2012 International Conference on Image, Vision and Computing (ICIVC 2012)
IPCSIT vol. 50 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V50.20

A Deduplication-based Data Archiving System

Ma Jianting[a,*]

[a] College of Computer Science, Sichuan University, Chengdu, 610064, China

Abstract. For the vast amount of duplicate data in the archiving server, a deduplication-based archiving system is proposed. At the time of data archiving, a snapshot for volume is generated and data are read directly from the snapshot. Then the data processed by data deduplication technologies are sent to archiving server for storing. After successful archiving, the original data on the client will be deleted. Through this method, storage space of server is economized, so does network bandwidth.

Keywords: SNAPSHOT, DATA DEDUPLICATION, INDEX, ADDRESS

1. Introduction

With the more importance of data for enterprises, a variety of backup systems emerged. They provide a good solution for preventing data loss. But with the explosive growth of data count, the demand for data archiving is significantly increased. Data backup is that data replication is to ensure that data can be copied back in the event of data loss or system disaster occurred. Its focus is the change and update of business information which will be stored transitorily and modified frequently. Data archiving is planned data migration. When the data are not frequently changed and used, they will be migrated to another place. Through data archiving, the storage space frequently used is economized.

Paper [1] proposed that large amounts of data in the file system are duplicate or similar, and average growth rate annual reaches 68%. If this duplicate data are also stored in the archive server, they will lead to great waste on server storage space. To economize storage space, how to eliminate duplicate data becomes hot research.

The data deduplication technologies are adopted in our achieving system. A snapshot for volume is generated at the time of data achieving. Then data will be read from snapshot and processed by data deduplication technologies. The processed data are sent to server and stored. At last, the effectiveness of economizing storage space for archiving server is realized.

2. System design

The data archiving system proposed by the paper is consisted of two components: client and archive server. The client is a computer which needs data archiving; Archive server is data archive storage container. System architecture is shown in Fig.1:

* Corresponding author.
E-mail address: jianting.ma@gmail.com.

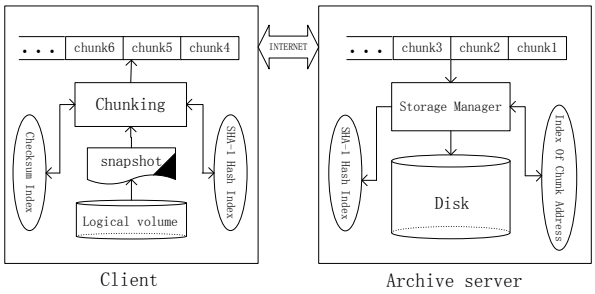

Fig.1. system architecture

2.1. Snapshot

SNIA(Storage Network Industry Association) has the definition of snapshot: A fully usable copy of a defined collection of data that contains an image of the data as it appeared at the point in time at which the copy was initiated. A snapshot may be either a duplicate or a replicate of the data it represents.

In our system, the snapshot is used at the time of needing data archive for logical volume. A replica of volume will generate through the snapshot. When reading data, the data will read directly from snapshot not from the logical volume. Through the method, the system can provide uninterrupted services for other task and ensure business continuity. The reason for using the snapshot is that the data which existed before the archiving time need reading [2]. If data are read directly from the volume without using volume snapshot, some changed data after achieving time will be read and eventually lead to data inconsistencies. Although the data variability is very small and data almost not change when the data need to archive, the data inconsistencies will occur once some data change. There are several kinds of snapshots for the volume. Here we adopt Windows Volume Shadow Copy Service, which is a snapshot of pointer remap. The implementation of Pointer-based snapshots is as follows:

- First, if the data which need to read is not modified in the period between the time point of snapshot being created and the time point of reading this data, this data will be read directly from the original volume.
- Second, if the data which need to read are modified after the time point of snapshot being created, a new and same size region on disk is created to store the previous data not modified. After the successful storing data, the modify operation on volume will be essentially executed. Then data will be read from the new created region. The advantage of this snapshot is less operation on original data and less storage needs. It just meets the character of little changes for data archiving.
- Third, through the operation above, the reading data is considered as reading unified from snapshot.

