Two-Threshold Chunking (TTC): Efficient Chunking Algorithm For Data Deduplication For Backup Storage

Anand Bhalerao, Ambika Pawar

Abstract: Large amount of data gets generated every day and storing that data efficiently becomes a heuristic task. Backup storages are more prominently used media for storing every day, the generated data. The significant amount of data that is stored in the backup storage is redundant and leads to the wastage of storage space. Storage space can be saved and processing speed of backup media can be improved using deduplication and variable size chunking. Various chunking algorithms have been presented in the past to improve deduplication process. This paper presents the design of an efficient chunking algorithm to achieve high throughput and to reduce processing time.

Index Terms: Backup Storages, Chunk Level Deduplication, Chunking Algorithm, Cloud Storages, Deduplication, Redundant data, Variable-Size Chunking

1. INTRODUCTION

In cloud data storages, where the chances of having more redundant data is high, the data deduplication technique is used because of its scalability and efficiency as compared to conventional compression techniques such as Huffman Compression [1] and LZ Compression [2]. There are two major benefits of data deduplication, first one is it detects and removes the chunk level redundancy unlike conventional compression techniques which works at the string/byte level. Other benefit is that it detects redundant chunk with the help of fingerprint whereas the traditional compression techniques [3] which does byte level comparison which increases the processing time.

2 DATA DEDUPLICATION

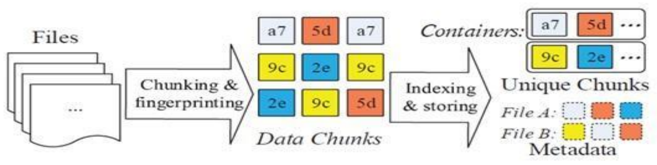

Fig. 1 *Data Deduplication*

Figure 1 shows the steps of data deduplication [3] Deduplication has four steps: i. chunking, ii. Fingerprinting, iii. Indexing and iv. Writing. In first step file gets divided into chunks with the help of chunking algorithm. For every generated chunk fingerprint/hash value is calculated using hashing functions like SHA1, SHA2 or MD5. Further in indexing step these calculated hash values are compared with existing hash values to find the similar chunk. Redundant chunk get discarded and only non redundant chunks are stored in the storage.

- *Anand Bhalerao has pursued master's degree program in Computer Science & Engineering from Symbiosis Institute of Technology, Symbiosis International (Deemed) University, India, E-mail: anand.bhalerao@sitpune.edu.in*
- *Ambika Pawar has pursued PhD degree program in Computer Science & Engineering from Symbiosis Institute of Technology, Symbiosis International (Deemed) University, India, PH- 020 39116482. E-mail: ambikap@mail.com*

2.1 Chunk-Level Deduplication

In this section we will discuss different chunk-level Deduplication techniques [10]: Figure 2 shows the classification of chunking algorithms and their characteristics. There are two types of chunking algorithms: i. Variable-size and ii. Fixed size chunking Fixed-size chunking: In this type of chunking, file/data stream gets divided into fixed size chunks. These algorithms suffer from low chunking throughput as it gets affected due to addition or deletion of data. Variable-size chunking: In this type of chunking file/data stream gets divided into variable size chunks. As compared to fix sized chunking, variable size chunking generates less redundant chunks due to less impact of addition or deletion of data on file chunks.Boundary shift problem occurs when new data gets added to the file/data stream and that leads to change in boundary breakpoints, which eventually affects the chunking throughput. Variable-size chunking has fewer boundaries shift problem as compared to fixed-size chunking. In fixed-size chunking has fix chunk boundary, therefore the indexes required for the chunks are less and thus requires less storage to store indexes. In variable-size chunking the chunk boundary is variable and may change when the file gets updated. Therefore, variable-size chunking generates more indexes as compared to the fixed-size chunking. Table 1. Presents comparison of fixed size and variable size algorithms.

