An Approach of Accessing Small Files on HDFS for Cloud Storage

Khin Su Su Wai, Julia Myint, Tin Tin Yee
University of Information Technology, Yangon, Myanmar
khinsusuwai@uit.edu.mm, *juliamyint@uit.edu.mm, tintinyee@uit.edu.mm*

Abstract

Hadoop distributed file system (HDFS) was originally designed for large files. When the large number of small files is accessed, NameNode often becomes the bottleneck. The access and storage efficiency is low for the mass small files. In order to solve this issue in HDFS, the approach of accessing small files on HDFS is proposed. In this approach, all correlated small files will be merged into a larger file based on the agglomerative hierarchical clustering mechanism to reduce NameNode memory space. Moreover, HDFS does not take the correlation among files into account and it does not provide any prefetching mechanism to improve the I/O performance. A prefetching mechanism will be proposed to improve the efficiency of accessing small files. This approach will provide small files for cloud storage.

Keywords- Hadoop distributed files system (HDFS), NameNode, Small files, prefetching, and hierarchical agglomerative clustering

1. Introduction

Cloud computing has become increasingly popular as the next infrastructure for hosting data and deploying software and services. Distributed file system is the foundation of cloud computing and provides reliable and efficient data storage for upper applications.

Today most popular storage system for cloud computing such as Google File System (GFS) and Hadoop Distributed File System (HDFS) are widely used and well known. However, HDFS is more light-weighted and open-source platform.

Hadoop is a software framework which is an open source that supports big data in distributed environment. Hadoop creates a cluster of machines and coordinates the work among them. It has two major components HDFS and Map Reduce. HDFS has master-slave architecture, with a single master called the NameNode and multiple slaves called DataNodes. NameNode manages the metadata and regulates client accesses. The metadata is maintained in the main memory of the NameNode to ensure fast access to the client, on read/write requests. DataNodes provide block storage and service read/write requests from clients, and perform block operations by contacting with NameNode.

The consumption of memory in NameNode is decided by the number of files stored in HDFS. Each file Metadata requires 150 bytes of space. DataNode is responsible for saving the real and replicated data. Each file is split into several blocks with the size of 128MB. These files are replicated in HDFS based on the configuration. The DataNode keeps on sending the heartbeat signal to the NameNode at regular intervals to indicate its existence in the system. The heart-beat consists of DataNode's capacity, used space, remaining space and some other information. Client can upload, download, update, and delete by contacting with NameNode.

A small file is a file whose size is less than the HDFS block size. Although the size of a file is less than the HDFS default block size, HDFS creates it as one block. When the large number of small files is stored, HDFS is inefficient because of high memory usage and unacceptable access cost. Hadoop does not provide optimal performance for small files processing. Furthermore, HDFS does not take the correlation files and it does not support any prefetching mechanism to improve the I/O performance. In this proposed approach, merging and prefetching will be presented to overcome small file problems in HDFS.

The rest of the paper is organized as follows. Section 2 is an illustration of related works about the proposed topic. Section 3 is the research methodology of the proposed system. Section 4 proposes the proposed system. Section 5 presents the expected result. Section 6 concludes the paper.

2. Related Works

MENG Bing and et al. [1] provided a solution to reduce NameNode memory consumption, by TLB-Map File. TLB-MapFile merges massive small files into large files by MapFile mechanism to reduce NameNode memory consumption and add fast table structure (TLB) in DataNode, and to improve retrieval efficiency of small files. A challenging work is to build up suitable TLB refresh cycle.

Zhipeng Gao and et al. [2] defined Logic File Name (LFN) and proposed the Small file Merge Strategy Based LFN (SMSBL). SMSBL is a new idea and a new

perspective on hierarchy; it improves the correlation of small files in the same block of HDFS effectively based different file system hierarchy. This system solved small file problem in HDFS and has appreciable high hit rate of prefetching files. The proposed system needs to combine SMSBL with other great solution to improve performance of HDFS.