2.2. The data chunking

Currently, data deduplication technologies are applied on the file level. The file is divided into some non-overlapping and different chunks. These chunks are unified together and used to determine whether redundancy, such as described in paper [3-8]. The papers [3,4] used Content-Defined Chunking algorithm (CDC) to divide files. The method is that selecting the split point on the file through Rabin fingerprints and dividing the file into some chunks of varying length. Paper [5] proposed another chunking algorithm based on content, named fingerdiff. It is an improvement of CDC. The fingerdiff algorithm merges the chunks which not change. Storage overhead will be reduced through the lower of the chunks amount. Paper [6] also proposed Two Thresholds Two Divisors (TTTD Algorithm) Algorithm to segment the files as the chunks.

Our system apply the data deduplicaiton technologies in the lower file system, the date is read directly through the logical volume or the snapshot ignoring file types. These data are considered as binary data. Here, we use the Sliding Blocking Method to chunking [1]. As follows:

- First, a snapshot for the volume of needing data archiving is generated. Then a data stream is formed by reading data from the snapshot or volume.
- Second, the rsync checksum and a block-sized sliding window are used to calculate the checksum of every over-lapping block-sized segment of the stream.
- Third, the checksum for each block-sized segment is compared to previously stored values in the checksum index. If a match is found, the more expensive SHA-1 hash of the block is calculated and compared to stored hash values in the SHA-1 hash index to detect duplicates. Because the SHA-1 hash will not be changed until the segment data changed .A SHA-1 hash can represent a segment data [9].
- Fourth, If a duplicate is found in the SHA-1 hash index, it is recorded as T0 (Fig.2) and the sliding

window is moved past the duplicate block to continue the process. Additionally, the fragment of the object between the end of the previous block and the newly detected duplicate must be recorded as T1 and stored. T0 is constituted only by the Chunk Header, which is < Type, Address, SHA-1 Hash, MD5 Hash>. Type is an integer value used to distinguish between T0 and T1. If the type equal 0, T0 is represented. Else if the type equal 1, T1 is represented; Address indicates that the chunk's address in the original sector; MD5 Hash is the Checksum of the rest of the structure except itself. T2 is constituted by the <Chunk Header, Chunk Data>. Chunk Header in T1 is more one field named Chunk Length than in T0, which represents the length of Chunk Data.

- Fifth, when a match for the checksum or the hash value is not found, the sliding window is advanced and the process continues. If the sliding window moves past a full block-sized segment without matching to a known block. Then the block is packaged as T1 and the check-sum, SHA-1 hash of that block are computed and stored in their respective index for comparison against the values for future blocks.

Through several steps as above, a new data stream (Fig.2) is formed which will be continuously sent to archive server. The stream is constituted of two types: T0 and T1.for a lot of T0 in the stream and no chunk data in T0, a lot of duplicates are eliminated.

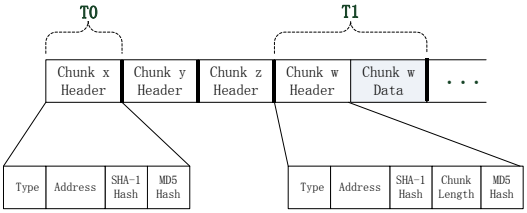

Fig.2. a new data stream

The Sliding Blocking Method combines with the advantages of fixed-sized partitioning (FSP) algorithm and the advantages of Content-Defined Chunking algorithm. The method has the advantages of easy to manage chunks and high efficiency to insert or delete data [1].The sector is the minimum data unit of disk and the sector size is fixed. Most of the chunks generated by the method are fixed size. So, the address of chunks is easy to locate. The chunks size is defined as multiple of sector size. In summary, for logical volume backup, the sliding blocking method is more reasonable.

2.3. Index

2.3.1. Checksum index

The index is deployed in the client and applied to chunking. As along as the shift of the sliding window, the checksum is computed, the computed speed of rsync checksum is faster than SHA-1 hash. The checksum is looked up in the index. If a match is not found, there is no necessary to compute the SHA-1 hash. Through the method, the dividing is speeded up. The checksum index is constructed by the B- tree.

