A Review on Small Files in HADOOP

A Novel Approach To Undestand Small Files Problem In Hadoop

AVANTI KHARE
Dept. of Information Technology
Sreenidhi Institute of Science and Technology
Hyderabad, India

Prof. Dr. B. INDIRA
Dept. of Information Technology
Sreenidhi Institute of Science and Technology
Hyderabad, India

Abstract— **Hadoop is an open source data management system designed for storing and processing large volumes of data, minimum size being 64MB. Storing and processing of Small Files smaller than the minimum block size cannot be efficiently handled by hadoop because Small Files results in lots of seeks and lots of hopping between the datanodes. A survey on the existing literature has been carried out to analyze the effect / solutions for the Small Files problem in hadoop. This paper presents the same and lists many effective solutions for this problem and further this paper says that there is a need to carry out lot of research on small file problem in order to attain effective and efficient solutions.**

Keywords- **Hadoop; HDFS; Small Files; Datanode;**

I. INTRODUCTION

Hadoop is an open source data management system and is a combination of Distributed file system and Analytic Algorithms. It consists of two core components:

First is HDFS (Hadoop Distributed file system) which helps to save the entire files and designed for processing large volumes of data, with a default size of 64MB [1]. Hadoop block size is by default set to 64, 128, or 256MB and so on, if the file size is less than 75% of default size then those files are considered as Small Files [16]. If many Small Files are stored then reading through Small Files will cause lots of seeks and lots of hopping from one datanode to other datanode for its retrieval, thus leading to inefficiency in data accessing patterns.

Second is MapReduce which help to do analysis on the entire set of the file, it usually processes a block of input at a time i.e. by default FileInputFormat. As you have lots of Small Files, then each time when task is processed it will process very little input causing extra logging of files [2, 3].

HDFS is a bucket where you can dump the data; MapReduce is used for data processing purpose. Hadoop is not really as that of database where you can store the data and pullout data and having no queries involved in it as that of SQL. Databases are centralized systems that use multiple machines. The entire work is divided between one or more systems and all the data processing software is housed on another server. [6] The way of utilizing RDBMS and Hadoop cloud storage includes two levels of storage modes, front end uses database for data accessing, and back end uses hadoop storage for storing large database files. Hadoop is more as that of data warehousing system, this is the reason it needs MapReduce for processing the data. [15]. Small Files are produced from larger logical files i.e. they are pieces of larger files which are produced during real time data collection and are directly copied into hadoop. HDFS supports appending i.e. adding or attaching files and very common way of saving files in hadoop is to write them in the form of chunks into HDFS. In some cases, files may be inherently small. The reason why companies produce Small Files is due to collection of real time data, system generates files which are smaller i.e. less than 75% of the default size is copied directly into hadoop.

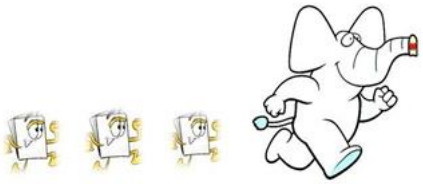

Fig 1. Small files in hadoop

II. SMALL FILES IN HADOOP: PROBLEMS AND SOLUTIONS

Why small file problems exist in hadoop?

There are two primary reasons namely NameNode memory management and MapReaduce Performance.

A. NameNode memory management problem:

If you have some millions of files and each file requires one block, NameNode may need some GB of memory. NameNode has to read the data of every file from cache, reading such GB of data will cause delay during the start-up time. To avoid the delay reduce the number of Small Files on your clusters [4]. Two solutions for solving memory problem is by Hadoop Archive Files (HAR) and Federated NameNode.

B. HAR files:

These files are produced from the files that exist in HDFS. It reduces the NameNode space footprint for Small Files, processing and accessing, which leads to less efficiency. HAR files are stored on disk

randomly, reading such file requires index access i.e. to find the location of file and finding HAR file. Reading these files is slower task than reading from HDFS. HAR files can solve the NameNode memory problem, but it reduces the processing performance. HAR Files can be a good solution if the files are not frequently access. [18] NHAR (New Hadoop Archive mechanism based on HAR to improve the memory utilization and efficiency of accessing Small Files. They tried to extend the capability which allows additional file insertion in existing files. Resulting improvement on small I/O performance and additional file storage to existing archive files, archiving time is reduced nearly 85%.

