Journal of computer science and software applications
https://www.mfacademia.org/
ISSN:2377-0430
Vol. 4, No. 2, 2024

Enhancing Data Recovery in Deduplication Backup Systems: A Novel Rewriting Algorithm to Minimize Fragmentation Effects

Edward Richardson
Shenandoah University
edwardri@sshenandoah.edu

Abstract:

Deduplication technology detects and eliminates redundant data, significantly conserving disk space, thus finding extensive applications across various domains. Nevertheless, the removal of numerous duplicate data blocks leads to data fragmentation, adversely affecting data recovery performance. To address fragmentation issues, rewriting algorithms have been introduced. However, current rewriting algorithms fail to precisely identify fragmented blocks. To resolve this issue, we propose a new rewriting algorithm called MERW. MERW takes into account fragmentation information on a broader scale and selects the most optimal fragmented blocks. Consequently, MERW effectively mitigates data fragmentation by accurately rewriting fragmented blocks. Experimental results demonstrate that MERW enhances recovery performance by 48% and improves the deduplication ratio by 6%.

Keywords:

Data Deduplication; Fragmentation; Rewriting; Recovery; Backup.

1. Introduction

The explosive growth of information data poses severe challenges to storage technology. There are many duplicate data in massive data, and storing duplicate data causes a waste of storage space. The deduplication technology uses a block algorithm to block the data stream [1,2], and saves only the unique copy of the duplicate data block, thereby greatly saving storage space. Data fragmentation refers to the continuous data blocks in the data stream are scattered in various locations on the disk after deduplication. We refer to the sequence of data blocks in the data stream as a logical sequence and the sequence of data blocks stored on the disk as a physical sequence. The rewrite algorithms [6, 7] are to make the logical sequence and the physical sequence as consistent as possible by rewriting the fragment blocks, thereby alleviating the fragmentation problem and improving the data recovery performance. However, rewriting fragmented blocks will reduce the data deduplication ratio [3]. The deduplication ratio refers to the size of the deleted data compared to the size of the original backup data. Precise deduplication methods [4] (does not apply any rewriting algorithm) delete every duplicate block to maximize the deduplication ratio, but the data recovery performance of the deduplication method is extremely poor. With the development of cloud storage, the index of data recovery performance becomes more and more important [5]. In this regard, the rewriting algorithms [6,7] identify fragmented blocks in repeated blocks, and rewrite these fragmented blocks, sacrificing a certain deduplication ratio in exchange for an improvement in data recovery performance. However, these rewrite algorithms use the local information of the backup stream (data stream) to identify the fragmented blocks, resulting in the identified fragmented blocks being not accurate enough. In this regard, we propose the MERW algorithm. The MERW algorithm uses as much information as

possible in the backup stream to more accurately identify and rewrite fragmented blocks, which ultimately improves not only the performance of data recovery but also the deduplication ratio.

2. MERW

2.1 System Overview

MERW is divided into three major modules: Information Recording Module (IRM), Optimal Selection Module (OSM), and Rewriting Module (RM). The information recording module is responsible for recording the fragmentation information of the data block; the optimal selection module is responsible for selecting the most fragmented fragment from the fragmentation information; the rewriting module is responsible for rewriting the fragment selected by the optimal selection module. The work flow of the three modules is as follows: (i) Each time a new data block is backed up, the information recording module records the fragmentation degree of the data block and sorts the data blocks according to the fragmentation degree; (ii) when rewriting is needed When fragmented blocks are used to improve data recovery performance, the optimal selection operation is triggered. At this time, the optimal selection module selects the top m most severely fragmented data blocks in the sorting table; (iii) the rewrite module divides these fragmented blocks Rewrite to disk.