TABLE 1 COMPARISON OF CHUNKING ALGORITHMS

Type of chunking	Advantages	Disadvantages
Variable-size	Fewer boundary shift	Generates more indexes
Fixed-size	Generates Few indexes	More boundary shift

2.2 Various Variable-Size Chunking Algorithms

There are various chunking algorithms available like Rabin, TD, TTTD, MAXP, Bimodal [11] and MCDC [12] etc. and few of them are explained below:

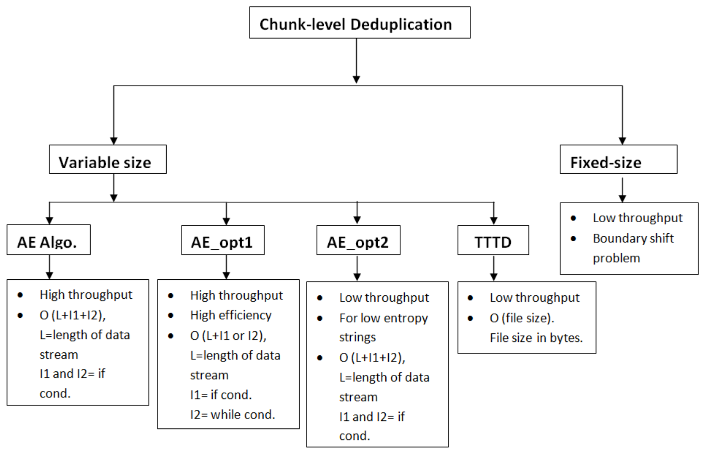
Chunk-level Deduplication
Variable size
Fixed-size
Low throughput
Boundary shift problem
AE Algo.
AE_opt1
AE_opt2
TTTD
High throughput
O (L+I1+I2), L=length of data stream I1 and I2= if cond.
High throughput
High efficiency
O (L+I1 or I2), L=length of data stream I1= if cond. I2= while cond.
Low throughput
For low entropy strings
O (L+I1+I2), L=length of data stream I1 and I2= if cond.
Low throughput
O (file size). File size in bytes.

***Fig. 2** Classification of Chunking Algorithms*

Rabin Fingerprinting Algorithm: As proposed by Michael O. Rabin [6] has implemented fingerprints using polynomials. It has used in CDC (Content Defined Chunking) algorithm. It has limitation of low chunking throughput which affects deduplication efficiency.Two Divisors (TD): This algorithm [7] predefined positive integer values, D and D' are used to determine the chunk boundaries. Using these chunk boundaries large chunks are divided into small chunks using secondary and smaller divisor. So there is more probability of getting same fingerprint. TD tries to overcome the boundary shifting problem of Breaking Large Chunks into Fixed Size Sub-Chunks algorithm [7]. TD has two divisors D and D' where D is secondary backup divisor and D is main divisor and D' is smaller than D, TD uses these divisors to find the breakpoints. TD algorithm scans the data stream; it starts from the last chunk boundary. It matches D and D' fingerprint at each position by memorizing the position when the last D' fingerprint match. The position is declared as a breakpoint if D-match is found before reaching the threshold Tmax. In case the D match is not found before reaching the Tmax threshold then earlier D' match position is used as the breakpoint else e. the threshold should be used as a breakpoint.Two Thresholds, Two Divisors (TTTD): TTTD algorithm [7] utilizes concept of TD algorithm and the SCM (Small Chunk Merge) [7] algorithm. TTTD modifies the TD algorithm, it considers minimum threshold for chunk size. This algorithm searches and compares fingerprints in backup and main divisor, until it passes the threshold. This algorithm takes four input values: D (main divisor), D' (backup divisor), Tmin (minimum chunk size threshold) and Tmax (maximum chunk size threshold).

***TABLE 2** ADVANTAGES AND DISADVANTAGES OF CHUNKING ALGORITHMS*

Algorithm	Advantages	Disadvantages
TD	Reduces chunk size variance	Misses out redundant chunks
TTTD	Less computational overhead	Low chunking throughput
MAXP	Reduces computational overhead	Low chunking throughput
AE	High throughput	Takes more time to process chunks

MAXP: [8] Overcomes the problem of chunk-size variance in Rabin-based chunking algorithm by working on local extreme values. MAXP is frequently applied in network storage system. It divides windows at byte level. MAXP has an approach in which it find the local extreme value using which MAXP algorithm revisits some already compared bytes. Thus MAXP reduces the throughput.AE Algorithm: [9] Addresses the boundary shifting problem occurring in various variable-size chunking algorithms. In AE algorithm asymmetric window of variable size is used for addressing the boundary shifting problem. AE is very fast as it does not backtrack and does single comparison. Also it has smaller chunk size variance as compared to other variable-size chunking algorithms