C. Federated NameNodes:

It allows having multiple NameNodes in cluster, storing subset of objects. This reduces need of storing subset of objects metadata on the single machine. NameNode knows about the particular object to be used, if you want to get any file then you must know which NameNode you have to use. Federation does not solve performance problem, it adds unnecessary complexity in hadoop installation and administration. If it is considered for solving small file problem, then more mechanisms are required for solving the problem [17]

D. MapReduce performance:

As discussed previously the performance will be affected due to combination of random disk I/O and managing many map tasks which are the major performance limiting factor in MapReduce. To solve this problem use hadoop append capabilities, file sequencing, Hive configuration settings, etc [17]

III. LITERATURE REVIEW

Grant Mackey, Saba Sehrish, Jun Wan Stated an approach for storing Small Files in HDFS efficiently and improving the memory utilization for metadata. Client is assigned with quota i.e. the available amount of files that are allowed to access and use from user directories. Depending upon maximum number of files per directory and memory for user directory. Here 'harballing' method is considered which collects Small Files in single large file. Provides the functionalities to complete a new job without killing by the JobTracker due to quota mechanism. The result of this experiment shows better functionality and efficient usage of HDFS. Also shows that there can be reduction in metadata footprint of main memory by factor of 42 of its original size. [4]

Bo Dong, Qinghua Zheng, Feng Tian, Kuo-Ming Chao, Rui Ma, Rachid Anane in journal they described how to store and access the Small Files. The issue regarding to the identifying the cut off points between the large files and Small Files i.e. "To what extent the Small Files are small". Next is the classification of files i.e. pieces of large files and the files that are originally small. The sequence file problems are that it does not support data deletion or updated. They adopted file merging and grouping strategy which leads to reduction in disk arm movements when requested. They described the three level prefetching and caching strategies for the purpose of improving the accessing efficiency. This approach can reduce the per- file metadata interactions with server, decrease the I/O cost of fetching the files from disk, reduce the time taken for transferring the files [5]. Another paper states that adopted FMP(File merging and prefeching) technique, in which merged files and index files are stored on datanode, which results in better efficiency in storing and processing Small Files in HDFS[13]

Sankalp Mitra, Suchit Bande, Shreyas Kudale, Advait Kulkarni, Asst. Prof. Leena A. Deshpande in journal they illustrated the use of MapReduce to implement FP growth algorithm. Their experiment shows that IPFP (Improved parallel FP growth) is a feasible solution for speedup and mining efficiency, avoid memory overflow and reduction in I/O overhead. The IPFP growth is migrated to MapReduce environment. There are still some defects in handling lots of Small Files datasets. [8]

Kashmira P. Jayakar, Y.B.Gurav They discuss about the EHDFS (Extended Hadoop Distributed file system**)** and HDFS. In EHDFS files are stored in a single file called combined file on Datanode. Indexing mechanism is to access the files and reduces the load on NameNode and I/O performance. EHDFS improves accessing and storing efficiency and reduces time required for processing the Small Files [7]. EHDFS includes operations like file extraction, file prefetching and file mapping. It reduces the number of node failure, thus reducing performance lost in the system. Burden on memory and time required for processing is reduced [10]. The paper proposes the Optimized hadoop mechanism, the experiment result shows that processing performance of Small Files is reduced up to 90.83% and memory utilization of NameNode for storing metadata of files [12]. A novel approach for handling Small Files considering the file size of range 32KB – 4096KB, cluster settings and varying the ranges of file size, results in reducing by 10% the storage required for metadata in main memory of NameNode [20]. Files are stored in index file in form of key value pairs, the files should be judged and then uploaded to HDFS cluster. Improved HDFS has lower memory consumption as compared to the original HAR and HDFS [21]

Priyanka Phakade, Dr. Suhas Raut When HDFS client requests file in HDFS. NameNode permits for storing files in HDFS. Then NameNode combines the file in a single split which will be acting as an input to map task, further output is given to multiple

reducers where reducers will give sorted merge output. Thus number of MapTask is reduced by reducing the processing time [9]

