Data Deduplication System for Supporting Multi-mode

Ho Min Jung*, Won Vien Park, Wan Yeon Lee,
Jeong Gun Lee, and Young Woong Ko

Dept. of Computer Engineering, Hallym University,
Okcheon-dong, Chunchon-si, Gangwon-do, Korea
{chorogyi,wonvien,wanlee,jeonggun.lee,yuko}@hallym.ac.kr
http://os.hallym.ac.kr

Abstract. The implementation approaches of data deduplication system divide into several modes including SBA(source-based approach), ILA(in-line approach) and PPA(post-process approach). Currently, most commercial systems are implemented and operated in an ILA and PPA approach, and some researchers have focused on the SBA approach. As data deduplication systems are widely used, to choose an appropriate mode considering operation environment becomes more and more important than ever. Because the overhead of each mode and resource usage wasn't fully studied, in some operating environments, the deduplication mode can lead to inefficiency and poor performance. In this study, we propose a data deduplication system supporting multi-mode. The proposed system can be operated in a mode that a user specifies during system operation, therefore, this system can be dynamically adjusted under consideration of system characteristics. In this paper, we operate the proposed system with the SBA, ILA and PPA mode, respectively, and we present the measurement results with a comparative analysis of the mode-specific performance and overhead.

Keywords: Deduplication, backup server, multi-mode, storage system.

1 Introduction

As the computer technology is rapid and widespread development, most information becomes digitalize and the amount of data is rapidly increasing. The amount of digital data around the world will be about 2052 Exabyte by 2012. Therefore, efficient data management will be more important and the data deduplication technique is regarded as an enabling technology. The data deduplication is a specialized data compression technique for eliminating redundant data to improve storage utilization. Generally, in the deduplication process, duplicate data is eliminated, leaving only one copy of the data to be stored, along with references

* This research was financially supported by the MEST and NRF through the Human Resource Training Project for Regional Innovation. And also was supported by Basic Science Research Program through the NRF funded by the MEST(2010-0016143).

N.T. Nguyen, C.-G. Kim, and A. Janiak (Eds.): ACIIDS 2011, LNAI 6591, pp. 78–87, 2011.

to the unique copy of data. With a help of data deduplication mechanism, the required storage capacity can be reduced since only the unique data is stored. In practical way, input data is divided into 4KB or more large blocks and given a hash value for each block. If the hash values are same between blocks, we regard that the blocks are identical. Therefore, before the data blocks are saved to storage system, we can eliminate duplicate data blocks in a file or between files. So, the data deduplication technique can be used efficiently on multiple versions of data such as a backup system. Generally, the functionalities of deduplication system are composed of hashing, comparing, input/output data blocks, searching and network transmission. The architecture of data deduplication system varies based on composing mechanism of the functionalities [1]. The deduplication system is classified as a source-based approach (SBA) and a target-based approach (TBA). If deduplication function is performed on client, it is classified into the source-based approach. If the key functionality of deduplication is performed on a deduplication server, it is classified into the target-based approach. The target-based approach is divided into an ILA and a PPA. In an ILA mode, the system processes a data deduplication function immediately when it receives data blocks. On the other hand, in a PPA mode, the system stores data blocks on the temporary storage at first, and it processes the data deduplication work if there are enough time and resource to do deduplication work.

The SBA, ILA and PPA are currently implemented several commodity product and are widely used. However, there is no concrete result that shows pros and cons of each method on different environments. For example, if a mobile user using 3G network tries to transfer a file, the user prefers to minimize network traffic because the user have to pay money for the amount packet data. In this situation, a SBA mode is suitable for the user, because it reduces network bandwidth. In addition, each method can cause a degradation and inefficiency of system resources because the patterns of system resource usage are different. For example, if the client machine is busy for processing CPU intensive job, then it prefers deduplication mode that uses less CPU resource. In this paper, we propose a multi-mode data deduplication system for file server. The proposed system can dynamically change deduplication mode by user. We measured the performance of deduplication system by operating SBA, ILA and PPA, respectively. The rest of this paper is organized as follows. In Section 2, we describe related works about deduplication system. In Section 3, we describe the operation mechanism of each deduplication mode, SBA, ILA and PPA. In Section 4, we explain the design principle of proposed system and implementation details for the deduplication system. In Section 5, we show performance evaluation result for each mode and we conclude and discuss future research plan.

