ISSN: 2321-7782 (Online)
Impact Factor: 6.047
Volume 4, Issue 6, June 2016
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Performance Evaluation of Various Chunking Techniques in Data De-duplication

Shobha
Dept. of Computer Science and Engineering
U.I.E.T., KUK
Kurukshetra – India

***Abstract:** The term "BIG DATA" requires new techniques and technology to capture, manage and process data within a tolerable elapsed time. De-duplication solves our purpose by storing and processing the big data. The first step of De-duplication system is to classify the big data according to the format of data. In de-duplication system various fixed and variable size algorithms are available that mainly focus on to reduce the storage space requirement by storing only single copy of content. Content Defined Chunking (CDC) techniques help to reduce the chunk size variation and maintain the Data elimination ratio. This paper review various content based chunking techniques that partition data stream into chunks of variable size.*

***Keywords:** Data de-duplication, Leap based chunking, Bimodal CDC, Multi-modal CDC, Asymmetric Extremum.*

I. Introduction

Big data is a broad term that is difficult to analyze and solve. It is a combination of structured and unstructured data that requires a set of techniques and technology to efficiently handle and process this massive amount of data. According to a report from International Data Corporation (IDC), in 2011, the overall created and copied data volume in the world was 1.8ZB (≈ 1021*B*), which increased by nearly nine times within five years [1]. The main challenge of big data is how to store and process the large amount of data. There are various solutions available but each has its own significances. Data de-duplication is one of solution in which the unnecessary data is to be deleted. It requires some algorithms that detect which data is useful and which is not.

The remainder of this paper is organized as follows. Section II represents Data de-duplication techniques. Section III represents Chunking Techniques. Section IV represents overview of Related Work of content based chunking. The paper is concluded with future work in Section V.

II. Data De-Duplication Techniques

Data de-duplication is technique through which similar copy of data is detected and eliminated. It reduces the amount of storage space by dividing a file into chunks and measures a hash value by using the MD5 or SHA1 [2]. De-duplication can be performs either on source side or target side. When it performs on source side the redundancies are removed from the file before transmitted to backup target. It minimizes the bandwidth consumption and improves the storage usage. Target based de-duplication perform the redundancies removal from backup server upon which the backup software resides. De-duplication can be performing at two stages: In-line and Post-process [3].

A. In line de-duplication

The inline De-duplication can perform on client side or target side before it is written. When a block of data is arrives it is analyses, if it already present then it is throw away as redundant block and create a pointer to it. If the block of data is unique, then it is written to the storage. The advantages of inline de-duplication that it requires less storage space and minimize I/O overhead.

B. Post process de-duplication

The post process de-duplication stores the data firstly and then written it. A separate process is followed in which the data is to be analyzed. If data is redundant then delete and replace it with a pointer; otherwise no change is made. This method reduces the time of data transfer from source to storage but it requires more free space.

III. Chunking Techniques

For each file the first step of de-duplication system applies chunking techniques that break a file into smaller chunks. Then hash algorithm is applied to generate the unique ID of chunk. The process of dividing the data stream into smaller, non overlapping units is called chunking and the resulting unit is called chunks. Different approaches for the chunking are given below:

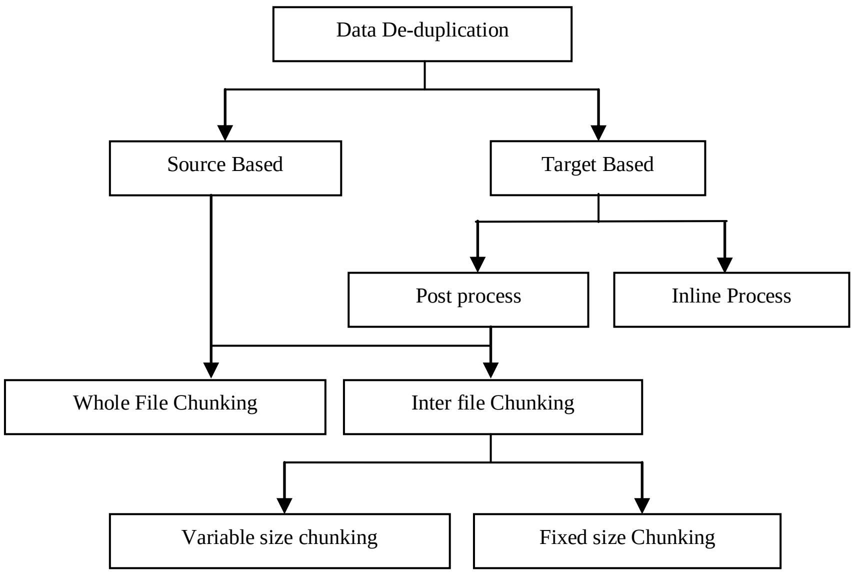

Fig. 1 Classification of De-duplication System

A. File-based Chunking

De-duplication can be applied complete file in which the complete file is taken as single chunk. If the file is unique, it is stored and the index is updated; if not, only a pointer to the existing file is stored. The other is breaking the file into smaller units.

