International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.1 (2014), pp.231-240
http://dx.doi.org/10.14257/ijmue.2014.9.1.22

Multi-level Metadata Management Scheme for Cloud Storage System

Jin San Kong[1] , Min Ja Kim[2], Wan Yeon Lee[3], Chuck Yoo[2] and Young Woong Ko[1]

[1]Dept. of Computer Engineering, Hallym University Chuncheon, Korea
[2]Dept. of Computer Science and Engineering, Korea University, Seoul, Korea
[3]Dept. of Computer Science, Dongduk Womens University, Seoul, Korea

{kongjs, yuko}@hallym.ac.kr, {mjfeel, hxy}@korea.ac.kr, wanlee@dongduk.ac.kr

Abstract

Data deduplication is widely used to reduce storage space requirement in cloud storage system. Especially chunking based deduplication scheme is very useful for handling data storage system that contains duplicated blocks of file stream. In this paper, we introduce a novel data deduplication scheme that manages metadata of cloud storage system in a rapid time. The key points of this paper are using tree map searching and classifying data as global and local metadata. The two-level metadata management scheme is the main aspects to influencing fast performance of the data deduplication.

Keywords: *deduplication, chunking, two-level, metadata, cloud storage*

1. Introduction

Nowadays, there are many cloud storage systems that are widely used for safe file store, convenient file access and file synchronization. For example, Skydrive, Dropbox, Google drive and Ndrive is well-known cloud storage service. Recently, in a cloud storage system, data deduplication is actively used for reducing storage capacity and network bandwidth. In cloud storages, very few vendors only provide data deduplication technology. For example, Dropbox adapts VLC (Variable-Length Chunking) for processing data deduplication, so Dropbox can reduce network bandwidth when data transfers between client and server. One of the key drawbacks of commercial deduplication approach in cloud storage is to use predefined chunk size for all files, so it is difficult to adjust the chunk size of storage system at runtime. In cloud storage system, the file size is varying from Kbytes to Gbytes. If we adapt small size of block for chunking a file then we have to handle large size of metadata for managing data deduplication. The well-known cloud storage system, Dropbox also uses fixed size chunk for data deduplication, so if a file is very big then the metadata size also very increased.

In this paper, we propose two-level metadata management scheme for supporting efficient data deduplication. The proposed system has its own characteristics for providing less storage requirement and fast data deduplication by using two-level metadata management. If the file size is smaller than predefined threshold value, the metadata of a file belongs to local metadata category. Otherwise it will be included in global metadata category. To provide usefulness of two-level metadata management scheme, we show intensive experiment results considering several aspects of cloud storage service.

[1] *yuko@hallym.ac.kr : corresponding author*

The rest of this paper is organized as follows. In Section 2, we describe related works about deduplication system. In Section 3, we explain the design principle of proposed system and implementation details for data deduplication using two-level scheme. In Section 4, we show performance evaluation result of the proposed system and we conclude and discuss future research plan.

2. Related works

In a backup system, a version control program, P2P system and CDN system, data deduplication scheme is widely used for minimizing disk capacity and reduce network traffic. The state of art works related to data deduplication is Rsync [1], LBFS [2], Venti [3] and Multi-mode [4]. Rsync is a software application which synchronizes files and directories from one location to another location while minimizing network traffic using rolling checksum. Rsync uses a reliable algorithm to bring remote files in rapid time by sending the differences in the files over the network. Venti is a network storage system with a 160-bit SHA-1 hash of the data that enforces a write-once policy since no other data block can be found with the same address. The addresses of multiple writes of the same data are identical. So duplicate data is easily identified and the data block is stored only once. LBFS, a network file system designed for low bandwidth networks. LBFS exploits similarities between files or versions of the same file to save bandwidth. It avoids sending data over the network when the same data can already be found in the server's file system or the client's cache. Using this technique, LBFS achieves up to two orders of magnitude reduction in bandwidth utilization on common workloads, compared to traditional network file systems. In multi-mode, they propose a data deduplication system using multi-mode (source-based approach, inline approach and post processing approach). The multi-mode system can be operated in several modes that a user specifies during system operation, therefore, this system can be dynamically adjusted under consideration of system characteristics.