2.3.2. SHA-1 hash index

In some existed system, to save the storage space, the hash index is only deployed in the server, and not deployed in the client. When the SHA-1 hash of chunk is calculated by the client, the SHA-1 hash is sent to server. Then server looks for the hash in the SHA-1 hash index. If a match is found, the duplicate is found. When the match for hash is not found, the server will notify the client to send entire chunk, not only SHA-1 hash. Such as in the paper [3,4].

Although this method can save the client's storage space, but it will add burden to server for all works of looking up in the index are accomplished by server itself. The system proposed by this paper deploys two same index named index 1 and index 2 on client and server separately. Because the looking up hash in index can processed in the client, the burden of sever is lightened. The speed of looking up and comparing is speed up for no transmission on the network. What is more, the storage space for index on the client is allocated temporary and then will be reclaimed. Because when all the data which required achieving is successfully stored on server, the original data on the client will be deleted. So there is no necessary to store the index which generated during chunking period and the index will be deleted eventually. The method has the

advantage of speeding up chunking, easing calculating burden of server and economizing the storage space of client.

For looking up and adding SHA-1 hash frequently in the index, a high efficient storage structure is required. The system adopts B+ tree structure to store index proposed in paper [6]. All non-leaf node of B+ tree only play the role of index and there is no pointer to chunk in non-leaf node. Only the leaf nodes contain the pointer to chunk. So B+ tree saves storage space of index. B+ tree is a balanced binary tree and the path length from each leaf node to the root is the same. It turns adding or looking up hash in the index to easy. The time of looking up will be reduced to O (log n). The index located server is upgraded after the new data stream arrived. The SHA-1 hash in T0 or T1 will be inserted into the index. The tree enable leaf node to point to its' only relational chunk.

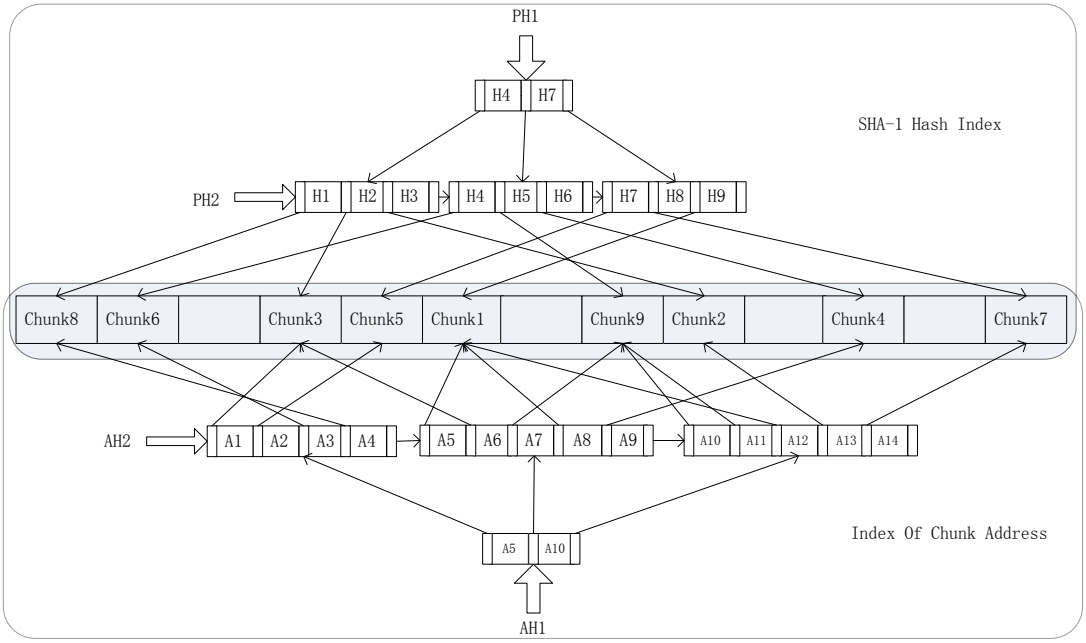

Fig.3. a SHA-1 Hash Index and Index of Chunk Address

2.3.3. Index of chunk address

The data exist in the spread form in the archive server. When the data needs recovery, the data needs to re-organize and sent back to client. A file index is proposed in paper [3,4]. There is a many-to-many relationship between the files and chunks. That is, a file can be composed of multiple chunks and a chunk can be a part of multiple files. When restoring data to client, file is as an organizational unit and all chunks from a file need to re-organize a file. Then it is sent to client. Eventually a file is restored in the client.