In the (i) step, the degree of fragmentation of a data block refers to the time cost of reading the data block during data recovery. The longer the read time, the more severe the fragmentation of the data block; In (ii), m can be adjusted. The larger m is, the lower the deduplication ratio is, and the better the data recovery performance is. The smaller m is, the higher the deduplication ratio is, the worse the data recovery performance is. By dynamically adjusting the value of m, NERW can very well weigh the relationship between the deduplication ratio and the data recovery performance. In (iii), the operation of writing fragmented blocks to disk is called rewriting, because a fragmented block must be A duplicate block, where writing a duplicate block to disk is called overwrite, and writing a unique block to disk is called write.

2.2 Rewriting the optimal fragmented blocks

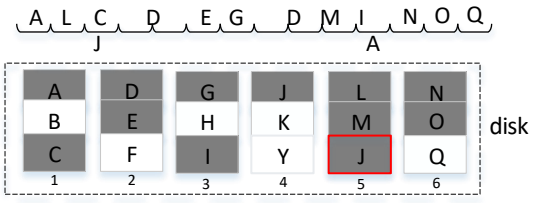


Fig.1. Rewriting the optimal fragmented blocks.

As shown in Figure 1, after the nth backup stream is deduplicated, the data blocks in the backup stream are scattered and stored in various containers [8] on the disk (that is, the gray data blocks in the figure). A container is a unit that stores several data blocks, and it is used to save the locality of the backup stream. The capacity of each container in the figure is 3 data blocks, and the white data blocks represent data blocks in other backup streams. In the process of restoring the n-th backup stream, reading containers 1, 2, and 3 can recover two data blocks, but reading container 4 can only recover data block J. The number of data blocks that can be recovered by reading a container can well reflect the degree of fragmentation. We define the degree of fragmentation as:

$$\text{Degree} = \text{Capacity}/\text{numb} \quad (1)$$

In the formula, degree represents the degree of fragmentation, numb represents the gray data blocks in a container, and the Capacity table indicates the maximum number of data blocks that a container can store. Then for data blocks A and C, they are fragmented to 1.5. Similarly, for data blocks D and E, and data blocks G and I, they are all fragmented to 1.5; but for data block J, it is fragmented. The

degree is 3. The rewrite algorithm selects the most fragmented data blocks and rewrites them. As shown, the most fragmented data block J is rewritten into the container 5.

By rewriting the fragmented block, when restoring the data block J in the nth backup stream, the data block can be directly obtained from the container 5 in the memory, because when the previous data block L is restored, the system has removed the container 5 from the disk. Read into memory. On the contrary, if the data block J is not rewritten into the container 5, the container 4 needs to be read from the disk to obtain the data block J, which results in an additional disk IO, which reduces the data recovery performance.By rewriting the fragmented block, when restoring the data block J in the nth backup stream, the data block can be directly obtained from the container 5 in the memory, because when the previous data block L is restored, the system has removed the container 5 from the disk. Read into memory. On the contrary, if the data block J is not rewritten into the container 5, the container 4 needs to be read from the disk to obtain the data block J, which results in an additional disk IO, which reduces the data recovery performance.

3. Evaluation

3.1 Experiment environment

To evaluate the efficiency of the MERW, we implemented the MERW on Destor [9]. The Destor is an open source project on a deduplication backup system. The None method [3, 9] does not use any rewrite algorithm to alleviate fragmentation, so we use it as a benchmark for our experiments; the CRW method [10] coarsely identifies and rewrites fragmented blocks, sacrificing the deduplication ratio in exchange for data recovery performance improvement However, because the fragments it recognizes are not accurate enough, its gain is very limited. Fortunately, MERW accurately identifies and rewrites fragments, which makes up for the shortcomings of CRW, and further improves the data recovery performance and deduplication ratio. We conducted comprehensive experiments and comparisons of None, CRW and MERW to prove the efficiency of MERW. The hardware configuration of the experiment consists of a quad-core CPU (Intel (R) Core (TM) i7-6700 CPU @ 3.40GHz), 4G memory and 1000G hard disk. The operating system used in the experiment is CentOS release 7.4 (Linux version 3.10. 0).

