for file Metadata. After getting Metadata, Client is connected to data node where the dataset is present. The local index file is read using the offset and length, the required file is taken out of the block and is provided to the client. Small files are placed in merging queue up to the block size. Various such file queues are generated and they are merged and converted to Map files.

2.2.4. Different Techniques to manage small files

i) Hadoop Archive – Hadoop Archive (HAR) is an archiving technique to merge small files into HDFS blocks to improve the efficiency. HAR file access takes in main memory. It reduces frequent access of NameNode and minimizes Map operations, which increase the performance. The disk space occupied by HAR files is same as an original file because it holds the copy of the original file. Archives are immutable. If Archive is created once, then addition and deletion of files are not possible. To add and remove files again new Archive should be created. Two index-file read operations are required for accessing a file. Reading files in HAR is efficient, but it is slow when compared with HDFS due to two index read operations.

ii) Sequence Files – The filename is normally used as a key in sequence file and the respective contents are stored as the value. One program is enough to place a large number of files in a Single sequence file. Each file operation can be carried out one by one using a map-reduce technique. The advantage of using sequence files is dividing the data sets. MapReduce splits the files into pieces and process takes place on divided files individually. The sequence files allow compression. Fig 2 depicts the Sequence File storage as key-value pairs.

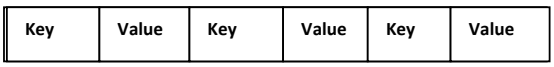

Fig 2 Sequence File Layout

To convert existing data into Sequence Files is a slow process. Data queue is used to write the data into a Sequence File instead of writing in intermediate small files. Accessing time is reduced when the files are stored in sequence files.

3. Small files storage problem in Hadoop – A review

The authors of [1] developed a Scheme by creating an index for each small file which is placed in a cluster while closely placed files are combined into a big file up to the size of blocks. The authors of [3] presented a method to merge small files which are stored separately in HDFS in various blocks, Archive file concept

is used to merge files and to improve the access performance. NHAR uses Index File Approach in Hashing Mechanism. Merging concepts by using the MapFile concept to reduce the memory size occupied by individual small files separately by merging them to reduce memory for better performance has been developed by [4].

The authors of [5] have proposed Merging file concept using Sequential file Technique. In this approach, while merging files cut-off point is analyzed to find the size and merging place of the files. This is a challenging task in this approach. Enhancement is needed to find the cut-off point in this technology for efficient results. [6].proposed Horton Sandbox Application to handle Hadoop data and sequential file Application used to sort the data before merging smaller files to use the memory efficiently. Merging files is done using Logic file name (LFN) using an Index file. Memory storage is reduced by combining related files is using Archive mechanism HAR. [7] proposed to increase the efficiency in accessing small files by merging all up to the block size using SFS (small file server) which acts as middleware and uses LRU for managing the cache.

The authors of [8]described the concept to merge the distributed small files using HAR concept with the Optimized MapFile based storage of small files. Internal fragmentation is reduced; hence Memory is utilized in an efficient manner. Files are merged to utilize the block size fully. Analyzing the cutoff point of file size is a difficult process in distributed files. If the merged files are stored in a block completely then there are no issues, but in merge files, if any one of the file sizes exceeds the block when combining it with other files, then that file is removed from the block. Then internal fragmentation takes place and this, in turn, takes more time. Thus, finding the Cut-off point before merging the files using file merging algorithm is of utmost importance.

TABLE 1
Summary of research on Small Files Problem handling in Hadoop and MapReduce

SL.NO	REFERENCE	ALGORITHM/ TECHNIQUES	DATA SET	RESULTS / OBSERVATION	PROPOSED FUTURE ENHANCEMENTS
1	[1]	Index file Approach.	Social media text files	The file number is reduced to access data	More focus needed to optimize Processing time
2	[9]	New Hadoop Archive with Indexing technique	Data files	Memory storage is reduced	Better performance should be found.

6

3	[10]	Merging technique using Map file	Files stored in clusters	File size reduced to improve performance	Improve performance on image file storage.
4	[11]	Indexing and Merging Techniques	Various Small files	File accessing speed increased	Improve processing and storage efficiency
5	[6]	Mapper and Reducer	Weather data	Achieves good effect	To merge image files efficiently using this technique
6	[14]	NHAR technique is improved with index representation	Structured and unstructured data	File merged and performance improved	Enhancement on NHAR with an index file
7	[5]	Sequential file Approach	Datasets	Reduces memory size	Cut off point should be analyzed to improve existing performance
8	[15]	Histogram equalization technique	Generate key-value pair of Map file function	Mapper function can perform adequately	Efficient and accurate Algorithm to be developed
9	[8]	Map file Technique & File Merging Algorithm	Structured and unstructured data	Memory consumption reduced.	Limit fragmentation for more enhancement
10	[16]	EHDFS is used to combine correlated files to utilize the memory in a good manner.	Groups of data stored in the cluster	Improved performance	Enhancement in the framework is needed for better improvement

3.2 Discussion

Table-1 summarizes the techniques and concepts used to solve the small files generated by various sources because small files occupy huge memory space by consuming individual blocks for each and every file and take more time to process da-

tasets in Hadoop cluster. In Hadoop system, HDFS is responsible for fetching data. The block size is 128MB the remaining free space in all the blocks is left unused. Because of this number of blocks use huge memory space which slows down the process of application.
The base concept used to overcome this problem is merging the small files using techniques like Archive files, a sequential file, and NHAR. Internal Fragmentation, Indexed file approach, Map file concept, and Histogram equalization concept is combined with these techniques to improve performance. Job scheduling Algorithms, File merging algorithms were generated to arrange the related files, then correlated files are merged to minimize the memory space. By this unused space in memory are utilized and performance speed also increased.