3 PERFORMANCE METRICS

In this section various performance metrics for finding the efficiency of different chunking algorithms have been presented.DER (Deduplication Elimination Ratio): DER is a measure of the deduplication efficiency. DER is calculated using input data size and total size of all non-redundant chunks generated as given in the following equation:

DER = (Input Data Size) / (Total Size of non-redundant chunks)

CPU Overhead: CPU/Computation overhead is one of the important metrics of the Deduplication technique. In chunking phase every byte has been compared in order to find the chunk boundary. This comparison incurs high computational cost.Processing Time: Total time required for completing deduplication process, including all the four phases is called as processing time. This metric can be the bottleneck because of the chunking algorithm. So it is important to design efficient chunking algorithm. Number of Chunks Generated: Total number of chunks generated affects the performance of the duplication technique. The algorithm should not generate too less or too high number of chunks. Too high number of chunks will require more comparisons to find duplicate chunks, which eventually lead to high processing time. Too less number of chunks affects the chunking throughput as redundancy will get reduced.

4 PROPOSED TWO-THRESHOLD CHUNKING (TTC) ALGORITHM

In this section the proposed Two-Threshold Chunking (TTC) Algorithm [13][14][15] for chunking in deduplication is discussed. TTC has two threshold values (Tmin and Tmax) which are used to restrict the chunks formation, so that it will not have more chunk size variance. To get the breakpoint for the chunk, the algorithm finds the maximum point in the incoming data stream and the point which satisfies both the Tmin and Tmax value will be considered as the breakpoint.

4.1 Flow of Two-Threshold Chunking Algorithm

Fig. 3 Shows how Two Threshold Chunking algorithm works. 'A' is the starting point for the incoming data stream. Tmin and Tmax are the threshold values. First the algorithm will find the maximum value in the data stream, let say 'B' is the maximum point. Then declare 'B' as the breakpoint for the chunk and generate chunk from 'A' to 'B'. For the next chunk consider next point after 'B' as the starting point and try to find maximum point from it and repeat this till the end of the data stream.

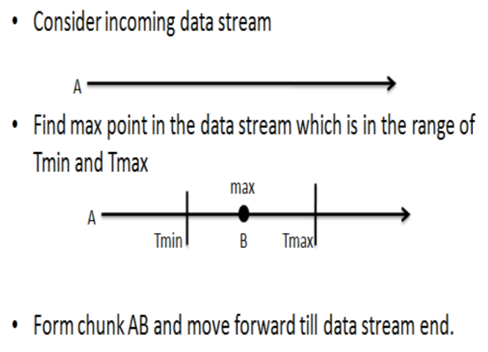
Consider incoming data stream
A
Find max point in the data stream which is in the range of Tmin and Tmax
max
A
Tmin
B
Tmax
Form chunk AB and move forward till data stream end.

Fig. 3 *Flow of Two-Threshold Chunking Algorithm*

4.2 Pseudo Code for Two-Threshold Chunking Algorithm

```
Input: input data, D; Length of the input data, L;

Predefined value: Tmin, Tmax;

Function Two_Threshold (D, L, Tmin, Tmax)
    1.   breakpoint = 1;
    2.   Max.value = D[breakpoint];
    3.   breakpoint = breakpoint + 1;
    4.   For ( ; breakpoint < L; breakpoint++)
           If (Max.value < Tmin or Max.value > Tmax)
             return breakpoint; //breakpoint
           Else
             Max.value = D[breakpoint];
          End if
End function
```

Algorithm takes input as the data stream, Tmin and Tmax value. Max.value will store the maximum value in the data stream and Tmin and Tmax are the threshold values.

5 RESULT AND DISCUSSION

To evaluate the performance of the algorithm, experimental setup consists of Intel Core i5-3320M CPU @ 2.60 GHz system with 4GB RAMS. A file is given as the input to the algorithm. Comparison is mainly based on the Average chunks size with DER factor, Processing Time and number of chunks generated. The figures given below show the comparison between AE_Opt1 and Two-Threshold Algorithm.