Linthala Srinithya, Dr. G. Venkata Rami Reddy Journal states analysis and performance of HDFS and LFS (Local File System) with respect to the read and write approach. They setup a hadoop cluster and design an interface which defines the size of file, time taken for uploading and downloading. The resulted that write performance for Small Files are not having much difference between the HDFS and LFS, while in case of large files they resulted in performance difference. That LFS shows better efficiency for handling Small Files, while in case of large files it will take lot of time for reading and writing. The dotted line represents the average of three tests for writing the files in LFS and solid lines represents performance of HDFS [11]

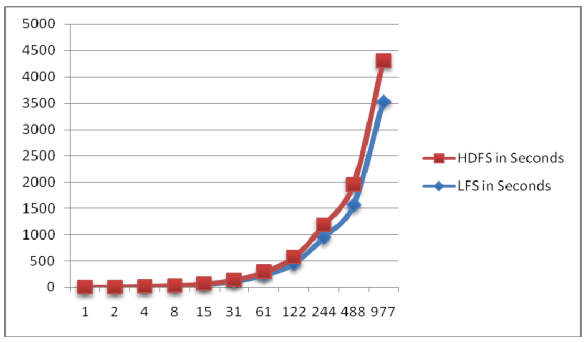

Fig 2. Comparison between HDFS and LFS files

Fang Zhou, Hai Pham, Jianhui Yue, Hao Zou, Weikuan Yu Stated an optimized MapReduce framework for Small Files problem, two techniques SF Layout (Small File Layout) and CMR (Customize MapReduce). Their experiment shows that SFMapReduce decrease memory pressure on NameNode, provides better retrieval throughput and loading. Also stated that nearly 14.5 – 20.8 times improvement was achieved by SFMapReduce as compared to HAR Layout. [14]

Passent M EIKafrawy, Amr M Sauber, Mohamed M Hafez Illustrated Enhanced HDFS architecture which they called as HDFSX, where the architecture would support large files and Small Files, this architecture solves the problem related to Small Files in HDFS such as NameNode memory usage, Centralized point failure and NameNode overhead. But this is not suitable for real environment. In future would like to apply some changes in hadoop codes and libraries [22]

Yingchi Mao, Bicong Jia, Wei Min and Jiulong Wang Stated the optimized scheme through SFIM (structured index file merging) by two level file index, prefetching and caching strategy, structured metadata storage, to reduce the I/O operations and improving the access efficiency of Small Files. Their experimental results states that the access time is reduced up to 50-80%. SGIM is proposed to improve storage performance, which out performs sequence file and HAR files [19]. Further some advancement made to HDFS in different paper stating that files should be sorted depending on their particular extension and merging into zip files. For reading file cache is established, thus resulting a program to read easily and quickly, reducing time of reading by 92% and writing by 80% [23]. In other paper depending on novel technique they stated that depending on file merge, caching and correlation approach it resulted that accessing efficiency was improved up to 88.57% compared to HAR solution [24].

IV. CONCLUSION

The objective of writing this paper is to summarize an overview of Small Files occurring in hadoop, and less focusing about the technology described in the paper or journals. The purpose is to do analysis on the problems occurring due to large number of Small Files in hadoop and to state the solutions proposed to overcome the Small Files problem. The paper discusses about the technologies used and their experimental results to improve the accessing and processing efficiency of Small Files. Also tried to discuss some of the application related to the subject.

V. REFERENCE

[1] "Hadoop: The Definitive Guide" Tom White

[2] The Small Files Problem by Tom White February 2, 2009

[3] "Apache Hadoop" http.apache.org/2009

[4] "Improving Metadata Management for Small Files in HDFS". Grant Mackey, Saba Sehrish, Jun Wang, University of Central Florida, Orlando. 978-1-4244-5012-1/09/$25.00 ©2009 IEEE

[5] "An optimized approach for storing and accessing Small Files on cloud storage". Bo Dong, Qinghua Zheng, Feng Tian, Kuo-Ming Chao, Rui Ma, Rachid Anane MOE. Journal of Network and Computer Applications 35 (2012) 1847–1862

