ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

A Study on Content based Chunking in Data Deduplication for Optimized Cloud Storage

Ruba S, Kalpana A M

Department of Computer Science and Engineering, Government College of Engineering, Salem, Tamil Nadu, India

Department of Computer Science and Engineering, Government College of Engineering, Salem, Tamil Nadu, India

ABSTRACT: With the massive growth of digital data in cloud storage, effective methods must be adopted to reduce hardware costs, meet bandwidth requirements, and improve storage efficiency. This can be achieved using data deduplication. Data deduplication is a technology that detects and eliminates duplicate copies of data. So, by storing less data, you'll need less hardware and make better use of your storage space. Data deduplication plays an important role in data transfer in various data-intensive network and cloud applications. Deduplication systems can use fixed-size or variable-size algorithms that primarily focus on space reduction requirements by storing only one copy of data. This white paper describes various chunking techniques used to divide a data stream or file into variable-sized blocks.

KEYWORDS: Cloud Computing, Cloud Storage, Data Deduplication,

I. INTRODUCTION

Basically, Data has been classified into two types namely ‘structured data’ and ‘unstructured data’ which are playing a major role in the recent trend. Normally the structure data can be easily organized includes website log data, customer call detailed records etc., Due to the rapid increase of social media usage and mobile usage the unstructured data cannot be easily organized includes blog data, social media interaction data, videos etc., So, unstructured data should be managed in a cost-effective way. Today in IT budgets, on an average of 13% of the money being invested on storage capacity. This digital data increase happens in all cloud deployment models and this requires more storage capacity, more costs, manpower and more time to handle data information like backup, replication and disaster recovery, more bandwidth utilization in transmitting the data across the network. This has created a big challenge to the cloud service providers to maintain all this massive amount of data that is nothing but big data and to offer these services at lower price to the customers. In veracity most of the data stored in the servers is often repeated. For example, a service may contain numerous instances of same data file, storing all these instances would require a huge amount of storage space. This problem can be solved by using Data Deduplication technique. Data deduplication stores only one unique instance of the data type on the disk or tape. In this method redundant data is replaced with a pointer to the unique data copy. It reduces the hardware used to store data and the bandwidth costs required for transmitting and receiving purposes. To understand the concept of de-duplication process along with application, various methods and technologies involved during each level implementation of this process.

The rest of this paper is organized as follows: Section 2 describes the techniques used in Data Deduplication. Section 3 represents various Chunking Algorithms. Section 4 represents the Conclusion.

II. DATA DE-DUPLICATION TECHNIQUES

Data deduplication is a technique for eliminating redundant copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth [1]. The process of data deduplication is explained in figure 1. Generally, there are five steps involved in data deduplication: a)*chunking* which uses a chunking algorithm to divide the file into chunks; b)*hashing or chunk fingerprinting* which generates the fingerprint and measures a hash value by using the MD5 or SHA1[2] for each of the chunks; c)*fingerprint indexing and querying* which stores the fingerprints

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

based on certain data structure for indexing and queries the index to find chunks of identical fingerprints – these chunks are considered as the duplicated chunks; d) *compression method* which eliminate the redundancy among the nonduplicate but very similar chunks in data deduplication system such as delta compression; e)*data storing and management* which stores new chunks and representative reference pointers for the duplicated chunks. There are two important measurements that evaluate a deduplication system: the deduplication ratio and performance. While other steps also play important roles, the chunking step has significant impact on the deduplication ratio and performance.

Classification of data deduplication is explained in figure 2. Deduplication can be performed either on source side or target side. When it performs on source side the redundancies are deleted from the file before transferred to target. It reduces the bandwidth consumption and improves the storage usage. When it performs on target side the redundancies are deleted from the file after transferred to target.