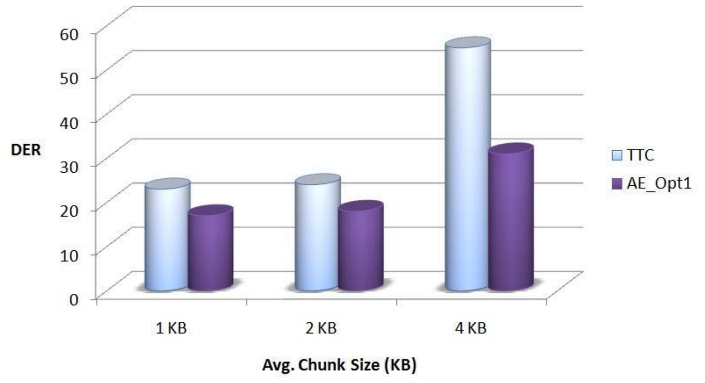

Fig. 4 *DER for the TTC and AE_Opt1*

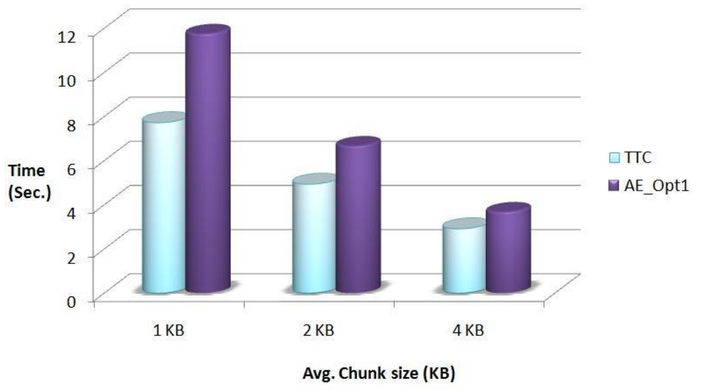

Fig. 5 *Time required for TTC and AE_Opt1*

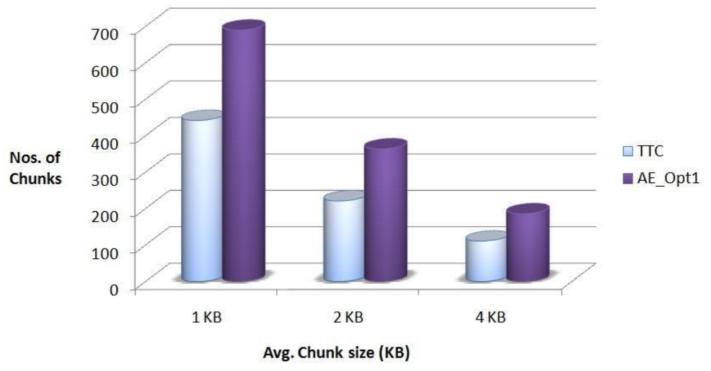

Fig. 6 *Number of chunks for TTC and AE_Opt1*

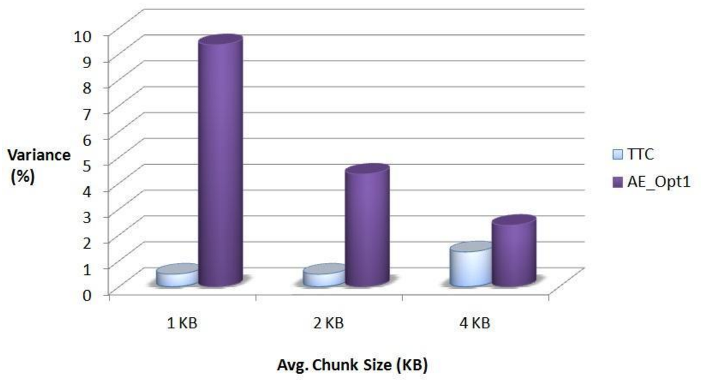

Fig. 7 *Chunk size variance for TTC and AE_Opt1*

In Figure 4, the DER factor for Two-Threshold Chunking Algorithm is higher in all the cases. Figure 5 shows the deduplication process time with the given chunking algorithm. For smaller chunk size takes more time to process the data In Figure 6 Number of chunks for TTC and AE_Opt1In Figure 6, the numbers of chunks generated while chunking are shown. The process should not generate so many chunks that it will affect the performance of the deduplication. In Figure 7, chunk size variance between TTC and AE_Opt1 is given. Due to the use of threshold values in TTC the chunk size variance is low compared to AE_Opt1. Threshold values limit the chunk size limit which helps to lower the chunk size variance.

