A Self-Matching Sliding Block Algorithm Applied to Deduplication in Distributed Storage System

Chuiyi Xie[1,2(✉)], Ying Huo[2], Sihan Qing[3,4,5], Shoushan Luo[1], and Lingli Hu[2]

[1] National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing, China
gdxcy@163.com

[2] Department of Information and Computing Science, Shaoguan University, Shaoguan, China

[3] Institute of Software, Chinese Academy of Sciences, Beijing, China

[4] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

[5] School of Software and Microelectronics, Peking University, Beijing, China

Abstract. The deduplication technology can significantly reduce the amount of storage in data centers, thus to save network bandwidth and decrease the cost of construction and maintenance. Having inspired by the sliding block method of the Sliding Block (SB) algorithm and independent block-dividing thought of the Content Defined Chunking (CDC) algorithm, a Self-Matching Sliding Block (SMSB) algorithm for deduplication is proposed. Via communication with metadata node, the storage system client builds a matching table in local memory that contains fingerprint and checksum, based on the matching table to realize sliding block self-matching so as to detect the duplicate blocks. The experimental results show that the deduplication rate and the disk space utilization rate of SMSB algorithm is respectively 2.03 times and 1.28 times of the CDC algorithm and that the data processing speed is 0.83 times of the CDC algorithm. The SMSB algorithm is suitable for distributed storage system.

Keywords: Distributed storage · Deduplication · Sliding block algorithm · Rabin fingerprint · Adler-32 checksum

1 Introduction

Distributed storage system can store huge amounts of data. Due to its characteristics of high performance-cost ratio and good scalability, it is widely used in cloud storage, disaster recovery, backup and other purposes. But distributed storage system has a lot of duplicate data, which brings a waste of storage space and adds a lot of unnecessary network traffic. So it is quite necessary to employ deduplication technology for distributed storage.

By the difference of detection granularity, deduplication can process in file-level and block-level or byte/bit-level. The smaller detection granularity, the more detected redundant data will be, but the complexity and consume will correspondingly increase. The Content-Defined Chunking (CDC), a classic algorithm for deduplication, employs Sliding Window technology to determine the split point of chunking, has been applied in LBFS, [1] Pastiche backup system [2] and Deep Store archive storage system [3] and the like. The CDC is suitable for frequently updated data set and has more advantages of reducing the occupying of storage space over the Fixed-Sized Block method. [1] However, in the extreme case, if the entire data stream does not find the matching boundary point, the CDC will degenerate to the FSB. To process the blocks beyond a certain length, Eshghi and Tang [4] proposed a Two Thresholds Two Divisors (TTTD) algorithm supporting the size of chunk which bears two expectations and two thresholds. Lu et al. [5] proposed the solution of counting the frequency of occurrences of block to improve the effect of the block and raise the rate of deduplication. Zhang et al. [6] proposed an asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication. Yu et al. [7] present a leap-based CDC algorithm which provides significant improvement in deduplication performance without compromising the deduplication ratio.

The chunks produced by CDC are varied with length; thereby more complexity of management appears. The Sliding Block (SB) algorithm [8] combines the advantages of fixed-size and variable-size, mainly produces fix-size chunks. The SB algorithm is efficient to solve the insert and delete problem. Wang et al. [9] employ backtracking sub-blocks strategy in SB algorithm, via backtracking the left/right 1/4 and 1/2 sub-blocks in matching-failed segments, improves the duplicate detection precision compared with the traditional SB algorithm and CDC algorithm. Zhu et al. [10] designed and implemented a backup system with intelligent data deduplication, which named Backup Ded up including four deduplication strategies, that is SIS, FSB, CDC and SB.

In this paper, a Self-Matching Sliding Block (SMSB) Algorithm applied to deduplication in distributed storage system is proposed. The SMSB creates a matching table in deduplication processing node and processes the self-matching on the fingerprint and checksum of sliding block. It ignores a large number of data blocks which do not meet the matching conditions, so as to improve the efficiency of deduplication. It is suitable for distributed environment.

