Efficient Resolution for the NameNode Memory Issue for the Access of Small Files in HDFS

Deeksha S P[1], R Kanagavalli[2], Dr. Kavitha K S[3], Dr. Kavitha C[4]

[1]*PG Student, Department of CSE, Global Academy of Technology, Bengaluru, Karnataka, India*
[2]*Associate Professor, Department of CSE, Global Academy of Technology, Bengaluru, Karnataka, India*
[3]*Professor, Department of CSE, Global Academy of Technology, Bengaluru, Karnataka, India*
[4]*Professor & HOD, Department of CSE, Global Academy of Technology, Bengaluru, Karnataka, India*

---***---

Abstract - Hadoop Distributed File System (HDFS) was initially designed for storing, processing and accessing huge files. But large number small files will not be processed and accessed efficiently. This paper introduces an access improvement approach for HDFS small files by using Map File called TLB-Map File. This TLB-Map File combines many small files into large files by Map File component to lessen NameNode memory utilization and include TLB table in DataNode, and to enhance access efficiency of small files. The method first merges small files into large file and stores it in HDFS. Then the retrieval frequency of each file is accessed from logs in the system, and stored in TLB table. The information of the location of the block where the small file is stored is also filled in TLB Table. This Table is regularly updated. Thus the TLB-MapFile Approach efficiently resolves the retrieval issues of small files by priory fetching the files based on table.

***Key Words*: HDFS, small files, TLBMapFile, retrieval, priory fetching.**

1. INTRODUCTION

Hadoop is an open source software framework used for storing, accessing, and processing of huge datasets in distributed environment. Hadoop is built on clusters of commodity hardware. Each server in single machine stores large data and provides local computation which can be extended to thousands of machines. It is derived from Google's file system and MapReduce[1]. It is also suitable to detect and handle failures. It can be used by the application which processes large amount of data with the help of large number of independent computers in the cluster. In Hadoop distributed architecture, both data and processing are distributed across multiple computers.

Hadoop consists of Hadoop Distributed File System (HDFS) which is used for storing data and MapReduce Programming model which is used for processing the data. HDFS is a Java-based Hadoop Framework which provides scalable, distributed and portable form of file system. HDFS is garble tolerant and is highly scalable. HDFS has a Master-Slave Architecture. It has a single NameNode and multiple DataNodes. NameNode stores the file system Metadata and connects clients to files. DataNode stores the actual data and is responsible for giving response to read and write requests of clients.

It can be seen that the structure with only one NameNode simplifies the file system, but HDFS is at first designed for storing and processing huge files. So the small files which can be saved in HDFS will consume a more memory in NameNode. Every metadata object occupies about one hundred fifty bytes of memory [1], assuming that the quantity of small files reaches a thousand million, the metadata holds memory almost 40G. Similarly, the mass of small files will cause a huge number of jumps and looking of to and fro in DataNode and time taken to access can be very high.

Based on MapFile, this paper provides a new small file accessing optimization scheme: TLB-MapFile. For this approach, small files are merged into large files, then the data of small files that being high-frequency accessed is obtained via access audit logs. Subsequently, the mapping data between block and small documents are made to be saved in TLB and are up to date frequently. When a file is accessed once more, the mapping data is retrieved inside the TLB table, then the mapping data of related files also are obtained. This can be done via pre fetching mechanism and this gains speedy prefetching small files.

2. LITERATURE SURVEY

Small file is the file whose size is less than the HDFS default block size which is 64MB. With a view to enhance the access efficiency of small files, a few scholars have carried out related studies.

With the intention to quickly discover small files, a common strategy is to merge small files into big ones via merge and index mechanisms.

HDFS distributed file storage system comes with a small handling mechanism: Hadoop Archive (HAR) [2], and SequenceFile [3-4] .

Hadoop Archive (HAR) is specifically used to archive files in HDFS for decreasing memory utilization of NameNode.