TAPER [5] is a redundancy protocol for replication at the file synchronization level. TAPER sends every chunk hashes to the clients and each open client receives it, check whether chunk existing by Rabin fingerprint. If there is chunk hash value emerged on any client, client sends responding to the TAPER. Finally TAPER collecting responding information from clients sends non-duplicated data to the corresponding client. Mogul [6] researches data deduplication usage in HTTP transferring. In every payload of each HTTP, hashing MD5 and checks the hash value is exist on browser's cache. In this technique, Web cache can use digests to detect and potentially eliminate all redundant payload transfers. Additionally, there is a research result exploiting file modification pattern for enhancing data deduplication performance [7].

3. Two-level Metadata Management Scheme

Figure 1 shows overall system architecture of the proposed system. We adapt VLC approach for block chunking and source-based data deduplication approach. In source-based approach, data deduplication process is performed in the client side and the client sends only non-duplicated files or blocks to deduplication server. The client performs file data deduplication process by sending file hash key to server. The server checks file hash key from file hash index on DBMS. If there is no matching file hash key in the server, the client starts block-level deduplication. The client divides a file into several blocks and calculates hashes of the each block. The list of hash keys is delivered to the server and the server checks duplicated blocks by comparing the hash key with hash keys in the server. The server makes a non-duplicated block list and sends it to the client. Finally, the client sends the non-duplicated data blocks to the server. The server manages metadata by preserving and comparing hashes from the client.

As we can see in Figure 1, the server maintains two types of metadata, one is global metadata and the other is local metadata. Briefly speaking, whenever there exist requests for file synchronization between the client and the server, the client chunks a file into predefined size chunk according to the server information. The server only maintains two types of chunk size for small file and large file.

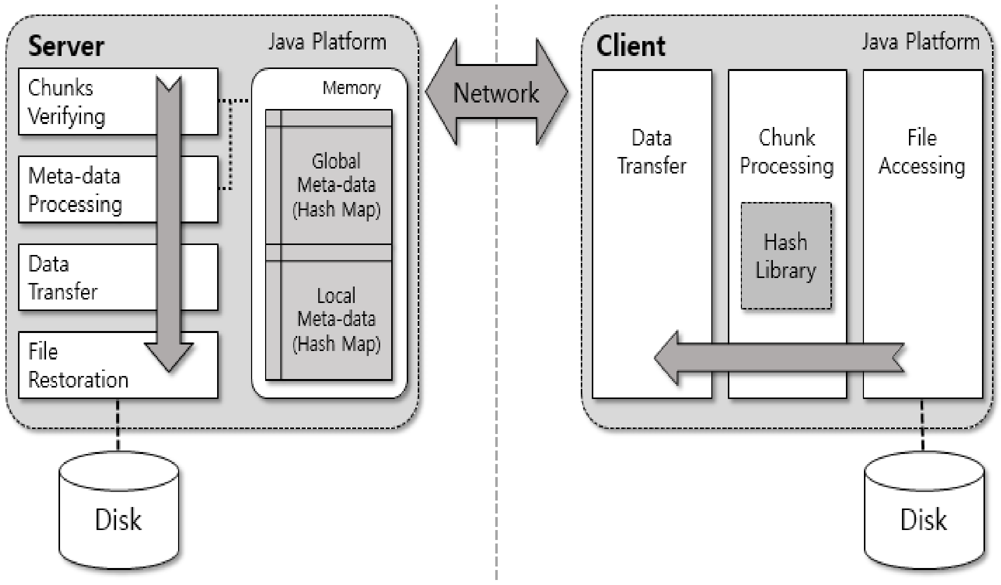

Figure 1. Overall system architecture in two-level metadata management system

The system is divided into several parts; chunk manager, chunker and protocol interface. These modules are used for reducing bandwidth of network and boosting up the performance when backing up in the client. The key contribution for faster synchronization performance than other deduplication system is the metadata separation between user local metadata and global metadata.

Chunk manager uses tree map which is synchronized internally. The hash values are expressed and managed as 20 byte in hexadecimal. Tree map includes this key value as sorted and also includes other data values. Map tree sets its values into as red-black tree, therefore using tree map is very fast compared with other data structure. Server chunk is stored in map tree as sorted by its hash value. Chunk manager has methods which finding overlapping data between client and server metadata chunks (*compareMeta*), loading memory from disk (*startManager*) and saving data to disk from the memory (*stopManager*).