2 Deduplication Problem in Distributed Storage

Distributed storage system, which is generally consisted of a single management node that can practice unified allocation and management of resources to all storage nodes and a plurality of data storage nodes, may be visited by multiple clients simultaneously. The management node saves data block information, the storage nodes store all data blocks. If need to perform deduplication in the distributed storage system, the task can be assigned to minor storage nodes or data sender. The transfer for duplicate data wastes so much bandwidth; therefore, the processing of eliminating redundant done before the transfer of data is a very appropriate option.

The deduplication rate of the SB algorithm ranks top in all types of the same data detecting method, but this algorithm needs to query checksum for matching the block when the sliding window move one byte forward. Since the metadata of distributed storage system is stored in one or more nodes, and the nodes performing task of matching cannot save the massive checksum and data, they can only complete the matching task by visiting metadata nodes via network. The processing speed is too slow! The CDC algorithm can take advantage of the internal characteristics of the data for deduplication without having to query the metadata node in the sub-block process, but compared with the SB algorithm, this algorithm has lower deduplication rate and disk space utilization.

3 SMSB Algorithm

In order to solve the deduplication problem in distributed storage, we proposed a novel Self-Matching Sliding Block (SMSB) algorithm. The SMSB uses a matching table as repetitive testing cache, and each node performing deduplication updates timely matching table data by querying metadata node in the process.

3.1 Description

The process of SMSB algorithm is shown in Fig. 1. Firstly, calculating the fingerprint and checksum of the data block in sliding window, if successfully matched, it shows that the block is likely to be repeated; and then calculate the hash value for further examination. If the match is not successful, it is probably not repeated block, and then the sliding window continues to move forward. When the sliding window has been moved to a block size distance, the block in it matching no blocks is likely a new one. The hash value of the block is calculated and stored, and later used as a comparison target block.

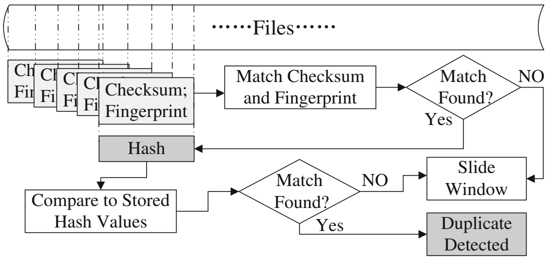

Fig. 1. Self-Matching Sliding Block method

3.2 Matching Table

The matching table consists of fingerprint and checksum. For a continuous string [X_1, X_2,…, X_W, X_{W+1}, X_{W+2},…] (each character has 8 bits), any X_i represents polynomial $X_i(t)$ Select a irreducible polynomial P(t) on Galois field GF(2^n), the Rabin fingerprint [11] $R_1(t)$ and Adler-32 checksum [12] S_1 of string [X_1, X_2, …, X_W] are calculated as follows:

$$R_1(t) = \sum\nolimits_{j=1}^{W} X_j(t)t^{8(W-j)} \bmod \ P(t) \tag{1}$$

$$\begin{cases} A_1 = 1 + \sum_{j=1}^{W} X_j \bmod 65521 \\ B_1 = \sum_{j=1}^{W} \left[(W - j + 1) \times X_j\right] + W \bmod 65521 \\ S_1 = B_i \ll 16 + A_i \end{cases} \tag{2}$$

For a string [X_{1+1}, X_{1+2},…, X_W, X_{W+1}]($i \geq 1$), its fingerprint $R_{i+1}(t)$ and checksum S_{i+1} are calculated as follows:

$$R_{i+1}(t) = R_i(t)t^8 - X_1(t)t^{8w} + X_{1+w}(t) \quad modP(t) \tag{3}$$

$$\begin{cases} A_{i+1} = A_i - X_i + X_{i+W} \bmod 65521 \\ B_{i+1} = B_i + A_{i+1} - WX_{i-1} \ \bmod 65521 \\ S_{i+1} = B_{i+1} \ll 16 + A_{i+1} \end{cases} \tag{4}$$

Equations (3) and (4) have the computation time complexity of $O(1)$.

3.3 Create and Maintain Matching Table

A matching table is created in memory, in order to achieve fast matching. Matching table is actually a simple index table stored in memory as an array, using the fingerprint of the data block as the array subscript (index), checksum as the value of the array element (index results). Matching table is updated by the latest block-priority strategy. When storing a new data block, according to the Rabin fingerprint(*rabin*) and Adler-32 checksum(*adler*) of the block, update Match[*rabin*] as *adler*.