However, correlations among documents are not taken into consideration during the system of archiving. HAR archive can be generated by using using the Hadoop archive command, via which small files are archived into HAR files. This technique can lessen the memory consumption of NameNode. The main disadvantage of this method is that, while HDFS reads the HAR file, system needs to have a look at index files and the statistics file. So the access efficiency of small files is very low.

A Sequence file SequenceFile is a specialized key value data structure that acts as a container for small files[4]. It takes a long time to transform small files into a SequenceFile. furthermore, to discover a specific key, we ought to search entire series record. For that reason, the access performance of files degrades. When reading the massive small files, HDFS can only retrieve the file content sequentially and the reading efficiency is still low.

CombineFileInputFormat is a new input format and the directory including a plurality of small files is used as one input, instead of using a file as input. In order to improve the executive speed of MapReduce task, CombineFile-InputFormat merges multiple files into a single split and make each Mapper task can handle multiple data. In addition, the storage location of the data will be taken into account.

For Small files for some particular occasion, there are many improved methods and architectures have been proposed [5-8]. However, these methods are not much in use and they occupy more memory.

3. PROPOSED SYSTEM

TLBMapFile is MapFile mechanism where the small files are first merged in large files, and the mapping information about the block and the file are stored in table called TLB table which has fields such as fileID, blockId, file_key and offset. TLBMapFile accesses the frequency of small files from audit logs. The TLB table also stores the mapping data about the blocks and small file being frequently accessed. The entire Architecture is implemented in nine modules and the below figure describes the activities performed.

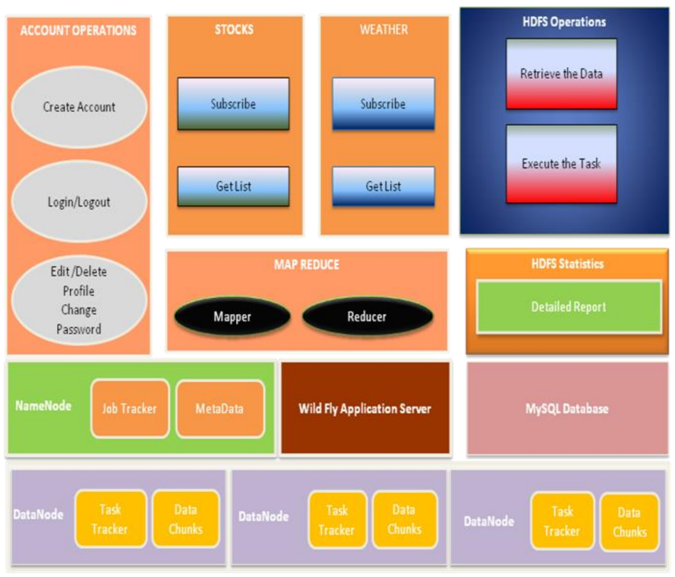

Fig -1: Proposed System Architecture

The architecture is split into following modules as shown in Fig-1: Data Access Layer, Account operations, Subscription, Data Pull Operation, Data Merge Operation, Statistics, and Background Tasks.

3.1 Data Access Layer:

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

3.2 Account Operations:

Account operations module provides the following functionalities to the end users:

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

3.3 Subscription:

Here, the end user will be able to subscribe for a new Stocks company or a new weather city and register it to our local mysql database. Our project will be guiding the end user in the subscription process by providing a link which helps them to find out the proper scrip code of a company for which they have to subscribe, along with that we also provide a link to the end user which finds the longitude and latitude of whichever city the customer wants to subscribe to.

3.4 Data Pull Operation:

Here, the end user will be able to pull the stocks data for the companies for which they have subscribed to, and, they will be able to pull the weather data for the locations for which they have subscribed to. Both the stocks data and the weather data will be provided as a web service for which our project will be making a Rest HTTP API call to retrieve the data. The stocks and weather data obtained from a web service call will be stored as a small file in the specific location inside HDFS. When the more number of users subscribes, large number of small files start getting stored in HDFS. Each of these large numbers of small files will be occupying an entire block in HDFS (128 MB by default in cloudera). This is called as Hadoop small files problem.