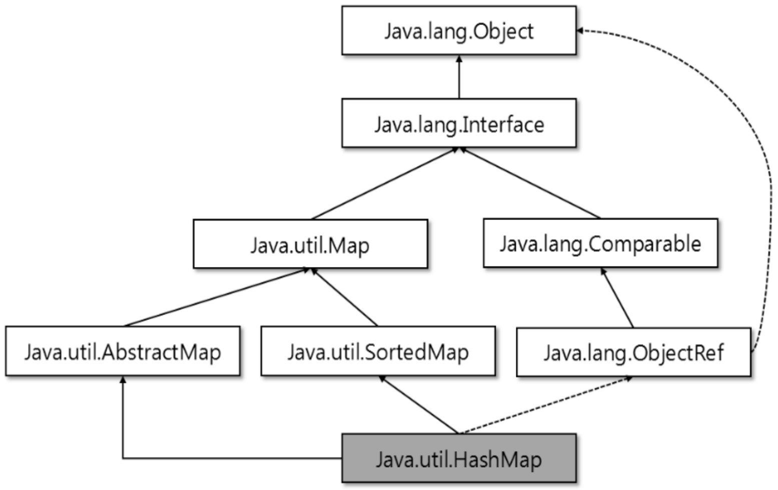

Figure 2. Tree map hierarchy in Chunk manager

Figure 3 explains metadata comparison algorithm in chunk manager for the proposed system. The server maintains one unique metadata without distinguishing each user when the file size is greater than predefined values. This global metadata is accessed by all users on the cloud storage systems; therefore, file synchronization is very important for global metadata.

```
compareMeta(List clientMeta, List nondupMeta, List retrMeta)
  CTree ctree;
  ServerChunk temp ← NULL;
  for i ← 0 to clientMetaSize do
    temp ← ctree.find( clientMeta[i] );
    if temp != NULL then
      temp.setState(true);
      retrMeta(temp);
    else
      ctree.add( clientMeta[i] );
      nondupMeta.add( clientMeta[i] );
      retrMeta.add( clientMeta[i] );
end
```

Figure 3. Metadata comparison algorithms

In the server, when a user logs on a system, each local metadata is loaded on the main memory. Using this metadata, each client performs file operations including file copy, delete, modify and rename. If there is a file with its size is bigger than predefined value, server stores this metadata to public. We call this kind of data to global data and this information has to be synchronized because all users can access this folder. Global chunk manager is responsible for managing this global data. Server loads corresponding local metadata memory when user is connected to the system. The local metadata is the metadata of file which size is lower than predefined value. The user usually manages the local metadata. The module is saving and loading for the metadata, therefore we can perform fast and efficiently update the individual file due to saves the metadata separately for each user.

Figure 4 shows the data deduplication processing flow of the proposed system. The numbers on the figure illustrates flow sequence of deduplication between server and client.