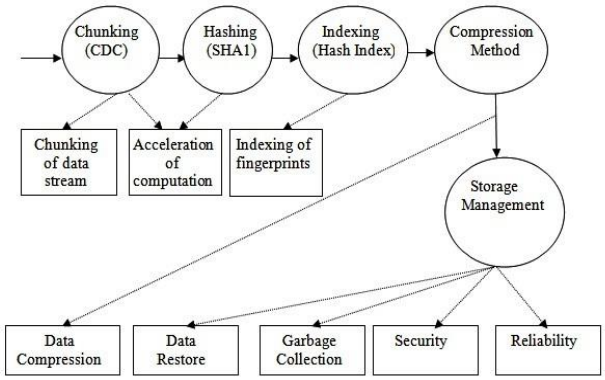

Figure 1 Process of Data Deduplication

Various benefits of Data Deduplication technologies are:
• Increases network efficiency.
• Lower storage space requirements.
• Storage cost is reduced.
• Reduced upload bandwidth.

Deduplication can be performed in two ways on the backup run, there are In-line and post-process [3].

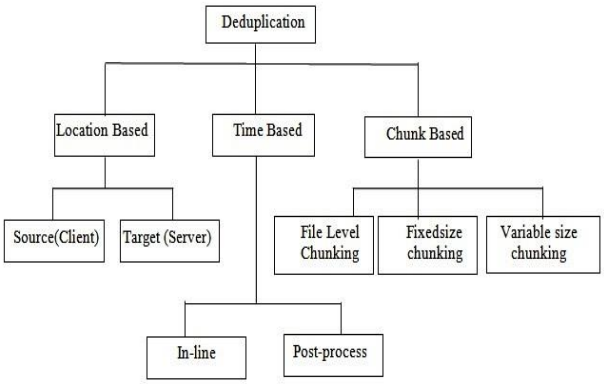

Figure 2 Classification of data deduplication

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

In-line deduplication: The inline deduplication can perform either on client side or receiver side before it is stored. When the block of data arrives, first it analyses the data, if it presents already then it is considered as a redundant data and immediately create a pointer to it. Suppose the block of data is unique, then it is stored into the disk or tape. It collides with the normal functioning of the computer. The benefits of inline deduplication are that it requires minimal storage space and reduces I/O overhead.

Post-process deduplication: The post process deduplication stores the data first and a separate process is used in which the data is to be analysed. Suppose the data is redundant then remove and replace it with a pointer; otherwise, no change is made. It doesn't collide with the normal functioning of the computer. This deduplication method reduces the data transfer time from source to storage but it needs more free space.

III. CHUNKING ALGORITHMS

The chunking techniques break the large data set into smaller chunks. Then hash algorithm is used to generate the unique identifier value which is assigned to each chunk. The process of splitting the data set into smaller, non-overlapping unit is called chunking and the resulting unit is called chunks. The new arrival chunk is compared with existing chunk using the unique identifier. If the unique ID matches, the duplicated data will be deleted and create a pointer to it; otherwise, new chunk will be stored. The main objective of chunking algorithm is to split the data set into smaller fragments. The data set may be a file, data stream or some other form of data. Different techniques for chunking are given below.

3.1 File-Level Chunking

In file level chunking or whole file chunking, an entire file is considered as single chunk rather than splitting the file into multiple chunks. In this method, only one unique ID is created for the entire file, if the file is unique, it is stored; if not, only a pointer to the existing file is stored. As it generates single index for the entire file, this method stores minimum number of index values, which in turn saves storage space, avoids maximum metadata lookup overhead and CPU usage. It also reduces the index lookup process as well as the I/O operation for each chunk. However, this method gets fail when a small part of the file is modified. Instead of generating the index for the modified portions, it computes the index for the entire file and moves it to the backup location. And it affects the throughput of the duplication system. Hence, this method is not suitable for backup systems and large files that change regularly.

3.2 Fixed Size Chunking

In fixed-size chunking algorithm, entire file needs to be partitioned into blocks with a fixed size. The chunk boundaries are based on offsets like 8, 16 Kbytes [4], [5]. To overcome the issue of file level chunking the fixed size chunking has been proposed. If a large file is modified in only a few bytes, only the changed chunks should be reindexed and moved back to its backup location. Fixed length chunking is used most often when general purpose hardware is involved for carrying de-duplication. Advantage of fixed size chunking is that it requires the minimum CPU overhead, and it is simple and fast. However, this approach generates more chunks for large file which need extra space for storing the metadata and the time for lookup of metadata is more. As it divides the file into fixed size, boundary (byte) shift problem occurs on the modified file.