2 Related Work

Venti [2] is a block-level network storage system which is similar to the proposed system. Venti identifies data blocks by a hash of their contents, because of using a collision-resistant hash function (SHA1) with a sufficiently large output, the

data block can be used as the address for read and write operations. The Low-Bandwidth File System [3] makes use of Rabin fingerprinting [4] to identify common blocks that are stored by a file system client and server, to reduce the amount of data that must be transferred over a low bandwidth link between the two when the client fetches or updates a file.

Zhu et al.s work [5] is among the earliest research in the inline storage deduplication area. They present two techniques that aim to reduce lookups on the disk-based chunk index. First, a bloom filter [6] is used to track the chunks seem by the system so that disk lookups are not made for non-existing chunks. Second, upon a chunk lookup miss in RAM, portions of the disk-based chunk index are prefetched to RAM. Lillibridge et al. [7] use the technique of sparse indexing to reduce the in-memory index size for chunks in the system at the cost of sacrificing deduplication quality. The system chunks the data into multiple megabyte segments, which are then lightly sampled (at random based on the chunk SHA-1 hash matching a pattern), and the samples are used to find a few segments seen in the recent past that share many chunks. Obtaining good deduplication quality depends on the chunk locality property of the dataset whether duplicate chunks tend to appear again together with the same chunks.

DEDE [8] is a decentralized deduplication system designed for SAN clustered file systems that supports a virtualization environment via a shared storage substrate. Each host maintains a write-log that contains the hashes of the blocks it has written. Periodically, each host queries and updates a shared index for the hashes in its own write-log to identify and reclaim storage for duplicate blocks. Unlike inline deduplication systems, the deduplication process is done out-of-band so as to minimize its impact on file system performance. HYDRAstor [9] discusses architecture and implementation of a commercial secondary storage system, which is content addressable and implements a global data deduplication policy. Recently, a new file system, called HydraFS [10], has been designed for HYDRAstor. In order to reduce the disk accesses, HYDRAstor uses bloom filter in RAM.

3 Multi-mode Deduplication

In this section, we describe SBA, ILA and PPA that are used on deduplication system and present the feature of each mode.

3.1 SBA Mode

In SBA mode, data deduplication process is performed in the client side and the client sends only non-duplicated files or blocks to deduplication server. First, the client performs file deduplication process by sending file hash data to server. The server checks file hash data from file hash list on DBMS. Second, if there is no matching file hash data in the server, the client starts block hashing.