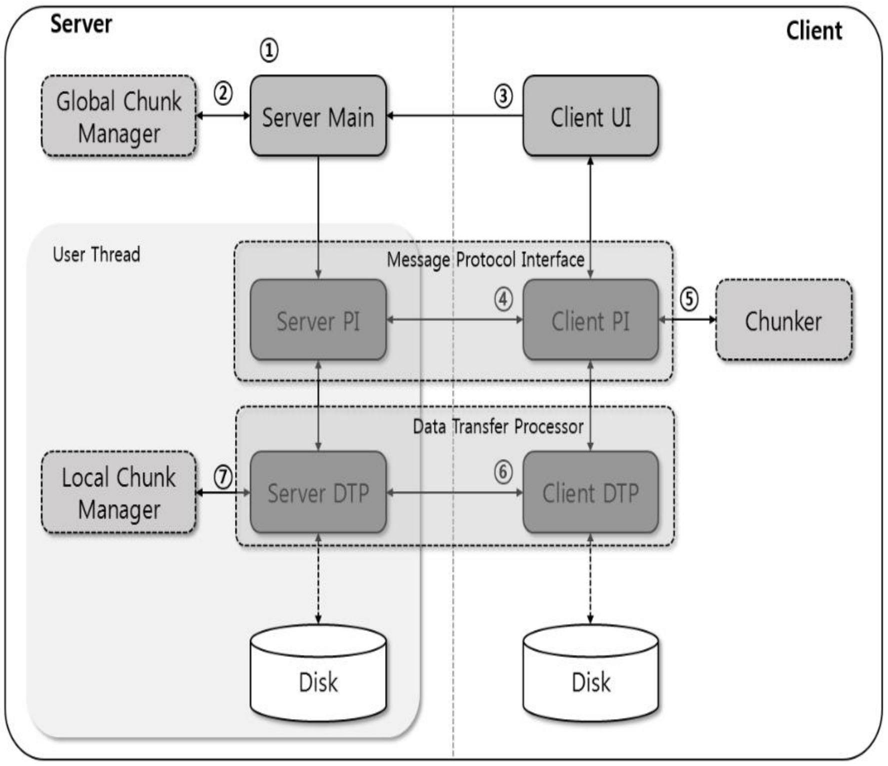

Figure 4. Flow of data deduplication

① Server loads global metadata into memory by global chunk manager when system starts.

② Global metadata can be accessed by all users and it is managed by Global chunk manager.

③ Client accesses server by port number 2121, the login process is completed after user putting User Id and Password.

④ Server and Client are connected by response message and commands through protocol interface.

⑤ Client passes the selected file to the chunker when it sends backup request to the server. The chunker returns metadata (hash value, offset and chunk size) of transmitting file.

⑥ Metadata of a client file is transmitted by Data Transfer Process.

⑦ Transmitted data is deduplicated and backed-up through chunk manager from global chunk manager and Local chunk manager, and also metadata comparison process.

4. Experiment Result

In this work, we evaluate the data deduplication system using two-level metadata management system. As described in table 1, the server and the client platform consist of Pentium 4 Processor with 4GByte RAM, Windows 7 OS.

Table 1. Experiment environment

SERVER		CLIENT	
CPU	Intel Core2 Duo 2.93GHz	CPU	Intel i5-2400 3.1GHz
RAM	4.00GB	RAM	4.00GB
OS	Windows 7	OS	Windows 7
Language	Java	Language	Eclipse, Java

We made experimental data set using for modifying a file in a random manner. In this experiment, we modified a data file using *lseek()* function in Linux system using randomly generated file offset and applied a patch to make test data file. The Input data of experiments are patched 40% and 80%. In this experiment, we used Netlimiter3 program for analyzing network packet usage.

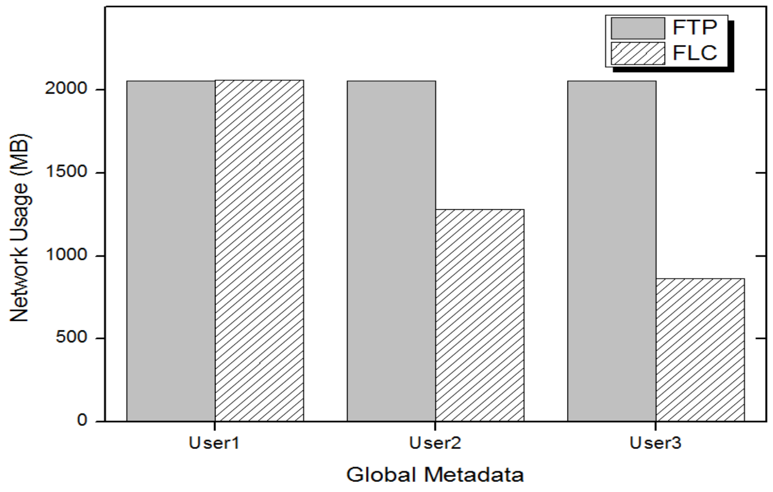

Figure 5. Evaluation result of network bandwidth on global metadata scheme

Figure 5 shows experiment result of network usage when data transfers using global metadata. There are 2GB original file and its copy versions with 40%, 60% duplicated files. User1, User2 and User3 each transfers original, 40% duplicated and 60% duplicated versions of file through the FTP between the client and the server. FTP transfers 2GB file whereas the proposed system consumes less network bandwidth when transfers file using global metadata. Figure 6 shows CPU utilization result of the proposed system. In Figure 6, (A) indicates original data, (B) 40% duplicated data and (C) 80% duplicated data. As can be seen figure, during 10 seconds, CPU utilization is almost 100% for hash calculation and after 10 sec data is transferred to the cloud server. In (A), data transferring time is taken 28 seconds between 10 second to 38 second for original data file. However, (B) and (C) show only 15 seconds and 6 seconds, respectively.