A new structure for HDFS (HDFSX) is presented by Passent M EIKafrawy and et al. [3] to avoid higher memory usage, flooding network, requests overhead and centralized point of failure of the NameNode. In the other word, the performance analysis of the systems is needed to be developed.

Tao Wang and et al. [4] defined a user access task. The correlations among the access tasks, applications and access files are constructed by the improved PLSA, and the research object is transferred from file-level to task-level. Then, an effective strategy is proposed to improving small file problem in distributed file system. The strategy merges small files in term of access tasks and selects a prefetching targets based on the transition of the tasks. This strategy reduces the MDS workload and the request response delay.

Parth Gohil and et al. [5] focused on a MapReduce approach to handle small files. This approach improves the performance of Hadoop in handling of small files by ignoring the files whose size is larger than the block size of Hadoop. This also reduces the memory required by NameNode to store these files. So, it requires very less amount of memory than original HDFS but it requires some more memory than HAR and Sequence.

Extended Hadoop Distributed File System (EHDFS) is used by Tanvi Gupta and et al. [6]. This paper focuses on increasing the 'efficiency' of the indexing mechanism for handling 'Small files' on HDFS. This also added the concept of 'Avatar node' that eliminates the single point of failure.

Kyoungsoo Bok and et al. [7] proposed a distributed cache management scheme that considers cache metadata for efficient accesses of small files in Hadoop Distributed File Systems (HDFS). The proposed scheme can reduce the number of metadata managed by a NameNode. Many small files are merged and stored in a chunk. It also reduces unnecessary accesses by keeping the requested files using clients and the caches of data nodes and by synchronizing the metadata in client caches according to communication cycles.

Yonghau Huo and et al. [8] used additional hardware named SFS (Small File Server) between users and HDFS to solve the small file problem. This approach includes a file merging algorithm based on temporal continuity, an index structure to retrieve small files and a prefetching mechanism to improve the performance of file reading and writing.

3. Background Theory

3.1. Hierarchical Clustering

Hierarchical clustering is a widely used data analysis tool. The idea is to build a binary tree of the data that successively merges similar groups of points. This tree provides a useful summary of the data. Hierarchical clustering only requires a measure of similarity between groups of data points. The hierarchies also involve ordering relations.

A hierarchical classification can be illustrated in several ways. The result of hierarchical clustering is a tree-based representation of the objects, which is also known as dendrogram. The dendrogram is a multilevel hierarchy where clusters at one level are joined together to form the clusters at the next levels. This makes it possible to decide the level at which to cut the tree for generating suitable groups of a data objects.

In data mining and statistics, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types: agglomerative and divisive. In this system, the agglomerative hierarchical clustering is focused to merge many small files. A measure of dissimilarity between sets of observations is required in order to decide which clusters should be combined. In most methods of hierarchical clustering, this is achieved by use of an appropriate matric and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

3.1.1. Agglomerative Hierarchical Clustering: Agglomerative hierarchical clustering is a clustering algorithm that builds a cluster hierarchy from the bottom-up. It starts by adding a cluster for each of the data points to be clustered, followed by iterative pair-wise merging of clusters until only one cluster is left at the top of the hierarchy. The choice of clusters is decided to merge at each iteration base on a distance metric.

The dissimilarity values between one file and another have to be calculated in advance. In this paper, the Euclidean distance measure is used to cluster the related files. The clustering is based on distance matrix. Only the half of the matrix is needed because the distance between objects is symmetric. The Euclidean distance measure is:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2)} \quad (1)$$

This formula defines data objects i and j with a number of dimension equal to p. The distance between the two data objects d(i,j) is expressed as given the

above formula x_{ip} is the measurement of object i in dimension p.

An agglomerative clustering algorithm is described in the single-linkage clustering. The single-linkage clustering is the minimum distance between elements of each cluster. Two clusters are merged if there is at least one edge which connects them.