3.3 Future research directions and research gaps identified from the review

One of the limitations found from the above-discussed literature is Internal fragmentation. By Internal fragmentation, memory space is reduced, but the time taken for fragmentation process is more. Therefore the accessing time can't be reduced. And this needs improvement. Related files are combined mostly to reduce the space, but all files in the cluster may not related ones. HDFS supports structured and unstructured files, some unrelated files may be present. Implementing file merging in unrelated files is challenging one. Analyzing Cutoff size of files before merging may be a solution for internal fragmentation problem. It will reduce the frequent access of the NameNode for the files. Overall from the review, it is understood that the time and space efficiency to deal with small files is to be improved.

4. Conclusion

In this paper, we discussed Merging concepts to merge small files to reduce the memory size occupied by those files. Hadoop archive techniques and sequential file access techniques are used to improve Mapfile and to provide optimized accessing methods to enhance the data stored in Hadoop clusters. Some improvements are required to access files from Hadoop. In the future, the Enhanced technique in Archive file will provide more efficiency in accessing and storing data. Before merging various files its file size identification will improve the merging process and memory allocation efficiency. To access small sized image files using the above techniques is a future enhancement in this concept.

References

[1] Z. Gao, Y. Qin, and K. Niu, "An effective merge strategy based hierarchy for improving small file problem on HDFS," in *Proceedings of 2016 4th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2016*, 2016, pp. 327–331.

[2] D. P. Acharjya and K. Ahmed, "A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 511–518, 2016.

[3] C. Vorapongkitipun and N. Nupairoj, "Improving performance of small-file accessing in Hadoop," in *2014 11th Int. Joint Conf. on Computer Science and*

Software Engineering: "Human Factors in Computer Science and Software Engineering" - e-Science and High Performance Computing: eHPC, JCSSE 2014, 2014, pp. 200–205.

[4] P. Gohil, B. Panchal, and J. S. Dhobi, "A novel approach to improve the performance of Hadoop in handling of small files," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2015, pp. 1–5.

[5] B. Gupta, R. Nath, G. Gopal, and K. K, "An Efficient Approach for Storing and Accessing Small Files with Big Data Technology," *Int. J. Comput. Appl.*, vol. 146, no. 1, pp. 36–39, 2016.

[6] R. Manjunath, Tejus, R. K. Channabasava, and S. Balaji, "A Big Data MapReduce Hadoop distribution architecture for processing input splits to solve the small data problem," in *Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016*, 2017, pp. 480–487.

[7] Y. Huo, Z. Wang, X. Zeng, Y. Yang, W. Li, and C. Zhong, "SFS: A massive small file processing middleware in Hadoop," in *18th Asia-Pacific Network Operations and Management Symposium, APNOMS 2016: Management of Softwarized Infrastructure - Proceedings*, 2016, pp. 3–6.

[8] S. Sheoran, D. Sethia, and H. Saran, "Optimized MapFile Based Storage of Small Files in Hadoop," in *Proceedings - 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017*, 2017, pp. 906–912.

[9] K. Elakkia and P. Narendran, "Survey of Medical Image Segmentation Using Removal of Gaussian Noise in Medical Image," *Int. J. Eng. Sci. Comput.*, vol. 6, no. 6, pp. 7593–7595, 2016.

[10] T. Hong, C. Chen, J. Huang, N. Lu, L. Xie, and H. Zareipour, "Big Data Analytics for Grid Modernization," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2395–2396, 2016.

[11] G. L. Yao, "A Survey on Pre-Processing in Image Matting," *J. Comput. Sci. Technol.*, vol. 32, no. 1, pp. 122–138, 2017.

[12] Y. Du, A. Song, L. Zhu, and W. Zhang., "A Mixed-Type Registration Approach in Medical Image Processing," in *International Conference on Biomedical Engineering and Informatics, 2009. BMEI'09. 2nd*, 2009, pp. 1–4.

[13] Y. Wang, J. Zheng, H. Zhou, and L. Shen, "Medical Image Processing by Denoising and Contour Extraction.," in *Information and Automation, 2008. ICIA 2008. International Conference on*, 2008.

[14] S. Bende and R. Shedge, "Dealing with Small Files Problem in Hadoop Distributed File System," *Procedia Comput. Sci.*, vol. 79, pp. 1001–1012, Jan. 2016.

[15] D. Goutam, S. Gupta, and S. Deepti, "Big Data Analytics: Image Enhancement Based Approach," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 6, pp. 570–573, 2016.

[16] T. Gupta and S. S. Handa, "An extended HDFS with an AVATAR NODE to handle both small files and to eliminate single point of failure," in *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, 2015, pp. 67–71.