3.3 Variable Size Chunking

This approach effectively resolves the issues of the fixed size chunking by breaking the file into multiple chunks of variable size based on the content, so it is more resistant to the insertion and deletion. In fixed size chunking, fixed boundaries are defined on the data set based on chunk size which do not change even when the data are altered where as in variable size chunking, different boundaries are defined, which are based on content that can shift when the content is altered or deleted. Hence, only less chunk boundaries need to be changed. Variable size chunking also named as Content-Defined Chunking (CDC) which is different from the fixed-size chunking. The following section reviews the various Content Defined Chunking algorithms.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

A. Basic Sliding Window Algorithm

Basic sliding window (BSW) eliminates [6] the boundary (byte) shift problem by making the chunk boundaries dependent on the local content of the file rather than distance from the beginning of the file. Chunk boundary is calculated based on the finger print algorithm. The sliding window algorithm works as follows: there is a predetermined integer D. A fixed size sliding window is moved across the file and at every position in the file; the content of the window is fingerprinted. Rabin algorithm is used to find out the fingerprint in sliding window algorithm. The main advantage of this BSW is that in every insertion ad deletion from the data set or file, the boundaries will not be affected. The main issue in BSW is the chunk size, the size of the chunk cannot be predicted, but it is possible to predict the probability of receiving a particular fingerprint. There are two other issues in the BSW algorithm.

1. The sliding window may define the break point in each boundary shifting process in the worst case if the data set or file contains a lot of continuous repeating string such as "ssssssss".
2. The sliding window cannot determine any breakpoint after the traversing the entire file. So the boundary shift problem is arising again.

In other words, the BSW algorithm has very poor controls on the variations of the chunk size and the chunk size may vary from very small to very large and it is not efficient and effective method to transmit very small or large. The main drawback is that waste only network resources while transmission can take place.

B. Two Thresholds and Two Divisors (TTTD) Algorithm.

TTTD was proposed [7]to resolve the issues of sliding window algorithm. This algorithm is the combination of SCM (Small Chunk Merge) and TD (Two Divisor algorithm). The main idea behind this algorithm is to modify the TD algorithm by applying the minimum threshold to the chunk size and it has to check the fingerprint matches for both main and backup, once the threshold is passed. A minimum and a maximum size limit is used for breaking a file into chunks for searching the duplicates.

It uses a minimum size and maximum size threshold for setting the maximum and minimum values of every chunk. Two divisor values namely main divisor or regular divisor and second divisor or backup divisor are also used for finding the boundary of the chunk. Main divisor determines the breakpoint and if it fails in doing so, then the second divisor finds the breakpoint. But TTTD has a drawback due to the second divisor which mostly produces breakpoints which are very near to the maximum threshold. A lot of time is wasted in performing unwanted calculations and comparisons for producing the result in larger sized chunk.

C. Two Thresholds and Two Divisors with Switch (TTTD-S) Algorithm

TTTD-S algorithm has proposed by T.S.Moh et.al [8], which overcomes the issues of the TTTD algorithm by introducing TTTD-S algorithm. It computes average threshold which is the average of maximum and minimum threshold. When TTTD-S algorithm reaches average threshold, the original values of main divisor and second divisor shrink to half. These values are switched back to the original values once the breakpoint is found. This method helps in avoiding unnecessary comparisons and calculations.

D. Bimodal Chunking Algorithm

Bimodal Chunking approach [9] is the improved version of CDC that combines the different size of chunks together. This method chunks the data stream or file into large chunks and then splits them into small chunks. The two principles are followed to eliminate the duplication in larger chunks. The first principle involves to create chunks of large average size and the data is in change region if some vicinity exists in both chunks that were encountered in past and that were not. Variation in vicinity size, and in how small the hidden data in a change region is chunked lead to different variant of bimodal algorithm. The second principle involves the small chunks to eliminate the boundary shift problem Bimodal chunking potentially improve the performance by increasing the chunk size (roughly 2-5 * in these

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

data set and 3-4 * in the 1.16 TB archival data set) and decreasing the storage cost for metadata. These algorithm increases the average chunk size while maintaining a duplication elimination ratio.