3.4 Self-matching

Assume that storage space has processed a certain number of data blocks, in the matching table established based on this, checksum and fingerprint has a corresponding relationship as follows: (R1, S1), (R2, S2), (R3, S3), (R4, S4). When the sliding window moves forward, the fingerprint and checksum of each block corresponding to it will be calculated, and as a matching pair. The course of their work is shown in Fig. 2.

found..

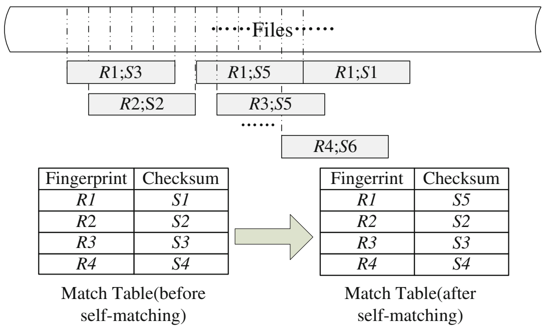

Fig. 2. Self-matching process

4 Experimental Results

The experiments use the actual data of the ordinary computer to extract three types of files (Word documents, source code, and BMP images) in the hard disk, less duplicate or identical files. The collected data sets are listed in Table 1.

In the experiments, the data sets on Table 1 were partitioned by using 64, 256, 512, 1024, 4096 bytes as a unit block. In CDC algorithm, the minimum block size is set to be a half of the maximum value.

Table 1. Testing data set

File type	Quantity of documents	Size (Bytes)
Office document	57	21,978 K
Source code	368	4,769 K
BMP image	12	60,258 K

4.1 Deduplication Rate Test

The deduplication rates and comparison results are shown in Fig. 3.

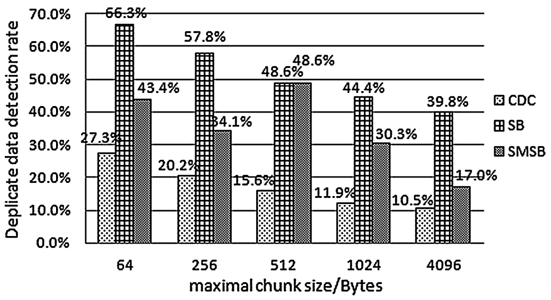

Fig. 3. Compare with different chunk size

4.2 Disk Space Utilization Test

The rates of disk space utilization and comparison results are shown in Fig. 4.

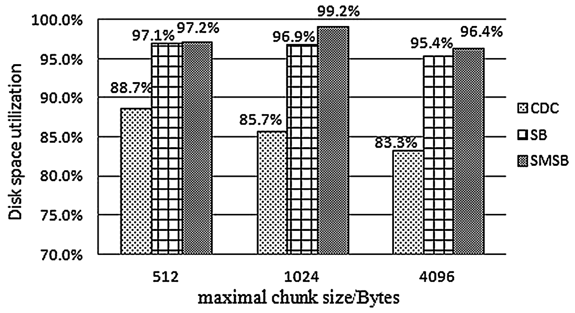

Fig. 4. Compare with different chunk size

4.3 The Data Processing Rate Test

The data processing rates and comparison results are shown in Fig. 5.

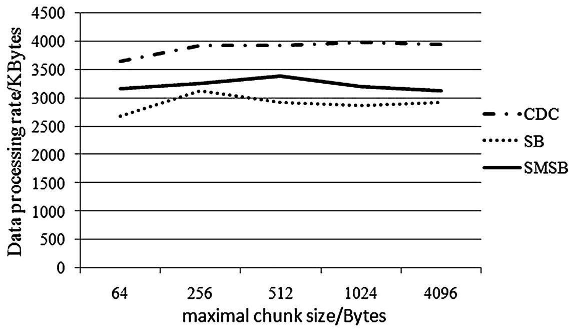

Fig. 5. Compare with different chunk size

4.4 Summary of Results

The comparison results of three algorithms by each evaluation criteria are summarized in Table 2. SMSB algorithm is better than the CDC algorithm in deduplication rate, disk space utilization, but it is lower in the data processing rate. The deduplication rate and the disk space utilization rate of SMSB algorithm is respectively 2.03 times and 1.28 times of the CDC algorithm and that the data processing speed is 0.83 times of the CDC algorithm. We are confident that the SMSB algorithm is more suitable than the CDC algorithm when used in a distributed storage system.