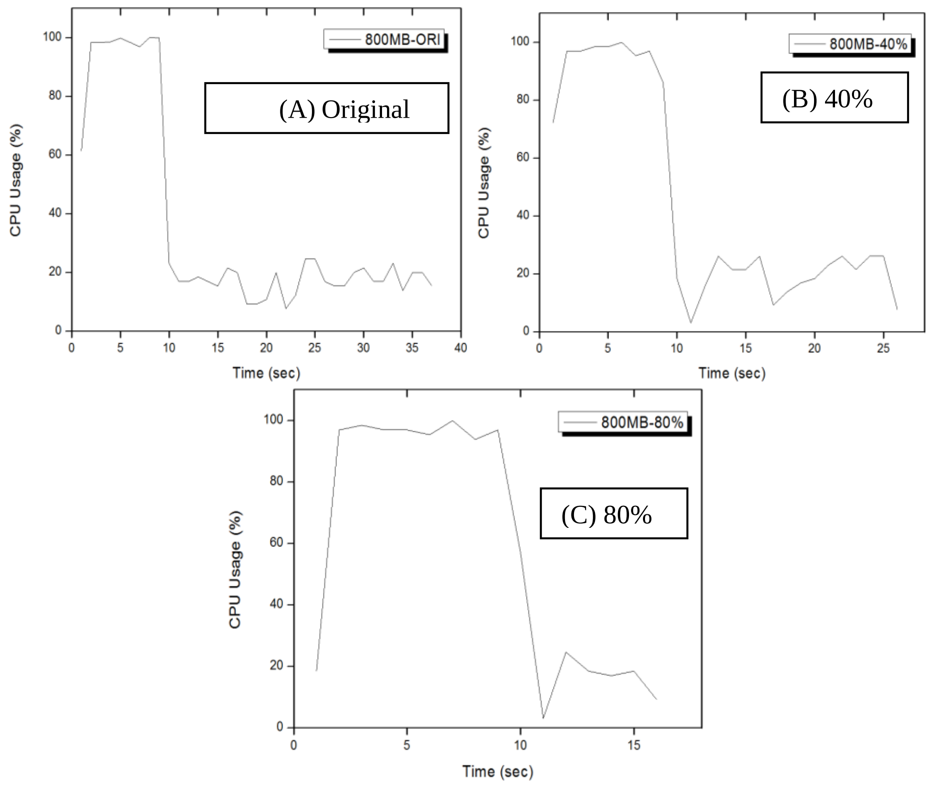

Figure 6. Experiment result of CPU utilization

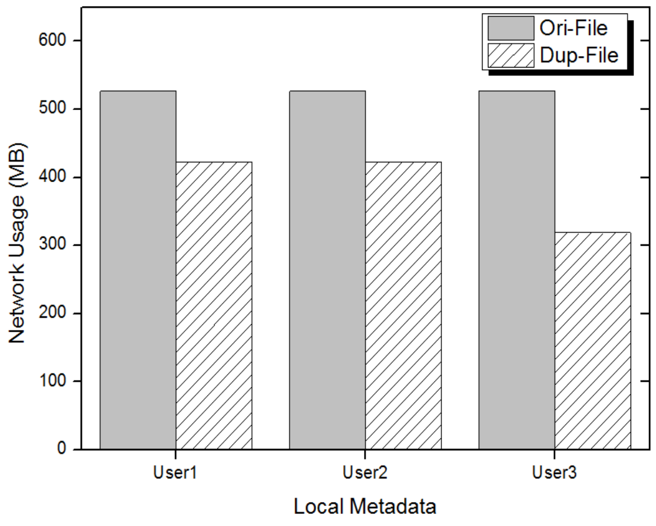

Figure 7. Experiment result of network usage on local metadata scheme

The above graph shows experiment result of network usage when data transfers using Global metadata. There are 500MB original file and its copy versions of 20%, 40% duplicated files. When each user transfers to save 500 MB file to their private folder, network usage is shown in Figure 7. Global-metadata and Local-metadata are the reason to be excellence to save less metadata search time.