TABLE 3 A
COMPARISON BETWEEN AE_OPT1 AND TTC ALGORITHM FOR 1 KB AVG. CHUNK SIZE

Properties/Algorithms	AE_Opt1	TTC
DER	17	23
Time(Sec.)	11.74	7.72
No. Of Chunks	690	442

TABLE 3 B
COMPARISON BETWEEN AE_OPT1 AND TTC ALGORITHM

Properties/Algorithm	AE_Opt1	TTC
DER	Low	High
Time	More	Less
No. Of chunks	More	Moderate

Table 3 (A) and 3 (B) summarizes the overall comparison between Two-threshold and AE_Opt1 algorithm.

6 CONCLUSION

In proposed algorithm the threshold values help to maintain the chunk generation in given chunk size limit. The maximum point helps to find the breakpoint for the algorithm. The proposed Two-Threshold Chunking algorithm works better than AE_Opt1 algorithm with respect to time, throughput, number of chunks and chunk size variance.

REFERENCES

[1]. J. Ziv and A. Lampel. A universal algorithm for sequential data compression. IEEE Transaction on information theory, vol. 23, pp. 337-343, 1977.

[2]. Huffman coding, Available at: https://www.cs.auckland.ac.nz/software/AlgAnim/huffman.html

[3]. Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, and Yukun Zhou. A Comprehensive Study of the Past, Present, and Future of Data Deduplication. In: Proceedings of the IEEE, 2016.

[4]. Quinlan, S., Dorward, S. Venti: a new approach to archival storage. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST), 2002.

[5]. Bo C., Li Z.F., Can W. Research on Chunking Algorithms of Data De-duplication. In: Yang G. (eds) Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering. Advances in Intelligent Systems and Computing, vol 181. Springer, Berlin, Heidelberg 2013.

[6]. Michael O. Rabin. Fingerprinting By Random Polynomials. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.

[7]. K. Eshghi and H. K. Tang. A framework for analyzing and improving content-based chunking algorithms. Hewlett-Packard Labs Technical Report TR, vol. 30, 2005.

[8]. N. Bjørner, A. Blass, and Y. Gurevich. Content-dependent chunking for differential compression, the local maximum approach. Journal of Computer and System Sciences, vol. 76, no. 3, pp. 154–203, 2010.

[9]. Yucheng Zhang. A Fast Asymmetric Extremum Content Defined Chunking Algorithm for Data De-duplication in Backup Storage Systems. In: 2015 IEEE Conference on Computer Communications (INFOCOM)

[10]. A. Venish,K. Siva Sankar. Study of Chunking Algorithm in Data De-duplication. In: Advances in Intelligent Systems and Computing 2016, 13-20.

[11]. Erik Kruus, Christian Ungureanu, Cezary Dubnicki. Bimodal Content Defined Chunking for Backup Streams. In: Fast 10 Preceding of the 8th USENIX Conference on file and Storage Technologies ,USENIX, 2010, pp. 239-252.

[12]. Jiansheng Wei, Junhua Zhu, Yong Li. Multimodal Content Defined Chunking for Data Deduplication. avaliable at : https://www.researchgate.net/publication/261286019, Research gate, 2014.

[13]. A. Bhalerao and A. Pawar, "A survey: On data deduplication for efficiently utilizing cloud storage for big data backups," 2017 International Conference on Trends in Electronics and Informatics (ICEI), Tirunelveli, 2017, pp. 933-938.
doi: 10.1109/ICOEI.2017.8300844

[14]. Marcel, "Performance Evaluation of NFS-based Primary Storage with Deduplication using Windows Server and RAM-based Cache on Small-scale VMware Environment," 2018 International Seminar on Intelligent Technology and Its Applications (ISITIA), Bali, Indonesia, 2018, pp. 227-232. doi: 10.1109/ISITIA.2018.8711330

[15]. Marcel, "Performance Evaluation of NFS-based Primary Storage with Deduplication using Windows Server and RAM-based Cache on Small-scale VMware Environment," 2018 International Seminar on Intelligent Technology and Its Applications (ISITIA), Bali, Indonesia, 2018, pp. 227-232. doi: 10.1109/ISITIA.2018.8711330