B. Fixed Size Chunking

In this the complete file is partition into fixed size chunks from the beginning of the file based on offset likes 4, 8 KB, ETC. But a single insertion or deletion in the file will shifted all data from the original position. This is known as boundary shift problem, for every new insertion or deletion could change the chunk boundary.

C. Variable Size Chunking

To resolve the problem of fixed size chunking the variable size chunking has been proposed. The insertion and deletion in main file only Shift the single chunk and rest of data are not effect. It divides the data stream depending on the content of the file. Basic Sliding window is used to divides the data stream according to the content. The BSW is moves across the file and compute the hash values. These hash values are compared with previously stored hash values. Then duplicate chunk are not stored only the pointer to previously stored chunk are to be indicated and original chunks are to be stored in the storage devices. The next section reviews the various content defined chunking algorithms.

IV. Related Survey

A low bandwidth network file system (LBNFS) framework [4] was proposed to analyze content based chunking algorithm. In the traditional system, a long sequence is broken into fixed size block. The algorithm relies only on the byte sequence that is stable under local modification. This is also a stateless chunking algorithm , the main problem in this approach is that when even a single byte is inserted or deleted in the middle of the sequence, the block boundaries are shifted to the different block. So after every insertion or deletion in the file result would be a different block.

A. Basic sliding window (BSW) Algorithm

BSW avoids [5] the boundary shift problem by making chunk boundaries dependent on the local content of the file. There is a pre-determined integer D. A fixed size sliding window is moved across the file. At every position in the file, the content of window is fingerprinted. To find the fingerprint in sliding window Rabin algorithm is used. The main advantage of this scheme is that in every insertion or deletion from the file, the boundaries are not affected. There are two main problems in the BSW algorithm:

- The sliding window may determine the breakpoint in each shifting in the worst case if the file contains a lot of continuous repeating string such as “*aaaaaaaaaa”*.
- The sliding window cannot find any breakpoint after traversing entire file, so the boundary shift problem is arise again.

In other words, the BSW algorithm has very poor controls on the variations of chunk-size and the chunk-size may vary from very small to very large. It is not efficient and effective to transmit very small data or large. They only waste network resources while these situations happen.

B. Two Thresholds and Two Divisors (TTTD) Algorithm

TTTD was introduced [6] that solves BSW problems. It is the combination of SCM (Small Chunk Merge) & TD algorithm. The fingerprint match is searches for both main and backup, once the threshold is passed. The TTTD used the four parameter D (main divisor), D’ (Backup divisor), T_{min} (minimum chunk size threshold) and T_{max}(maximum chunk size threshold). The TTTD algorithm works good on the real and non-random data, but in random data it works equally as the SCM done set.

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

An improvement in the TTTD (two divisors and two thresholds) algorithm was presented to remove the limitation of TTTD algorithm [7] by adding a new parameter SwitchP instead of second divisor. TTTD algorithm uses the maximum and minimum that increases the number of chunks so that the searching time in lookup-table is increase. The probability by which cut-point is find by second divisor is double than the first divisor. The new TTTD-S algorithm removes the second divisor and adding new parameter SwitchP that switch the value of two divisors at specific position. The different tar file, Text file and Source code was taken for the experiment. The overall running time by using TTTD-S algorithm reduces to 7% and the 50% of larger chunks reduced to the expected chunk size when compare with TTTD algorithm.

D. Bimodal chunking algorithm

Bimodal Chunking presented as opposed to the (uni-modal) baseline CDC approach [8]. This is the improved version of CDC that mixes chunks of different average size together. The algorithm first chunks the data stream into large chunks and then splits parts of them into small chunks. The two principles are takes to eliminate the duplicity in larger chunks. The first principle said that an algorithm should produce chunks of large average size when in an extended region of previously unseen data. The data is in change region if some vicinity exist in both chunks that were encountered in past and that were not. Variation in vicinity size, and in how small the unseen 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. The data set for testing consisting of 1.16 terabytes of full Netware backups of hundreds of user directories over a four month period. Bimodal chunking has the potential to improve the performance. The increase chunk size (roughly 2-5 * in these data set and 3-4 * in the 1.16 TB archival data set) decrease the storage cost for metadata. These algorithm increase average chunk size while maintain a duplication elimination ratio.