Table 2. Self-Matching Sliding Block

Evaluation criteria	Results
Deduplication rate	CDC < SMSB SB
Disk space utilization	CDC < SB SMSB
Data processing rate	SB < SMSB < CDC

5 Conclusion

Due to the limitations of the experimental conditions, the program has only been tested on small data sets, in the actual environment, the distributed storage system data block size will be bigger, generally between 4 KB to 128 KB, the proportion of duplicate data in the system will also be larger. The main purpose of this paper is to compare the processing performance of different algorithms, so as to find better deduplication method. Therefore, the work in this paper has a practical guiding significance.

For the distributed storage system with the feature of the Locality Preserved Caching (LPC), such as disaster recovery system and the backup system, we can modify the maintenance rules of the matching table according to this feature thus to identify the successor of duplicate blocks in the storage system, thus the coming block will be able to be predicted. We are ready to further improve and enhance the performance of SMSB algorithm for disaster recovery, backup storage system.

Acknowledgment. This study is supported by National Natural Science Foundation of China (61170282), Guangdong Laboratory Research Foundation (GDJ2014081), Shaoguan Innovation Foundation (2012CX/K123), Scientific Research Project of Shaoguan University (201216), Discipline Construction Project of Guangdong Province (2013KJCX0168), and Guangdong Natural Science Foundation (2014A030307029).

References

1. Muthitacharoen, A., Chen, B., Mazieres, D.: A low-bandwidth network file system. Proc. ACM Symp. Oper. Syst. Principles **35**(5), 174–187 (2001)
2. Cox, L.P., Murray, C.D., Noble, B.D.: Pastiche: making backup cheap and easy. ACM SIGOPS Oper. Syst. Rev. **36**, 285–298 (2002)
3. You, L.L., Pollack, K.T., Long, D.D.E.: Deep store: an archival storage system architecture. In: 2014 IEEE 30th International Conference on Data Engineering, pp. 804–815. IEEE Press, New York (2005)
4. Eshghi, K., Tang, H.K.: A framework for analyzing and improving content-based chunking algorithms. Technical report, Hewlett-Packard Labs (2005)
5. Lu, G.L., Jin, Y., Du, H.C.: Frequency based chunking for data de-duplication. In: Modeling Analysis & Simulation of Computer & Telecommunication Systems, pp.287–296. IEEE Press, New York (2010)
6. Zhang, Y.C., Jiang, H., Feng, D., Xia, W., Fu, M., Huang, F.T., Zhou, Y.K.: AE: an asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp.1337–1345. IEEE Press, Kowloon (2015)

7. Yu, C., Zhang, C., Mao Y., Li, F.L.: Leap-based content defined chunking — theory and implementation. In: 2015 31st Symposium on Mass Storage Systems and Technologies (MSST), pp.1–12. IEEE Press, Santa Clara (2015)
8. Bobbarjung, D.R., Jagannathan, S., Dubnicki, C.: Improving duplicate elimination in storage systems. ACM Trans. Storage **2**(4), 424–448 (2006)
9. Wang, G.P., Chen, S.Y., Lin, M.W., Liu, X.W.: SBBS: a sliding blocking algorithm with backtracking sub-blocks for duplicate data detection. Expert Syst. Appl. **41**, 2415–2423 (2014)
10. Zhu, G.F., Zhang X.J., Wang, L., Zhu, Y.G., Dong, X.S.: An intelligent data deduplication based backup system. In: 2012 15th International Conference on Network-Based Information Systems (NBiS), pp. 771–776. IEEE Press, New York (2012)
11. Rabin, M.: Fingerprint by random polynomials. Technical Report, Center for Research in Computing Technology, Harvard University (1981)
12. Deutsch, L.P., Gailly, J.L.: RFC 1950: ZLIB compressed data format specification version 3. In: RFC 1950, Aladdin Enterprises, Info-ZIP (1996)