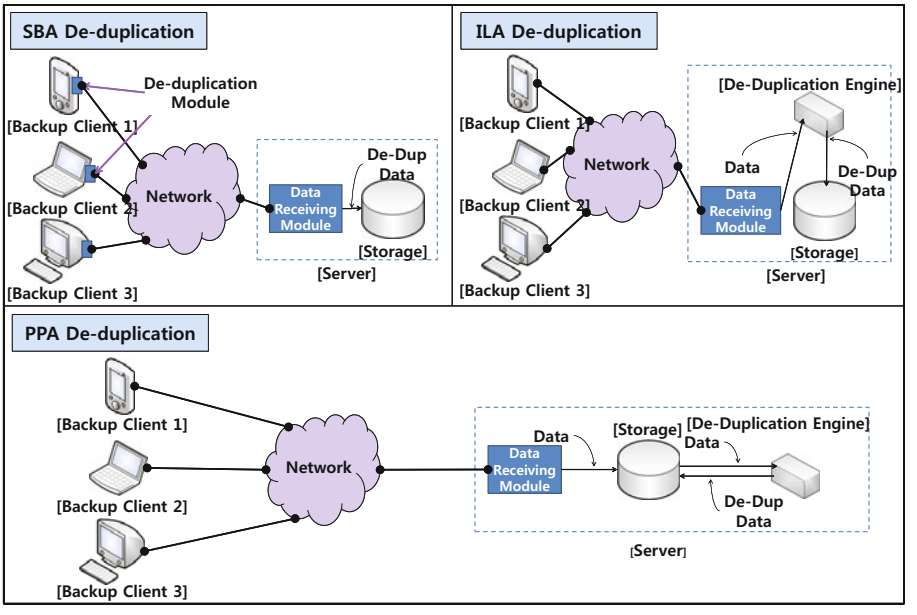


Fig. 1. The Conceptual Diagram of Deduplication Mode

The client divides a file into several blocks and hashes each block. The list of hash data is delivered to the server. Third, the server checks duplicated blocks by comparing the delivered hash data to hash data in the server. The server makes non-duplicated block list and send to the client. Finally, the client tries to send the non-duplicated data blocks to the server. In this mode, we can save network bandwidth by reducing the number of data blocks, therefore it is useful for a mobile device using 3G network. However, this approach requires more intensive CPU loads and the client may consume extensive system resource.

3.2 ILA Mode

ILA method performs elimination of duplicated data on server side when data is transferred from client to server. The client sends backup data to server and the server process deduplication work on the fly. Therefore, the client should send enough data blocks for handled in server side. The ILA mode can reduce disk storage of the system because the system does not need to preserve additional storage system for temporal data storing. In addition, this mode have decreased the entire working time by the immediate deduplication work and can save the DR(Disaster Recovery)-Ready time. However, it has drawback for managing the server not efficiently when large amounts of clients access to the server and the CPU resource of the server is all consumed.

3.3 PPA Mode

This mode performs deduplication work after data is temporary written to the disk in a server. The server saves the data from the client at first, then it send all data to deduplication engine for doing deduplication processing. After eliminating the duplication data, the server saves only non-duplicated data blocks to storage. The PPA usually consumes the system resource of a server while reducing the system resource of the client because all the deduplication work is processed on the server side. However, it needs additional disk storage for saving data on a disk. Moreover, DR-Ready time also increase because there are the gap between backup time and data deduplication.

4 Architecture of the Proposed System

To provide the context for presenting our methods for supporting the multi-mode and file deduplication, this section describes the architecture of deduplication system and explains the internal of each mode.

In figure 2, we show the main modules of the proposed system. The deduplication system are composed of several modules: File Accessing, Data Deduplication, Data Transfer, File Deduplication, Data Receiving, Data Scheduling, Data Deduplication, Compressing and DBMS.

File accessing module performs file grouping for efficient file deduplication by considering disk buffer cache size. If there are too many files to deduplication, the buffer cache will be busy for handling disk I/O. Therefore, we only treat enough files by grouping it within the size of disk buffer. With this approach we can avoid heavy disk I/O traffic. In file deduplication module, duplicated files are eliminated by using file hash. In client, it creates file hash for the file group and passes the file hash list to file deduplication module on the server. In