4. Proposed System

HDFS has been adopted to support the Internet applications because of its reliable, scalable and low-cost storage capability. It is a file system that supports for cloud storage. However, it does not present good storage and access performance when processing a huge number of small files. Firstly, all correlated small files will be clustered into a large file to reduce NameNode memory consumption. The small file is a file, whose size is less than 75% of default block size (128MB). Secondly, a prefetching mechanism will be introduced to improve the efficiency of accessing small files. The new HDFS structure will support a new approach. The proposed system architecture is shown in figure 1.

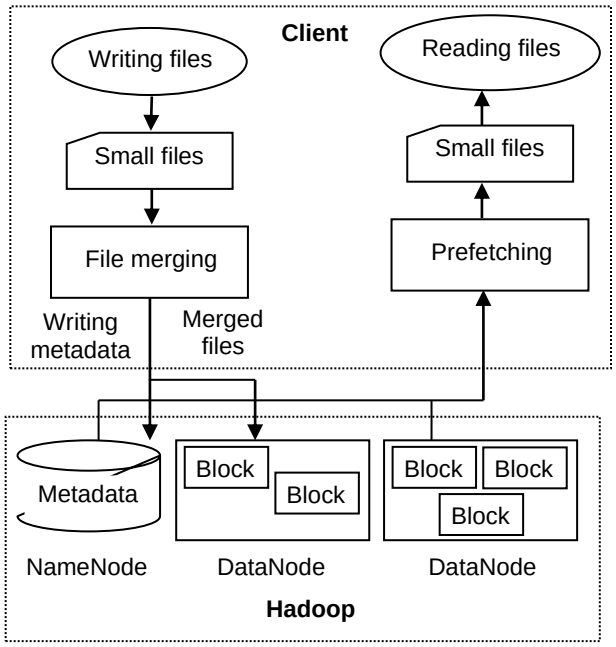

Figure 1. Proposed system architecture

4.1 File Merging Operation

File merging operations are carried out in processing layer. The related small files are merged into a large file. NameNode only maintains the metadata of merged files and does not store the original small files. File merging reduces the length of the metadata of many small files. The data access and latency can be improved. Algorithm of file merging is as follow.

Algorithm of file merging

Input: The number of small files
Output: The number of clusters hierarchy
Method:

(1) Calculate the Euclidean distance matrix between
Small files
(2) Repeat
(3) Merge a pair of file with single-linkage based on the distance matrix
(4) Update the distance matrix
(5) Until size of cluster is greater than or equal to default block size

In this approach, the files will be ignored to merge as they are already larger than the threshold value. The default threshold for this system is set to (0.75) 75% of default block size (128 MB). If the user accesses the files, the system will check the size of file. If the file is a small file, the system will put it in the hierarchical structure. The small files are nested in large cluster of files. These larger clusters are joined until its size is less than the default block size. Thus, a small file belongs to many clusters depending on its size and threshold value. Otherwise, the file is directly operated in the original HDFS.

The following table is the sample input to trace a hierarchical clustering of distances in sizes between files.

Table 1. Sample input size of small files

File	Size(MB)
S1	40
S2	10
S3	50
S4	20
S5	60
S6	30

The Euclidean distance matrix of small files in table 1 is shown in table 2.

Table 2. Euclidean distance matrix of small files

	S1	S2	S3	S4	S5	S6
S1	0	30	10	20	20	10
S2	30	0	40	10	50	20
S3	10	40	0	20	10	20
S4	20	10	20	0	40	10
S5	20	50	10	40	0	30
S6	10	10	20	20	30	0

According to the file merging algorithm, the sample input from table 1 will have two clusters output. The file merging dendrogram of small files is shown in figure 3.