E. A Multimodal Content Defined Chunking (MCDC) Algorithm

Multimodal Chunking was introduced as a new enhancement in Bimodal Content defined Chunking. MCDC [9] determines the optimal chunk size according to data size and compressibility. It is implemented in two ways: First is selective compression that divides 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 technique has boundary shift problem so the second way was used. Second implementation was data block with variable size sampling that is a sequential process in which multiple candidates was generated and selected by Uni-modal chunking. After that determine the CR for each candidate, then chunk boundary is selected for each segment and at last the fingerprint generated. It lowers the system throughput due to workload. The overall result on different dataset shows that MCDC reduces the number of Chunks by 29.1% to 92.4% and also maintain the De-duplication efficiency with different chunk size.

F. Leap-based CDC Algorithm

Leap based chunking algorithm improves the de-duplication performance by adding another judgment function [10]. The pseudo-random transformation is used to define whether a window is qualified or not. This is the replacement of rolling hash function that is used in the sliding window CDC. The Transformation came from the locality sensitive hashing and the theorem that the sum or the difference of normal distribution is still a normal distribution. The difference between sliding window CDC and leap based CDC is determined by parameters M and P_w ,where M is the number of satisfied window and the P_w is the probability that window satisfied. 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 P_w =3/4. The sys dataset is collected from 20 users. The office, pdf, music and video data set is also taken from the real environment. The duplication ratio of sliding window CDC and leap based CDC is same but the performance of leap based is improve 50%-100% in Sandy bridge CPU and 10%-30% in older Westmere CPU. By adding a secondary judgment function the computational overhead is reduce and the de-duplication ration is maintained.

G. Asymmetric Extremum (AE) Algorithm

It was introduced to eliminate the limitations of MAXP-based and RABIN-based. This new CDC algorithm called as asymmetric extremum (AE) algorithm [11]. The low chunking throughput and chunk size variation are main problem in MAXP and RABIN-based algorithm. To tackle these problem AE algorithm is proposed 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 attribute: position and value. In input data stream each byte is treated as a value and each byte in stream except last (N-1) byte has a value attached to it. To find the first byte and the first byte after cut point AE has two conditions: first is the first byte or its value is greater than the value of byte before it and second is that the byte value is not less than the value of all bytes in its right window. To evaluate the performance of AE the three data sets are used network traffic, TAR file and movies file. The matrices used for evaluation are chunking throughput and byte saved per second (BSPS). The maximum threshold reduces the chunking throughput and the performance of BSPS in also increase due to the efficient chunking scheme.

V. CONCLUSION AND FUTURE SCOPE

This paper has reviewed different content based chunking techniques of data de-duplication. The fixed size chunking does not allow data modification such as insertion or deletion whereas content based chunking allows any kind of modification in the data stream. The property of being robust against byte shifts in data make content based chunking a better approach than fixed size chunking. In future, researchers can work to improve de-duplication ratio in less time as content based is time consuming approach.

References

1. Min Chen, Shiwen Mao and Yunhao Liu," Big Data: A Survey", Springer Science, Springer, pp. 171–209, 2014.
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, No 4, May 2014.
4. Athicha Muthitacharoen, Benjie Chen, and David Mazieres, "A Low-Bandwidth Network File System" , in the proceeding of the 18th ACM symposium on Operating System principles (SOSP 01) Review, vol. 35, no. 5, pp. 174- 187,2001.
5. 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.
6. 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.
7. Teng-Sheng Moh, BingChun Chang," A Running Time Improvement for the Two Thresholds Two Divisors Algorithm" ACMSE '10, April 15-17, 2010.
8. Erik Kruus, Christian Ungureanu, Cezary Dubnicki,"Bimodal Content Defined Chunking for Backup Streams",Fast 10 Preceding of the 8th USENIX Conference on file and Storage Technologies ,USENIX ,pp. 239-252, 2010.
9. Jiansheng Wei, Junhua Zhu, Yong Li," Multimodal Content Defined Chunking for Data Deduplication", https://www.researchgate.net/publication/261286019, Research gate, 2014.
10. 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.
11. 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.