3.2 Workload

Table 1 The dataset characteristics

Attributes	Value
Total size	1487GB
Average chunk size	4KB
# of versions	15
Average size of one backup	99GB

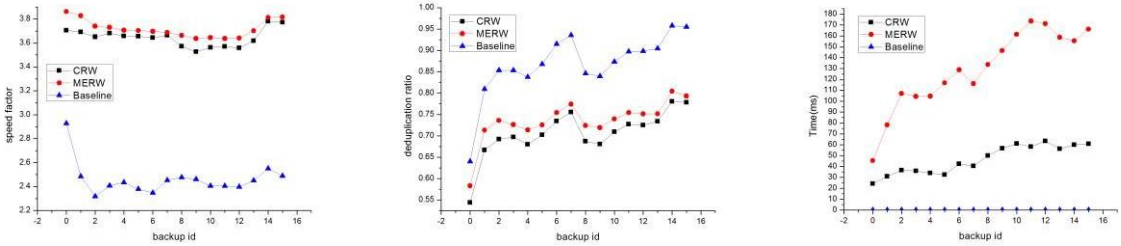
We use a representative subset of the FSL [11] dataset to deploy our experiments. FSL is a commonly used dataset [12]. This dataset is collected on the server of Stony Brook University and represents the real deduplication backup scenario. Data. We analyzed the characteristics of the dataset, and Table 1 lists some important characteristics of the dataset. We performed a total of 15 backups, the average data size of each backup was 99GB, the total size of the data set used was 1487GB, we used the content-based variable-length block algorithm [2], and the average block size was 4KB.

3.3 Performance Evaluation

We measure three performance indicators: data recovery performance, deduplication ratio, and computational overhead. The speed factor [8] reflects the performance of data recovery well. It represents the number of data blocks that can be recovered by reading a container. The larger the value of the speed factor, the better the recovery performance. The deduplication ratio refers to the deletion. The data size is divided by the original data size. The higher the deduplication ratio, the more storage space can be saved. The calculation overhead refers to the time it takes for the system

to recognize fragmented blocks. The lower the calculation overhead, the better the performance.

Restore performance: Fig.2 (a) shows the speed factor of the three methods, where the abscissa is the backup id (backup version number) and the ordinate is the speed factor. As shown in the figure, Baseline has the lowest speed factor. This is because Baseline does not use any rewrite algorithm to rewrite fragmented blocks. As a result, during the data recovery process, the system frequently accesses the disk, and frequent disk IO will reduce data recovery performance. (Ie speed factor). The speed factor of CRW is much higher than the speed factor of Baseline. This is because CRW coarsely identifies and rewrites fragmented blocks, which alleviates the degree of fragmentation, so its speed factor has been greatly improved. As we expected, the speed factor of MERW is the highest, because MERW more accurately recognizes fragmentation than CRW, and rewrites the optimal fragmented block, thereby further improving data recovery performance. The average speed factors for MERW, Baseline, and CRW are 3.7, 2.5, and 3.5, respectively.



(a) restore performance (b) deduplication ratio (c) computing overhead

Fig. 2 The restore performance, deduplication ratio and computing overhead, driven by the real workloads

Deduplication ratio: Fig.2 (b) shows the deduplication ratio of the three methods, where the abscissa is the backup id and the ordinate is the deduplication ratio. As shown in the figure, CRW has the lowest deduplication ratio. This is because CRW is not accurate enough to identify fragmented blocks, resulting in rewriting some unnecessary data blocks, thereby wasting disk space. By rewriting the optimal fragmented block (the most fragmented fragmented block), MERW avoids rewriting unnecessary data blocks, thereby improving the deduplication ratio. Baseline's deduplication ratio is the highest, because baseline does not consider the performance of data recovery, it only writes the unique block to disk, and does not rewrite fragmented blocks. The average deduplication ratios of NERW, CRW, and Baseline were 0.75, 0.71, and 0.87, respectively.