This paper proposes a new method to restore volume data through the index of chunk address. The index is deployed in the server and constructed by the B+ tree. The key of index is address which is from chunk header and represents the address of chunks in client. The leaf node of the index point the chunk stored in server. In the index, there is many-to-one relationship between address and chunks. That is, many addresses can share the same chunk. As shown in Fig.3, A5, A8, A12 share chunk 1.there are two head pointer named AH1 and AH2. AH1 points the root of the tree and AH2 points the node which has the least key. All leaf nodes are lined by the key and the line form an orderly sequence. Once restoring data, the data will be read by the sequence of line from AH2. Because the chunk is sent to client by the address of chunk exists in client, the client write directly chunk data into volume once receiving data from server. Through this method, it eliminates seeking disk track time and speed up recovering.

3. Archiving and restore

When there is a requirement to archiving data on the volume, as follows:

- First, a snapshot for volume is generated. Data is read directly from the snapshot to form a data stream.
- Second, the original data stream is transformed for a new data stream which is constituted of T0 or

T1.Then the new data stream is sent to server and the indexes located client need to upgrade.

- Third, the server picks out T0 or T1 from the received data stream. If T0 is picked out, the index on server is upgraded by information of T0. No chunk will be stored for the chunk dedicated by T0 being duplicated.
- Fourth, if T1 is picked out, at the time of index upgrading, the chunk dedicated by T1 will be stored.
- Fifth, repeating the steps from 2 to 4 until all data is sent to and stored on the server. At last, this data on client is deleted completely, so do indexes on client.

Lots of companies almost never view the archive data for many years. Although the purpose of archiving system is used to archive not to restore, but a restore method must be provided in case there is a requirement to view archive data. The restoring process is much easier than the archiving processing. Once restoring data, the data will be read by the sequence of line from AH2 and sent to client.

4. Conclusion

This paper designed a deduplication-based data archiving system. Duplicate will be not archived through the data deduplication technologies. The duplicate in the server will be reduced sharply and the storage space is economized. The requirement of network bandwidth is reduced for lower transferring data. According to the restoring, the paper proposed a new method based on the index of chunk address. The data will be returned to client by the sequence of data address in the client. It eliminates seeking disk track time and speed up recovering.

5. References

[1] Timothy E. Denehy, Windsor W. Hsu. Duplicate management for reference data. *IBM Research Report*, RJ 10305 (A0310-017), IBM Research Division, 2003.

[2] PU Xingyu，LU Zhengtian. Design and implementation of a timing network backup system. *Journal of Sichuan University* .2010, 47(2)

[3] Tianming Yang, Dan Feng. FBBM: A new Backup Method with Data De-duplication Capability. *International Conference on Multimedia and Ubiquitous Engineering*, 2008

[4] Tianming Yang, Dan Feng. 3DNBS: A Data De-duplication Disk-based Network Backup System. *IEEE International Conference on Networking, Architecture, and Storage*, 2009.

[5] Bobbarjung DR, Jagannathan S, Dubnicki C. Improving duplicate elimination in storage systems. *ACM Trans. on Storage*, 2006, 2(4):424−448.

[6] Tin Thein Thwel , Ni LarThein. An Efficient Indexing Mechanism for Data Deduplication. *International Conference on Current Trends in Information Technology*, 2009

[7] Youjip Won, Jongmyeong Ban. Efficient index lookup for De-duplication backup system. *IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, 2008.

[8] Youjip Won, Rakie Kim. Eliminating Information Redundancy for Large Scale Data Backup System. *International Conference on Computational Sciences and Its Applications*, 2008.

[9] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. *In Proceedings of the Conference on File and Storage Technologies. USENIX Association*, 2008.