[6] "Hadoop: What It Is And How It Works" Brian Proffitt May 23, 2013

[7] "Efficient Way for Handling Small Files using Extended HDFS". Kashmira P. Jayakar, Y.B.Gurav, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.6, June- 2014, Pune University, India

[8] "Efficient FP Growth using Hadoop-(Improved Parallel FP-Growth)". Sankalp

Mitra, Suchit Bande, Shreyas Kudale, Advait Kulkarni, Asst. Prof. Leena A. Deshpande. International Journal of Scientific and Research Publications, Volume 4, Issue 7, July 2014 1 ISSN 2250-3153, VIIT, Pune

[9] "An Innovative Strategy for Improved Processing of Small Files in Hadoop". Priyanka Phakade1, Dr. Suhas Raut, International Journal of Application or Innovation in Engineering & Management (IJAIEM) Volume 3, Issue 7, July 2014, Solapur

[10] "Managing Small Size Files through Indexing in Extended Hadoop File System". K. P. Jayakar, Y. B. Gurav. International Journal of Advance Research in Computer Science and Management Studies Volume 2, Issue 8, August 2014 University of Pune India

[11] "Performance Evaluation of Hadoop Distributed FileSystem and Local File System". International Journal of Science and Research (IJSR) ISSN +Volume 3 Issue 9, September 2014 Linthala Srinithya, Dr. G. Venkata Rami Reddy, JNTUH, Hyderabad, India

[12] "Improving the Performance of Processing for Small Files in Hadoop: A Case Study of Weather Data Analytics". Guru Prasad M S1, Nagesh H R 2, Deepthi M. Guru Prasad M S et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6436-6439

[13] "An Optimized Storing and Accessing Mechanism for Small Files on HDFS". Shrikrishna Utpat, K. A. Dehamane, Srinivasa Kini. Maharashtra, India. IJARCSSE, Volume 5, Issue 1, January 2015

[14] "SFMapReduce: An Optimized MapReduce Framework for Small Files". Fang Zhou, Hai Pham, Jianhui Yue, Hao Zou, Weikuan Yu, Auburn University 978-1-4673-7891-8/15/$31.00 ©2015 IEEE

[15] "Research on the Small Files Problem of Hadoop". Xiao Jun Liu, Chong Peng, Zhi Chao Yu, Huanggang Normal University, Hubei Huanggang, China. International Conference on Education, Management, Commerce and Society (EMCS 2015)

[16] Working with Small Files in hadoop – Part 1. Chris Deptula, Feb 11, 2015

[17] Working with Small Files in hadoop – Part 2. Chris Deptula, Feb 18, 2015

[18] "Performance enhancement for accessing small-files in hadoop". Varun Pal, Mrs. D. Hemavathi, SRM University, Chennai, India. IJERSS Volume 2 | Issue 4 APRIL 2015

[19] "Optimization Scheme for Small Files Storage Based on Hadoop Distributed File System". Yingchi Mao, Bicong Jia, Wei Min and Jiulong Wang, Hohai University, China. International Journal of Database Theory and Application Vol.8, No.5 (2015)

[20] "Improving Access Efficiency of Small Files in HDFS". Monica B. Bisane, Asst.Prof. Pushpanjali M. Chouragade, Amravati, India. International Journal of Scientific & Engineering Research, Volume 7, Issue 2, February-2016 ISSN 2229-5518

[21] "An Improved HDFS for Small File". Liu Changtong, Huazhong University, China. ISBN Jan. 31 ~ Feb. 3, 2016.

[22] "HDFSX: Big Data Distributed File System with Small Files Support". Passent M EIKafrawy, AmrM Sauber, Mohamed M Hafez, Menofia University, Egypt. 9781509028634/16/$31.00 ©2016 IEEE

[23] "Optimization Scheme for Storing and Accessing Huge Number of Small Files on HADOOP Distributed File System". L. Prasanna Kumar, Sampathirao Suneetha. Andhra University, Visakhapatnam. IJRITCC | February 2016

[24] "An Efficient Approach for Storing and Accessing Small Files with Big Data Technology". Bharti Gupta, Rajender Nath, Girdhar Gopal, Kartik. Kurukshetra University, Haryana, India. International Journal of Computer Applications (0975 – 8887) Volume 146 – No.1, July 2016.

