Evolution of Data De-duplication Strategies and Its Recent Developments

Seemant kumar Sharma, Saurabh Sharma
Gyan Ganga College of Technology, Jabalpur (M.P.)

Abstract*—This Digital world is adapting new digital gadgets, new technologies, Converting every information from legacy for- mat to digital format. Data is available in digital form everywhere. In order to store this large scale of new data producing day by day, the storage methodology should be efficient as well as intelligent enough to find the redundant data to save. Data de-duplication refers to the methods that shrink the need of storage capacity to store data or the quantity of data that has to be transferred over a network. These methods detect coarse-grained redundancies within the data chunk and eliminate them. Some most significant applications of data de-duplication are backup storage systems where these methods considerably reduce the storage requirements to a small fraction of the logical backup data size. This paper reviews methods of data de-duplication, evolution of data de-duplication strategies and their recent advancements.

Keywords - Data De-duplication, Digital Data, Logical Backup.

I. INTRODUCTION

This digital world is generating large amount of digital data and this growth is increasing rapidly. Conferring to a study, the information producing per year to the digital universe will rise more than six times from 161 Exabyte to 988 Exabyte between 2006 and 2010, growing by 57% annually. This enormous growth of information is imparting a considerable burden on storage systems. The terror attacks of the 9/11 events and the data loss of enterprises in those attacks evidenced that data loss is devastating to a modern enterprise which will further lead to shutdown of that enterprise. So it is critical to back up the data regularly to a DR (disaster recovery) site for data availability and integrity [1]. Enterprise Data consists of pictures, audio, video, email conversations, scanned documents etc. Every organization achieves this data for business and legal issues. Rapidly increasing data arises many challenges to the existing storage systems. The large amount of data requires more storage medium to be used. As the data increases, more data is for backup [2]. The cost of the storage media has decreased, but the main problem is to manage number of disks in the back-up systems.

In fact, In enterprise archives a large amount of data is redundant with a slight change to another chunk of data. There are many techniques been developed for removing redundant copy from the stored data. Now days, data de-duplication has gaining popularity in research community. Data de-duplication is a specialized data compression technique for removing redundant data, typically to improve storage utilization. In the de-duplication process, redundant data is left and not stored, leaving single copy of the data chunk to be stored, and a pointer to the unique copy of data [3]. De-duplication is a

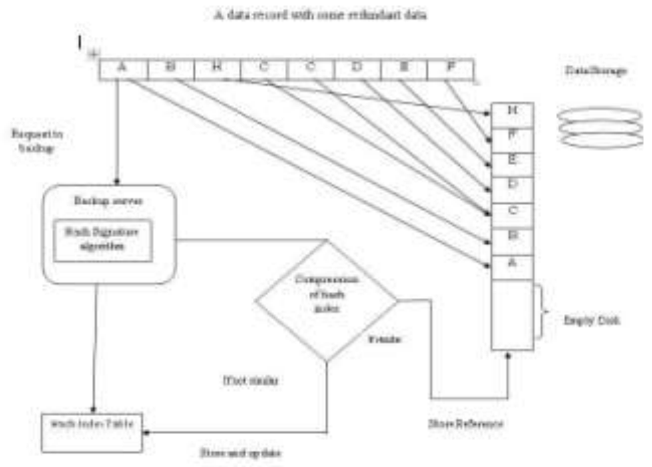

Fig. 1: De-Duplication process [4]

Method to reduce the required storage capacity because only the unique data is stored, refer figure 1.

II. DATA DE-DUPLICATION METHODOLOGY

De-duplication methodologies find duplicate data by cal-culating cryptographic hash of the incoming data. Hash is a fixed length representation of arbitrary length message. Fixed size hash comparison is much easier than the tedious task of comparing two big data chunks or data records. For each data chunk, de duplication server calculates its hash signature and searches this hash signature in already stored hash index record in the database. If there exist an entry for this signature in database records then, server put a reference of the existing record in the place instead of storing entire data chunk. Otherwise if there were no match found than server write this data file to disk and adds an entry for its hash signature in the hash index database.

III. TYPES OF DATA DE-DUPLICATION

Data de-duplication can be classified in many categories, four major classifications are on the following basis:

- Point of application
- Time of application,
- Granularity
- Algorithm

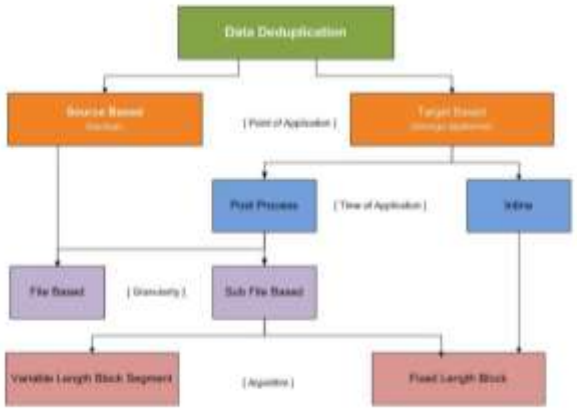

Fig. 2: De-Duplication Classification

The relation between these categories are shown in figure 2.