3.5 Data Merge Operation:

Here, the end user can execute the Data merge algorithm which will combine large number of small files into a single large file thus removing the small files issue in HDFS. The algorithm works by constructing a map with the keys as the names of all the small files and the value being the corresponding contents of the small file. This map is considered as a TLB Map file (Fast Table) because this helps us to retrieve the content of any small file just by knowing its name. This map will be constructed in JVM's in-memory and we will be persisting this map in the HDFS system using Serialization concept in Java.

3.6 Statistics:

This module helps the end user to realize the importance of the TLB Map file in removing the name node memory issue and Hadoop small file problem. It does so by displaying the statistics of the details of the small files stored in HDFS and the details of the merged file stored in HDFS. The details that are displayed here are the names of the small files, size of each of the small files, total memory blocks occupied etc.

3.7 Background task:

The background task here in our project will be responsible for pulling the data for the stocks and the weather cities for those subscribed by all the users of this project. Advantage of this module is, the end user need not execute the data pull operation every time he need the data, instead he can just create a background task that will be running all the time and automatically executes the data pull operation. This background task can be scheduled to execute at a specific time of the day by the end user. This task not only executes the data pull operation, but also executes the merge algorithm so that the result will be available to the end user instantly.

4. CONCLUSIONS

TLB-MapFile excavates the small file index being frequently visited inside the audit log and stores those indexs into the TLB table. TLB-MapFile is a brand new reading policy for small files in HDFS, which can successfully enhance the analyzing speed. In analyzing files, the technique firstly gains the file mapping indexes inside the TLB table, and directly locates the places of the small files and reads the small files. through which, the studying efficiency of small files is progressed.

5. FUTURE WORK

Integrate the system with the Hadoop Distributed File System which is provided as a service by the cloud hosting provider. Extend the solution to various other formats of the small files of the form PDF, XLS etc

REFERENCES

[1] Konstantin S, Hairing K, Sanyjy R, et al. The Hadoop Distributed File System[C]. Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). May 03-07, 2010: 1-10.

[2] C Vorapongkitipun, N Nupairoj. Improving performance of small-file accessing in Hadoop. Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on.

[3] Chang Xiao, Qiang Li.A novel approach to record file correlation and reducemapping frequency on HDFS based on ExtendHDFS.Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference.

[4] (Dong, Bo)[1,2] , Zheng, QH (Zheng, Qinghua)[1,2], Tian, F (Tian, Feng)[1] .An optimized approach for storing and accessing small files on cloud storage .JOURNAL OF NETWORK AND COMPUTER APPLICATIONS. DOI: 10.1016/j.jnca.2012.07.

[5] Dong Bo, Qiu Jie, Zheng Qinghua, et al. A novel approach to improving the efficiency of storing and accessing small files on Hadoop: A case study by PowerPoint files[C]. //IEEE International Conference on Services Computing, 2010: 65-72.

[6] Xuhui Liu, Jizhong Han, Yunqin Zhong, et al. Implementing WebGIS on Hadoop: a case study of improving small file I/O performance on HDFS[C]. //IEEE International Conference on Cluster Computing

and WorkShops. Piscataway, NJ: IEEE, 2009: 1-8.

[7] Liu Jiang, Bing Li, Meina Song. THE optimization of HDFS based on small files. Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on. 912 - 915, DOI: 10.1109/ ICBNMT. 2010. 5705223.

[8] S. Chandrasekar, R. Dakshinamurthy, P G Seshakumar, B. Prabavathy, Chitra Babu. A novel indexing scheme for efficient handling of small files in Hadoop Distributed File System. Computer Communication and Informatics (ICCCI), 2013 International Conference on. 10.1109/ ICCCI.2013.6466147.2013.