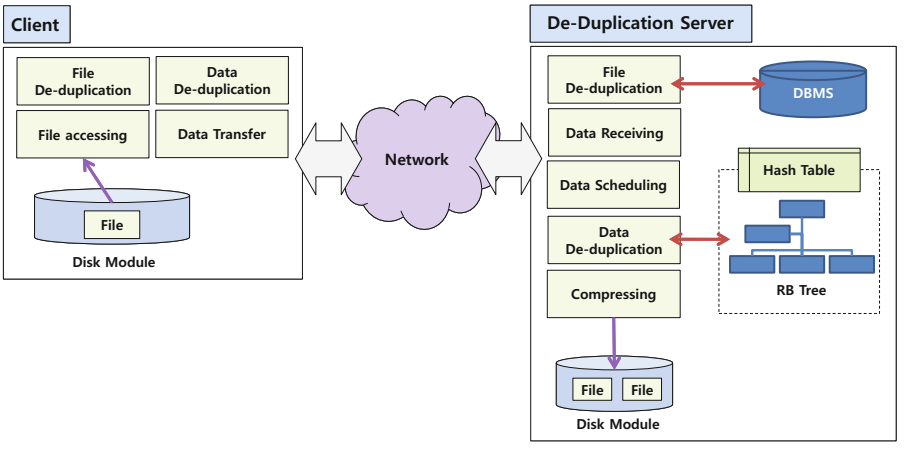


Fig. 2. The Architecture of the Proposed System

server, it checks duplicated files by comparing existing file hashes on a DBMS. The server sends only duplicated file hash list to the client. With this approach, we can prevent duplicated files are transferring to the server. In data deduplication module, block-level data deduplication is processed. The system divides data stream into blocks with chunking function. And then, we can get each data block hash using hash function such as SHA1, MD5 and SHA256. Generally, a chunking function can be dived into a fixed length chunking method and variable length chunking method. In our work, we adapted a fixed length chunking method because it is much more simple and easy to implement. The chunking size of data block is varying from 4Kbyte to several Mega Byte. In our work, we fixed 4Kbyte chunking size for increasing the performance of data deduplication. By choosing small chunking block, we can increase the possibility of finding

Algorithm 1. Multi Mode Algorithm

```
begin
    modeselection ← receivemode();
    filededuplication();
    if modeselection = SBA then
        while eof ≠ null do
            hash ← receivehash();
            check ← checkhash(hash);
            sendcheck(check);
            if check ≠ false then
                continue;
            else
                receivedata();
            end
            savemeta();
        end
    end
    if modeselection = ILA then
        while eof ≠ (data data ← receivedata() ) do
            hash ← makehash(data);
            check ← checkhash(hash);
            if check ≠ false then
                continue;
            else
                savedata();
            end
            savemeta();
        end
    end
    if modeselection = PPA then
        data ← receivedata();
        meta ← todisk(data);
        dataschedule(meta);
    end
end
```

duplicated block. Hash retrieval also very important because it causes frequent comparison, insert and delete operation. So, we adapted a red-black tree data structure for high performance hash operation. To process file and block deduplication, all the metadata have to be efficiently managed in a database module. The metadata includes file and block information, file and block hash data, file and block location, etc. Moreover, each file is composed of several blocks with and without deduplicated blocks. To build file from blocks, we have to carefully manage each block index.

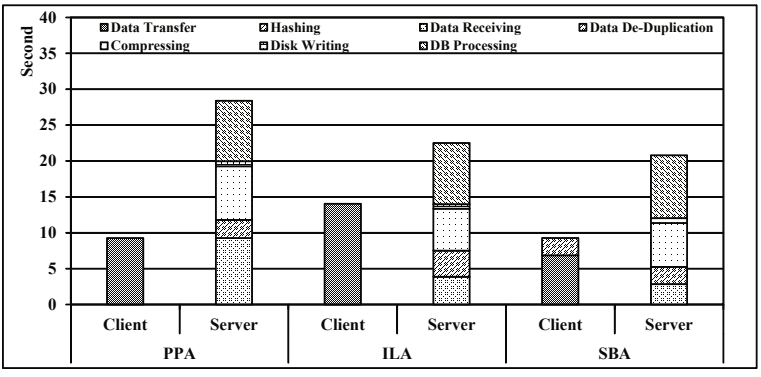
4.1 Multi-mode Algorithm for Deduplication

Algorithm 1 shows how to support multi-mode deduplication. First, the server receives mode information from the client. Second, the server performs file deduplication work, which is a common work regardless of each mode. Third, if the mode is set to SBA, then the client chunks the data and gets the block set and creates hash for each block before sending data blocks to the server. After verifying that the hash file is in the server, if server has the hash file then the server skips the data transferring step. Otherwise the server receives the data from the client and saves it in the disk. Fourth, if the mode is set to ILA, the server receives data from client and makes a data hash file by splitting data blocks. Similar to the SBA mode, it verifies if the hash file is in the server. If the server doesn't have the same hash value, the server saves the hash and metadata to the disk. The server repeats this routine until it receives EOF control message from the client. Finally, if the mode is set to PPA, the server receives file metadata from the client until all the data blocks are transferred and it saves the blocks temporarily into the disk. The server eliminates duplicated files and blocks when it scheduled to do deduplication work. Algorithm 1 shows the overall steps of the proposed system.