E. Multimodal Content Defined Chunking (MCDC) Algorithm

Multimodal Chunking was introduced as a new improved algorithm of Bimodal Content defined Chunking. MCDC [10] defines the optimal chunk size according to data size and compressibility. It is implemented in two ways: First is selective compression which breaks the data block with fixed size sample. Compression ratio (CR) is partitioned into three ranges and then three chunk sizes are to be selected, but this algorithm produces boundary shift problem, so this implementation method was not used. Second implementation was data block with variable size sampling that is a sequential process in which multiple candidates was generated and chosen by Uni-modal chunking. After that, it determines the CR for each candidate, then chunk boundary is selected for each segment and at last the fingerprint generated. It reduces the system throughput due to workload. The overall result on different dataset shows that MCDC minimizes the number of Chunks by 29.1% to 92.4% and also maintain the De-duplication efficiency with different chunk size.

F. Leap –based Content Defined Chunking Algorithm

An effective chunking algorithm has to satisfy two properties: chunks must be produced in a content-defined manner, and all chunk sizes must have equal probability of being selected. The CDC algorithm only splits the data stream when the content window before the breakpoint satisfies a predetermined condition to meet the *content defined condition*. In addition to that the average chunk size also affects the deduplication ratio. However, the chunk size cannot be too small, because it has to index the fingerprints of all the chunks of given data stream.

Leap-based chunking algorithm [11] was introduced to improve the performance of deduplication. This algorithm uses pseudo-random transformation that is used to define whether a window is qualified or not. This technique is the replacement of rolling hash function which is used in the sliding window content defined chunking. The difference between sliding window CDC and leap based CDC is computed by parameters M and Pw, where M represents the number of satisfied window and the Pw represents the probability that window is qualified. These two parameters determine the chunk size and performance of leap-based CDC. After the analysis and experiment the optimal parameter value of M=24 and Pw =3/4. The duplication ratio of sliding window CDC and leap-based CDC is similar but the performance of leap based is improved 50%-100% in Sandy bridge CPU and 10%-30% in older Wetmore CPU. By adding a secondary judgment function, the computational overhead is reduced and the de-duplication ration is maintained.

G. Asymmetric Extremum (AE) Algorithm

Asymmetric Extremum algorithm [12] was introduced to eliminate the limitations of MAXP-based and RABIN-based. The variation of chunk size and low chunking throughput are major issue in RABIN-based algorithm. To resolve this issue problem AE algorithm is introduces that is based on the observation that extreme value in asymmetric local range is not replaced by new extreme value, when deal with boundary shift problem. In AE, a byte has two attributes: position and value. Position number is assigned to every byte in the input data stream, and the position of the n^{th} byte ($1 \leq n \leq$ *stream length*) in the stream is *n*. At every interval of *N* consecutive characters/bytes in the input data stream is considered as a value and each byte in stream except last (N- 1) byte has a value assigned to it. The extreme value in the AE algorithm can be either the maximum value or the minimum value. For better results extreme value is the maximum value.

To find the first byte and the first byte after break point asymmetric extremum has two conditions: first condition is that the first byte or its value is greater than the value of all bytes before it and second is that the byte value is not less than the value of all bytes in its right window. To calculate the performance of asymmetric extremum, the three data sets are used network traffic, TAR file and movies file. It achieves a chunking throughput of 1GBytes/s, which is about 3× higher than the state-of-the-art CDC algorithms, with comparable or higher deduplication efficiency.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

AE algorithm significantly improves the chunking throughput while providing better deduplication efficiency by using the local extreme value in a variable sized asymmetric window to resolve the above-mentioned boundaries-shift problem. With a variable-sized asymmetric window, instead of a fix-sized symmetric window, the AE algorithm finds the extreme value in the window and thus requiring only one comparison and two conditional branch operations per byte perused. It has smaller chunk size variance than existing CDC algorithms and enforces no limitation on chunk size. Moreover, AE is able to remove low-entropy strings than the other algorithms.