Computing overhead: Fig.2 (c) shows the computational cost of the three methods. The abscissa is the backup id, and the ordinate is time. The unit is millisecond. As we expected, Baseline's computational overhead is 0, because it does not recognize the operation of fragmented blocks at all. The computational cost of CRW is higher than that of Baseline, because CRW requires additional calculations to identify fragmented blocks. It is expected that the computational cost of MERW is higher than CRW, because MERW needs more calculations in order to identify the optimal fragment. However, even though the calculation overhead of MERW is high in the three methods, the calculation overhead required by MERW is still small. The highest calculation overhead of MERW in all backups is only 176ms (backup id11), which can be ignored.

Baseline severely degrades the performance of data recovery. In response to this problem, CRW improves the data recovery performance by coarsely identifying and rewriting fragmented blocks, but also reduces a certain deduplication ratio. CRW's identification of fragmented blocks is not accurate enough, resulting in rewriting unnecessary data blocks, not only wasting storage space (such as disks), but also limiting the performance of data recovery. In response to this problem, we proposed MERW, which accurately identified fragmented blocks and rewritten the optimal fragmented blocks, which ultimately improved not only the performance of data recovery, but also the deduplication ratio.

4. Conclusion

In disaster recovery systems, data recovery performance is especially important. The fragment blocks identified by the current rewrite methods are not accurate enough. The rewritten fragment blocks are not optimal fragment blocks, so their improvement of data recovery performance and deduplication ratio are limited. In response to these problems, in this pepar, we propose MERW. MERW selects the optimal fragmented blocks through a large number of calculations, and rewrites these optimal fragmented blocks, which makes up for the shortcomings of traditional methods of rewriting unnecessary fragmented blocks, thereby not only improving data recovery performance, but also improving the deduplication ratio. We conducted a comprehensive experiments. Experimental results show that, compared to Baseline and CRW, MERW improves the data recovery performance by 48% and 5.7%, respectively. Compared to CRW, MERW improves the deduplication ratio by 6%.

References

[1] Quinlan S, Dorward S. Venti: A New Approach to Archival Storage[C]//FAST. 2002, 2: 89-101.

[2] Rabin M O. Fingerprinting by random polynomials[J]. Technical report, 1981.

[3] Zhu B, Li K, Patterson R H. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System[C]//Fast. 2008, 8: 1-14.

[4] Meister D, Kaiser J. Block locality caching for data deduplication[C]//Proceedings of the 6th International Systems and Storage Conference. 2013: 1-12.

[5] Cao Z, Liu S, Wu F, et al. Sliding look-back window assisted data chunk rewriting for improving deduplication restore performance[C]//17th {USENIX} Conference on File and Storage Technologies ({FAST} 19). 2019: 129-142.

[6] Nam Y J, Park D, Du D H C. Assuring demanded read performance of data deduplication storage with backup datasets[C]//2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE, 2012: 201-208.

[7] Lillibridge M, Eshghi K, Bhagwat D. Improving restore speed for backup systems that use inline chunk-based deduplication[C]//Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13). 2013: 183-197.

[8] Fu M, Feng D, Hua Y, et al. Accelerating restore and garbage collection in deduplication-based backup systems via exploiting historical information[C]//2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14). 2014: 181-192.

[9] Fu M, Feng D, Hua Y, et al. Design tradeoffs for data deduplication performance in backup workloads[C]//13th {USENIX} Conference on File and Storage Technologies ({FAST} 15). 2015: 331-344.

[10] Lillibridge M, Eshghi K, Bhagwat D. Improving restore speed for backup systems that use inline chunk-based deduplication[C]//Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13). 2013: 183-197.

[11] http://tracer.filesystems.org/..

[12] Tarasov V, Mudrankit A, Buik W, et al. Generating realistic datasets for deduplication analysis[C]//Presented as part of the 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12). 2012: 261-272.