5 Performance Evaluation

In this work, we implemented the proposed system on Fedora core 9 operating systems. The hardware platform is equipped with Pentium 4 3.0 GHz CPU, 1024MB RAM, and WD-1600JS(7200/8MB) hard disk. All experiments are performed in the computer with Fedora Core 9. The Client and deduplication server is under the same hardware. In our experiment, data stream is composed of data blocks which duplicated rate varying 0, 40, and 80 percentages with 100 MByte size. 0 percentages means that there is no duplication in data blocks and 100 percentages mean that all of the data are duplicated. We measured the execution time, CPU resource and network bandwidth on the client and the server. All experiment is performed 10 times with SBA, ILA and PPA mode.

Figure 3 shows the execution time measurement for each mode. For the client, PPA is the fast because the client only transfers data to the server without deduplication work. And the execution time is proportional to the size of data. ILA shows much more delayed for processing data stream because ILA must



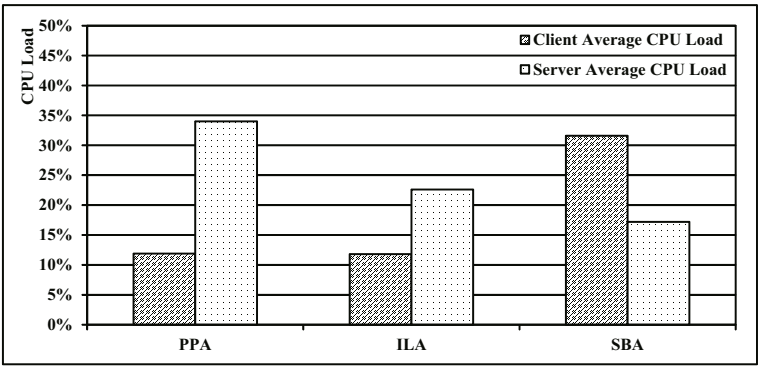
unit : ms

Duplication Ratio	De-dup Scheme	Send Data	Hashing	Recv File	Data Dedup	Compress	Disk Write	DB Process
0%	PPA	9172	0	9162	2601	10242	967	11029
	ILA	16447	0	1395	3815	10147	1006	11062
	SBA	10804	2346	2817	2339	9564	1105	11175
40%	PPA	9262	0	9251	2541	7469	708	8427
	ILA	14044	0	3872	3640	5848	607	8538
	SBA	6861	2411	2832	2413	6109	681	8759
80%	PPA	9077	0	9055	2431	4729	426	6464
	ILA	10688	0	5114	3144	1992	324	6756
	SBA	2432	2395	2616	2395	2136	343	6383

Fig. 3. Execution Time Analysis for Each Mode

wait until the server finishes deduplication work. We can minimize waiting time by increasing buffer size in the server. However, spacious buffer size can result in system performance degrade because physical memory is critical resource handling large hash index data. The execution time of SBA mode is depending on duplication rate. If duplication rate is high, then overall execution time will be decrease because the data transfer time is decreased. For the server, SBA mode needs small portion of system resource and PPA mode needs much more system resource. The system with SBA mode process all the hash work in the client side, therefore the server only store non-duplicated data blocks. However PPA mode has to do all the deduplication work in the server side, which makes the system very busy.