H. Fast Content Defined Chunking Algorithm (FastCDC)

Fast Content Defined Chunking [13] was proposed to improve the performance of Gear-based CDC approach. Generally, there are three metrics needed for the evaluation of CDC performance, they are chunking speed, the average generated chunk size and deduplication ratio. The basic idea behind FastCDC is the combined use of three key techniques, namely, simplifying and enhancing the hash judgment to address from challenges facing Gear-based CDC, skipping sub-minimum chunk break point to further speed up CDC, and normalizing the chunk-size distribution in a small specified region to address the problem of the decreased deduplication ratio stopping from the break point skipping. By using the combination of the three techniques, FastCDC is about 10x faster than the open-source Rabin-based CDC, and about 3x faster than the state-of-the-art Gear- and AE-based CDC, while succeeding nearly the same deduplication ratio as the classic Rabin-based approach.

IV. CONCLUSION

This paper has reviewed various existing work performed on the data deduplication. Generally, the file level chunking technique is efficient for small files deduplication, but not suitable for the large file deduplication. The fixed size chunking algorithms do not allow any data modification like insertion and deletion whereas the variable size chunking algorithms overcome the problem of fixed size chunking by creating the boundaries based on the content of the files. This paper also reviewed on various variable chunking algorithms. From this review, it is obvious that still a lot of challenges need to be addressed in the future researches.

REFERENCES

1.Jin Li, Xiaofeng Chen, Xinyi Huang, Shaohua Tang, Yang Xiang, Mohammad Hassan, Abdulhameed Alelaiwi, "SecureDistributed Deduplication Systems with Improved Reliability", IEEE Transactions on Computers Volume: PP,Year – 2015

2.Qinlu He, Zhanhuai Li, Xiao Zhang,"Data De-duplication Techniques ", International Conference on Future Information Technology and Management Engineering, IEEE, pp. 430-433, 2010.

3.Rashmi Vikraman, Abirami S," A Study on Various Data De-duplication Systems", International Journal of Computer Applications, Volume 94, No4, May 2014.

4.Kubiatowicz J et al (2000) Oceanstore: an architecture for global store persistent storage. In: Proceedings of the 9th international conference on architectural support for programming languages and operating systems.

5.Quinlan S, Dorwards S (2002) Venti: a new approach to archival storage. In: Proceedings of USENIX conference on file and storage technologies.

6. Kave Eshghi, Hsiu Khuern Tang, ,"A framework for analyzing and improving content based chunking Algorithms" Technical Report TR 2005-30, Hewlett-Packard Development Company, http://www.hpl.hp.com/techreports/2005/HPL-2005-30R1.html.

7. Cai Bo, Zhang Feng Li, and Wang can," Research on Chunking Algorithms of Data De-duplication,proceeding of the ICCEAE, Springer, AISC 181, pp. 1019-1025, 2013.

8. Teng-Sheng Moh, BingChun Chang," A Running Time Improvement for the Two Thresholds Two Divisors Algorithm" ACMSE '10, April 15-17, 2010.

9. Eshghi, K. A. 2005. Framework for Analyzing and Improving Content-Based Chunking Algorithms.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 5, May 2018

Technical Report HPL-2005-30(R.1), Hewlett Packard Laboratories, Palo Alto, CA.

10. Jiansheng Wei, Junhua Zhu, Yong Li," Multimodal Content Defined Chunkingfor Data Deduplication",https://www.researchgate.net/publication/261286019, Research gate, 2014.

11. Chuanshuai Yu, Chengwei Zhang, Yiping Mao, Fulu Li, "Leap Based Content Defined Chunking- Theory and Implementation", 31st Symposium on Mass Storage Systems and Technologies (MSST), IEEE, pp. 1-12, 2015.

12. Yucheng Zhang, Hong Jiang, Dan Feng, Wen Xia, Min Fu, Fangting Huang, Yukun Zhou,"AE: An Asymmetric Extremum Content Defined Chunking Algorithm for Fast and Bandwidth-Efficient Data De-duplication", 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, pp. 1337- 1345, 2015.

13. Wen Xia, Huazhong University of Science and Technology and Sangfor Technologies Co., Ltd.; "FastCDC: a Fast and Efficient Content-Defined Chunking Approach for Data Deduplication" on 2016 USENIX Annual Technical Conference (USENIC ATC '16). June 22–24, 2016 • Denver, CO, USA 978-1-931971-30-0