A. Point of Application De-duplication

There are two places where data de-duplication can occur i.e. source-de-duplication or target-dedulication.

1) Source-de-duplication (Client Side): If de-duplication is performed at client side where the data is created before sending it for backup at storage server, this type of method is known as source-de-duplication. The main benefit of this method is the reduction of bandwidth utilization. We can save large amount of bandwidth and reduce network traffic as well as low overhead on backup servers. Authors [5] used this method to create LBFS(low-bandwidth network file system). OpenDedup [6] and S3QL [7] are cloud based file system wrappers which uses this approach to reduce network bandwith.

2) Target-de-duplication (Server Side): If de-duplication is performed after transmission of data at the backu server end, this method is called as target-de-duplication or server side. Client side processing for de-duplication can cause performance bottleneck at client side for large amount of data hence server side de-duplication can reduce this problem by processing data at server side. Drawback with this scheme is that it requires high end hardware at server side which may be costly in some cases. This method is not an optimal scheme for low bandwidth networks. Venti [8] and ZFS [9] are two famous systems which uses target based method.

B. Time based De-duplication

There are major two methods for de-duplication techniques on the basis of time i.e. the de duplication is performed offline and Inline (online).

1) Offline de-duplication: The incoming data for storage is stored to disk first and then de-duplication happens after storing. After storage de-duplication process analyzes the stored data chunks and if the process detected duplicate chunks it will delete those chunks and put a pointer, pointing to existing data stored earlier. If no duplication has found then no change takes place. Offline method shows improved write performance due to almost no CPU intensive calculations during data write on disk. IBM Store Tank [10], Netapp ASIS [11] and EMC Clara [12] are typical example which are using this method for storage de-duplication.

2) Inline de-duplication: This methods search for dupli- cates in file data from the incoming request before being written to disk. The incoming data will remain in RAM until de-duplication detection get complete. If the data is unique it is transmitted to storage disk through IO channel as well as a new hash entry is created on hash index database. This method increases data write time for storage due to intensive CPU calculations at the time of IO. Venti [8], OpenDedup [6] and S3QL [7] uses inline de-duplication method for data archival.

Both methodology eliminates duplicate data from a storage system and frees up more space for new data. This reduction in storage requirement allows companies or organizations to withstand for long time without needing to purchase new storage. Low storage need requires lower costs in storage devices and result in cost saving.

C. Algorithm based De-duplication

The main logic behind de-duplication is the algorithm it uses for redundancy check. Basically two algorithms are used to check de-duplication:

- Delta Encoding
- Hashing

1) Delta Encoding based Dedulication: Earlier, Delta encoding is used as a main idea behind file compression methods like Bzip and 7Zip. For de-duplication purpose delta encoding compare two files for similarity and instead of writing entire second file, An encoded file of difference with first file is saved having a reference linking from the first file [13]. Delta encoding proved to be very effective when there is cases of storing highly similar file to the backup storage. The encoded file resulted after the delta de-duplication is known as patch file or diff file.

2) Hash based Dedulication: It is the widely used deduplication method. the main idea behind this method is to calculate hash of the data chunks and comare it with existing stored chunks to find the similarity between them. If hash values are same then the data being analyzed is already in storage and if no matching is found the chunk is unique and needs to be write on disk.

Hash based de-duplication can be classified into three methods:

- File Level De-duplication
- Block Level De-duplication
- Byte Level De-duplication

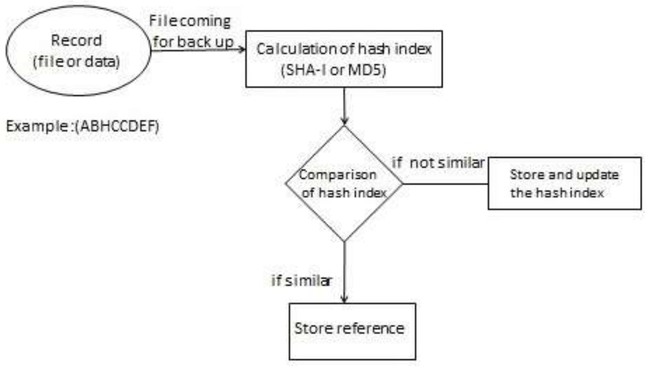

Fig. 3: File Level De-Duplication

a) File Level De-duplication:

In file level de-duplication, Whole file is used as a record. When a file is reached to storage server for back up, server calculates and compare the hash signature of incoming file to already stored files hash index database. If the hash value matched means file is already present, so the server store a reference of it otherwise store entire file and make an entry for hash signature of this file in the hash table, Ref. Figure 3.