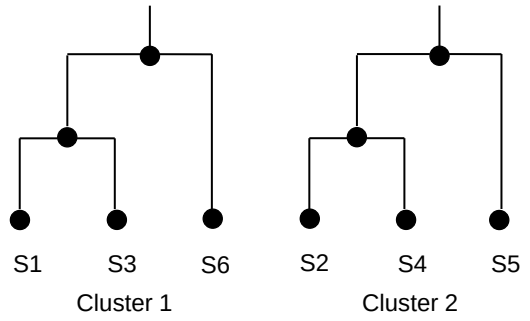

Figure 3. File merging dendrogram of small files

4.2 Prefetching

Prefetching scheme is widely used for improving the access efficiency. Prefetching can avoid disk I/O cost and reduce the response time by considering the access locality and fetching data into cache before they are requested.

The correlated small files are merged in a larger one file. When the user reads a small file, the system would need to find in hierarchical structure. Otherwise, the user has to read the file by connecting the DataNode. Prefetching schema helps in metadata caching and fetches small files. This technique optimizes fetching small files and reduces the communication cost when reading huge number of small files.

In the proposed paper, metadata and blocks of correlated files will be prefetched from NameNode and DataNodes respectively. The metadata prefetching reduces access latency on metadata server. The blocks prefetching is used to reduce visible I/O cost.

5. Expected Result

The proposed algorithm will be implemented and tested in simulated cloud environment. In the proposed paper, the strategy of merging small files will reduce the usage of metadata stored on NameNode. The processing time will also be optimized by comparing the original Hadoop and existing approach. Client will achieve better performance on accessing small files into HDFS.

6. Conclusion

In this paper, the small file problem of HDFS is focused. The memory space will be reduced to store the metadata of many small files in NameNode by the proposed agglomerative hierarchical merging approach. The number of small files will be eliminated in the HDFS by merging into a large file. The proposed method for small files will provide the infrastructure of Hadoop. As a future work, many experiments have to be done in order to get the efficiency of proposed merging algorithm. Proposed prefetching mechanism has to be verified as a future work to improve the access of massive small files.

7. References

[1] MENG Bing and GUO Wei-bin and FAN Gui-sheng, “A Novel Approach for Efficient Accessing of Small Files in HDFS: TLB-MapFile”, 2016 IEEE SNPD 2016, Shanghai, China, May 30-June 1, 2016.

[2] Zhipeng Gao, Yinghao Qin and Kun Niu, “AN EFFECTIVE MERGE STRATEGY BASED HIERARCHY FOR IMPROVING SMALL FILE PROBLEM ON HDFS”, 2016 IEEE.

[3] Passent M EIKafrawy, Amr M Sauber and Mohamed M Hafez, “HDFSX: Big Data Distributed File System with Small Files Support”, 2016 IEEE.

[4] Tao Wang, Shinhong Yao, Zhengquan Xu, Lian Xiong, Xin Gu and Xiping Yang, “An effective strategy for improving small file problem in distributed file system”, 2015 IEEE, 2015 2nd Internation Conference on Information Science and Control Engineering.

[5] Parth Gohil, Bakul Panchal and J. S. Dhobi, “A Novel Approach to Improve the Performance of Hadoop in Handling of Small Files”, 2015 IEEE.

[6] Tanvi Gupta and Prof. SS Handa, “An Extended HDFS with an AVATAR NODE to handle both small files and to eliminate single point of failure”, 2015 IEEE, 2015 International Conference on Soft Computing Techniques and Implementations- (ICSCTI) Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015.

[7] Kyongsoo Bok, Hyunkyo Oh, Jongtae Lim and Jaesoo Yoo, “An Efficient Cache Management Scheme for Accessing Small Files in Distributed File Systems”, 2017 IEEE, BigComp 2017.

[8] Yonghau Huo, Zhihao Wang, XiaoXiao Zeng, Yang Yang, Wenjing Li, ZHONG Cheng, “SFS: A Massive small file processing middleware in Hadoop”, IEICE-The 18th Asia-Pacific Network Operations and Management Symposium (APNOMS) 2016.