5. Conclusion

In this paper, we proposed two-level metadata management for efficient data deduplication. The main idea is to separate metadata management by considering file size. If the file size is smaller than threshold value, the metadata of the file belongs to local metadata. On the contrary, if file size is bigger than threshold value, file metadata belongs to global metadata. Server creates private metadata folder for each user and only owner has a permission to access for corresponding folder. When user performs to run and backup file for local metadata, server loads the metadata from its private folder to process deduplication process. However, when a file is restored or backed-up, file is classified as global, server loads the data from global folder to execute the process. Our approach shows fast and low bandwidth performance compared with FTP approach.

Acknowledgements

This research was supported by Basic Science Research Program through the NRF funded by the MEST(.2012R1A1A2044694) , and this work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2011-0029848).

References

[1] A. Tridgell and P. Mackerras, “The Rsync algorithm”, Tech. Rep. TR-CS-96-05, The Australian National University, **(1996)** June.
[2] A. Muthitacharoen, B. Chen and D. Mazieres, “A low-bandwidth network file system”, ACM SIGOPS Operating Systems Review, vol. 35, no. 5, **(2001)**, pp. 174-187.
[3] S. Quinlan and S. Dorward, “Venti: a new approach to archival storage”, In: Proceedings of the FAST 2002 Conference on File and Storage Technologies, **(2002)**.

[4] H. M. Jung, W. V. Park, W. Y. Lee, J. G. Lee and Y. W. Ko, “Data Deduplication System for Supporting Multi-mode, Intelligent Information and Database Systems”, Lecture Notes in Computer Science, vol. 6591, **(2011)**, pp. 78-87.
[5] N. Jain, M. Dahlin and R. Tewari, “TAPER: tiered approach for eliminating redundancy in replica synchronization”, In Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies, **(2005),** pp. 21-21, Berkeley, CA, USA.
[6] J. C. Mogul, Y. M. Chan and T. Kelly, “Design, implementation, and evaluation of duplicate transfer detection in HTTP”, In Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, **(2004)**, Berkeley, CA, USA.
[7] H. M. Jung, S. Y. Park, J. G. Lee and Y. W. Ko, “Efficient Data Deduplication System Considering File Modification Pattern”, International Journal of Security and Its Applications, **(2012)**.

Authors

Jin San Kong

He graduated from Dept. of computer engineering, Hallym University in 2011. He also graduated Department of Computer Engineering, Hallym University with master’s degree in 2013. He is currently working as Intern in NHN. His research interests include Data deduplication and Cloud system.

Min Ja Kim

She received the B.S. degree in computer engineering from Dongduk Women's University in 2000 and M.S. degree in Computer Science from Korea University in 2002. She is currently pursuing her Ph.D degree in College of Information and Communications, Korea University, Seoul, Korea. Her research interests include Operating System, File system and multimedia streaming.

Wan Yeon Lee

He received the BS, MS, and Ph.D. degrees in computer science and engineering from Pohang University of Science and Technology in 1994, 1996, and 2000, respectively. He is currently a Professor in the Department of Computer Science, Dongduk Women's University, Seoul, Korea. His areas of interest include mobile network, real-time system, multimedia communication, and parallel computing.

Chuck Yoo

Chuck Yoo received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea and the M.S. and Ph.D. in computer science in University of Michigan. From 1990 to 1995, he worked as researcher in Sun Microsystems Lab. He is now a Professor in College of Information and Communications, Korea University, Seoul, Korea. His research interests include Operating System, Virtualization and multimedia streaming.

Young Woong Ko

He received both a M.S. and Ph.D. in computer science from Korea University, Seoul, Korea, in 1999 and 2003, respectively. He is now a professor in Department of Computer engineering, Hallym University, Korea. His research interests include operating system, embedded system and multimedia system.