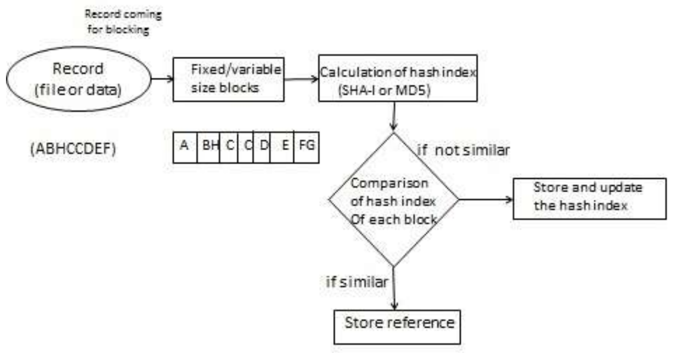
Record coming for blocking
Record (file or data)
Fixed/variable size blocks
Calculation of hash index (SHA-I or MD5)
(ABHCCDEF)
A BH C C D E FG
if not similar
Comparison of hash index Of each block
Store and update the hash index
if similar
Store reference

Fig. 4: Block Level De-Duplication

b) Block Level De-duplication:

In block-level data de- duplication method, first of all the data stream to be stored is divided into data blocks(can be fixed or variable), hash signature of these blocks are calculated and compared with the hash index database of stored data block, and server checks for the similar block with the previously stored data block. If the hash of the data block is unique, server write this block to disk, and store its hash value in the hash index database; otherwise, only store the pointer which will point to the existing block address refer figure 4. This location pointer drastically reduces the storage need as compared to storing the entire block of data. Contrary to file level de-duplication, block level hashing has low granularity, resulting a better de-duplication ratio.

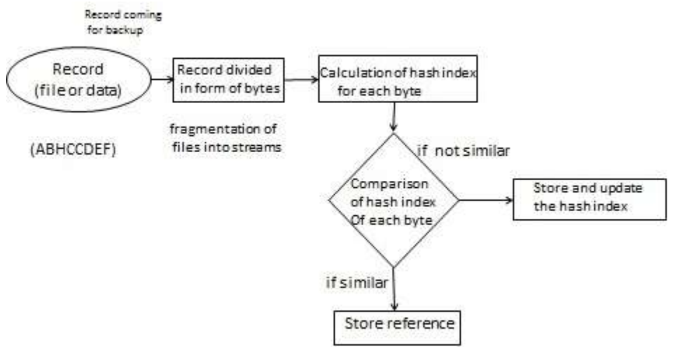
Record coming for backup
Record (file or data)
Record divided in form of bytes
Calculation of hash index for each byte
fragmentation of files into streams
(ABHCCDEF)
if not similar
Comparison of hash index Of each byte
Store and update the hash index
if similar
Store reference

Fig. 5: Byte Level De-Duplication

A negative point for this significant storage saving is that the block-level de-duplication need a large hash table since hashes will be generated for every block of data, which also means more computing time will be required when duplicates are being checked and the backup performance will be slower than File based de-duplication.

c) Byte Level De-duplication: The data to be stored is divided into the bytes and hash signature is calculated for byte, these hash signature are compared with the stored bytes signature on the server and take appropriate action according to matching of mismatching of records, refer figure 5. Byte level de-duplication performs with high de-duplication ratio as compared to file level de-duplication and block level de- duplication, but byte level de-duplication causes many perfor- mance issues, which are as follows:

- Size of the hash table will become very large.
- It may lead to large file fragmentation.
- Finally, byte level de-duplication will lead to perfor- mance degradation.

IV. VARIOUS RESEARCHES AND PRODUCTS

Presently, the exploration in de-duplication area focused on two facets. Major part is the effectiveness of data reduction, that is, to eliminate the redundant data as much as possible in order to minimize the storage capacity constraint. Another focus is on increasing the efficiency of de-duplication process to harness the optimal utilization of the hardware.

almost every existing traditional storage system uses file level de-duplication[14]. Very few of the existing architecture uses the source de-duplication method and offer the de- duplication in the users file system [15]. Because of file system de-duplication, there is delay in transmitting data to backup server, the rest of the available architectures which uses target de-duplication strategy have single system de-duplication that means at the server side a single system (Server) handles all the IO requests to archive data and keeps the data signatures for the number of disks attached to it.