In figure 4, top graph show the average CPU load when duplication rate is 40 percentage and bottom table shows that average CPU load when duplication rate is 0, 40, 80 percentage, respectively. CPU load of the PPA and ILA show almost same value, 11 percentages, regardless of duplication rate. While SBA mode shows 31 percentage CPU loads. The reason why CPU load of SBA is higher than other mode is that file chunking and hashing work is done in the client mode. For the server, the mode that require higher CPU load is PPA because PPA mode requires deduplication work and additional disk I/O for processing temporal data storing. With this experiment, we conclude that if a user want to fast file deduplication in the client side without delay, then PPA is preferable. However, if a user wants to minimize CPU load for both the client and server,



Duplication Ratio	De-dup Scheme	Client Time (ms)	Client CPU Avg(%)	Server Time (ms)	Server CPU Avg(%)
0%	PPA	9172	11.8	34001	40
	ILA	16447	11.9	27425	32
	SBA	13150	31.6	27000	20.2
40%	PPA	9262	11.9	28396	34
	ILA	14044	11.8	22505	22.6
	SBA	9272	31.6	20794	17.2
80%	PPA	9077	11.9	23105	28.7
	ILA	10688	13.1	17330	20
	SBA	4727	30.9	13873	9.3

Fig. 4. Comparison of Average CPU Load

then ILA mode is a good solution. And if a user want to fast file transfer and low CPU load for the server, then SBA is the best choice.

6 Conclusion

In this paper, we presented data deduplication system supporting multi-mode which can dynamically change deduplication mode. Our key idea is to provide suitable deduplication mode considering system resource and user preference. We have designed and implemented the multi-mode deduplication server on the Linux system. We analyzed the performance of SBA, ILA and PPA modes, Experiment results show that PPA can minimize execution time for handling data deduplication and CPU load for the client. Our result shows that if a user wants to fast processing for file deduplication with low CPU workload in the client side, then PPA is preferable. However, if we want to minimize the system overhead on the server side, we have to use SBA mode. For future work, we will study how to convert the optimized method to dynamically considering CPU, I/O and network bandwidth. Moreover, we will design the optimized deduplication module providing QoS(quality of service) concept by continuously monitoring server and client behavior. Also, we believe that energy efficient data deduplication is useful for smartphone environment in the future.

References

1. Tan, Y., Jiang, H., Feng, D., Tian, L., Yan, Z., Zhou, G.: SAM: A Semantic-Aware Multi-tiered Source De-duplication Framework for Cloud Backup. In: 39th International Conference on Parallel Processing (2010)
2. Quinlan, S., Dorward, S.: Venti: a new approach to archival storage. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST (2002)
3. Muthitacharoen, A., Chen, B., Mazieres, D.: A Low-Bandwidth Network File System. In: Proceedings of the Symposium on Operating Systems Principles (SOSP 2001) (2001)
4. Rabin, M.O.: Fingerprinting by random polynomials:Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University (1981)
5. Zhu, B., Li, K., Patterson, H.: Avoiding the disk bottleneck in the data domain deduplication file system. In: Proceedings of the 6th USENIX Conference on File and Storage Technologies, FAST (2008)
6. Broder, A., Mitzenmacher, M.: Network Applications of Bloom Filters: A Survey. In: Internet Mathematics (2002)
7. Lillibridge, M., Eshghi, K., Bhagwat, D., Deolalikar, V., Trezise, G., Campbell, P.: Sparse Indexing, Large Scale, Inline Deduplication Using Sampling and Locality. In: Proceedings of the 7th USENIX Conference on File and Storage Technologies, FAST (2009)
8. Clements, A., Ahmad, I., Vilayannur, M., Li, J.: Decentralized Deduplication in SAN Cluster File Systems. In: Proceedings of 2009 USENIX Technical Conference (2009)
9. Dubnicki, C., Gryz, L., Heldt, L., Kaczmarczyk, M., Kilian, W., Strzelczak, P., Szczepkowski, J., Ungureanu, C., Welnicki, M.: HYDRAstor: a Scalable Secondary Storage. In: Proceedings of the 7th USENIX Conference on File and Storage Technologies, FAST (2009)
10. Ungureanu, C., Atkin, B., Aranya, A., Salil Gokhale, S.R., Calkowski, G., Dubnicki, C., Bohra, A.: HydraFS: a High-Throughput File System for the HYDRAstor Content-Addressable Storage System. In: Proceedings of the 8th USENIX Conference on File and Storage Technologies, FAST (2010)