Some earlier proposed architectures are VENTI [8], LBFS[15], SIS (single instance store)[16], and PASTICHEL[17]. VENTI and SIS uses fixed-size file partition method to divide the file into blocks. LBFS and PASTICHEL partition each file into variable sized blocks. Fixed-size partitioning technique is simple and easy to implement, but one of the significant drawback is that all the data blocks after the change point will get affected, and then misjudged as non-duplicate blocks.

Has collision is another problem in the available archi- tectures will lead to data corruption, e.g. two different data chunks can produce the same hash signature, which will lead to removal of unique block erroneously. Still methods like LBFS [15] uses hash algorithm (SHA-1), and these hashing algorithms are considered that the hash collision chances are negligible.

In the present scenario, many organizations are involved in working with data de-duplication concept. Few of the organizations are IBM, Symantec, and NetApp. NetApp de-duplication is a fundamental component of Data ONTAP operating system. NetApp de-duplication is the first that can be used broadly across many applications, including primary data, backup data, and archival data. Symantec also provides backup appliances that provide three step reduction processes. First it provides data de-duplication at source and targets both and reduces the data de-duplication complexity. IBMs TS7610 ProtecTIER De-duplication Appliance Express provides fast, reliable easy backup with de-duplication technology.

REFERENCES

[1] D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz, “The expanding digital universe,” 2007.

[2] D. Bhagwat, K. Eshghi, D. D. E. Long, and M. Lillibridge, “Extreme binning: Scalable, parallel dedu- plication for chunk-based file backup.” in MASCOTS. IEEE, 2009, pp. 1–9. [Online]. Available: http://dblp.uni-trier.de/db/conf/mascots/mascots2009.html/BhagwatELL09

[3] WikiPedia, “Data de-duplication.” [Online]. Available: http://www.wikipedia.com/de-duplication

[4] H. Biggar, “Experiencing data de-duplication: Improving efficiency and reducing capacity requirements,” The Enterprise Strategy Group, 2007.

[5] A. Muthitacharoen, B. Chen, and D. Mazieres, “A low-bandwidth network file system,” in ACM SIGOPS Operating Systems Review, vol. 35, no. 5. ACM, 2001, pp. 174–187.

[6] J. Bowling, “Opendedup: open-source de-duplication put to the test,” Linux Journal, vol. 2013, no. 228, p. 2, 2013.

[7] H. F. Deus, M. C. Correa, R. Stanislaus, M. Miragaia, W. Maass, H. De Lencastre, R. Fox, and J. S. Almeida, “S3ql: A distributed domain specific language for controlled semantic integration of life sciences data,” BMC bioinformatics, vol. 12, no. 1, p. 285, 2011.

[8] S. Quinlan and S. Dorward, "Awarded best paper! - venti: A new approach to archival data storage," in Proceedings of the 1st USENIX Conference on File and Storage Technologies, ser. FAST '02. Berkeley, CA, USA: USENIX Association, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083323.1083333

[9] J. Bonwick and B. Moore, "Zfs: The last word in file systems," 2007. [10] J. Menon, D. A. Pease, R. Rees, L. Duyanovich, and B. Hillsberg, "Ibm storage tanka heterogeneous scalable san file system," IBM Systems Journal, vol. 42, no. 2, pp. 250–267, 2003.

[11] C. Alvarez, "Netapp de-duplication for fas and v-series deployment and implementation guide," Technical ReportTR-3505, 2011.

[12] EMC and E. E. Services, Information Storage and Management: Storing, Managing, and Protecting Digital Information. LibreDigital,
2010.

[13] K. Jin and E. L. Miller, "The effectiveness of de-duplication on virtual machine disk images," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference. ACM, 2009, p. 7.

[14] G. Wang, Y. Zhao, X. Xie, and L. Liu, "Research on a clustering data de-duplication mechanism based on bloom filter," in Multimedia Technology (ICMT), 2010 International Conference on. IEEE, 2010, pp. 1–5.

[15] A. Muthitacharoen, B. Chen, and D. Mazie`res, "A low- bandwidth network file system," SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pp. 174–187, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502052

[16] W. J. Bolosky, S. Corbin, D. Goebel, and J. R.Douceur, "Single instance storage in windows&#174; 2000," in Proceedings of the 4th Conference on USENIX Windows Systems Symposium - Volume 4, ser. WSS'00. Berkeley, CA, USA: USENIX Association, 2000, pp. 2–2. [Online]. Available:http://dl.acm.org/citation.cfm?id=1267102.1267104.

[17] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche: Making backup cheap and easy," in Proceedings of the 5th Symposium on Operating Systems Design and implementation Copyright Restrictions Prevent ACM from Being Able to Make the PDFs for This Conference Available for Downloading, ser. OSDI '02. New York, NY, USA: ACM, 2002, pp. 285–298. [Online]. Available: http://doi.acm.org/10.1145/1060289.1060316