e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 01
January 2018

Reducing Fragmentation for In-line De-duplication Backup Storage via Exploiting Backup History and Cache Knowledge

1Ch.Shiva Kumar[1], Ragi Rajesh[2] & Devarakonda Krishna[3]
[1]ASSISTANT PROFESSOR, CSE DEPARTMENT, ST.MARTINS ENGINEERING COLLEGE, JNTUH
[2]ASSISTANT PROFESSOR, CSE DEPARTMENT, ST.MARTINS ENGINEERING COLLEGE, JNTUH
[3]ASSISTANT PROFESSOR, CSE DEPARTMENT, ST.MARTINS ENGINEERING COLLEGE, JNTUH

Abstract:

In backup systems, the chunks of each backup are physically scattered after de-duplication, which causes a challenging fragmentation problem. We observe that the fragmentation comes into sparse and out-of-order containers. The sparse container decreases restore performance and garbage collection efficiency, while the out-of-order container decreases restore performance if the restore cache is small. In order to reduce the fragmentation, we propose History-Aware Rewriting algorithm (HAR) and Cache-Aware Filter (CAF). HAR exploits historical information in backup systems to accurately identify and reduce sparse containers, and CAF exploits restore cache knowledge to identify the out-of-order containers that hurt restore performance. CAF efficiently complements HAR in datasets where out-of-order containers are dominant. To reduce the metadata overhead of the garbage collection, we further propose a Container-Marker Algorithm (CMA) to identify valid containers instead of valid chunks. Our extensive experimental results from real-world datasets show HAR significantly improves the restore performance by 2.84-175.36at a cost of only rewriting 0.5-2.03% data..

Keywords

De-Duplication , HAR Algorithm, CAF .

1. Introduction

The present challenge in backup storage infrastructure is the management we need to handle the ever increasing volume of data. To face known challenge and to convert management scalable, de-duplication technique is very well known and new techniques and research are being done to make this technique more optimized for future use. Data de-duplication is a special technique used in data compression. These work by eliminating the duplicate copy in storage. Hence it helps in improving the total experience and utilization of the data storage in dataset. Also help in managing the data transfer via network where the amount of data transfer will get reduced significantly. In de-duplication we keep only one copy of data and eliminated redundant data and refer other data which are redundant to the same copy original copy. De-duplication can happen either in the file level system or can happen in the block level. In file-level it removes duplicate copies of the files which are same. The fragmentation is classified into two categories: sparse containers and out-of-order the former reduces restore performance, which might be self-addressed by increasing the size of cache used for restoration. The latter reduces each restore performance and garbage collection, and that we need a editing rule that's capable of accurately distinguishing sparse containers. So as to accurately establish and reduce sparse containers, we have a tendency to observe that sparse containers stay sparse in next backup, and hence propose HAR..HAR considerably improves restore performance with a small decrease of de-duplication ratio. We have a tendency to develop CAF to take advantage of cache information to spot the out-of order containers that might hurt restore performance. CAF is employed within the hybrid theme to boost restore performance underneath restricted restore cache while not a major decrease of de-duplication magnitude relation.

So as to scale back the data overhead of the garbage collection we have a tendency to propose CMA that identifies valid containers rather than valid

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 01
January 2018

chunks within the garbage collection. De-duplication also can also happen at the block level, thus eliminate duplicate set of blocks of knowledge that occur in non-identical files. Although the data de-duplication methods brings lots of advantages to user along with security and privacy issues. Still the users' sensitive data can be in danger from insider and outsider attacks. Ancient encoding, while providing the knowledge confidentiality to the user, is incompatible with data de-duplication. Specifically, ancient encoding needs completely different set of users to cipher their own data of which they have their own keys. Thus, identical dataset copies of completely of various users can result in different cipher texts, creating de-duplication not.

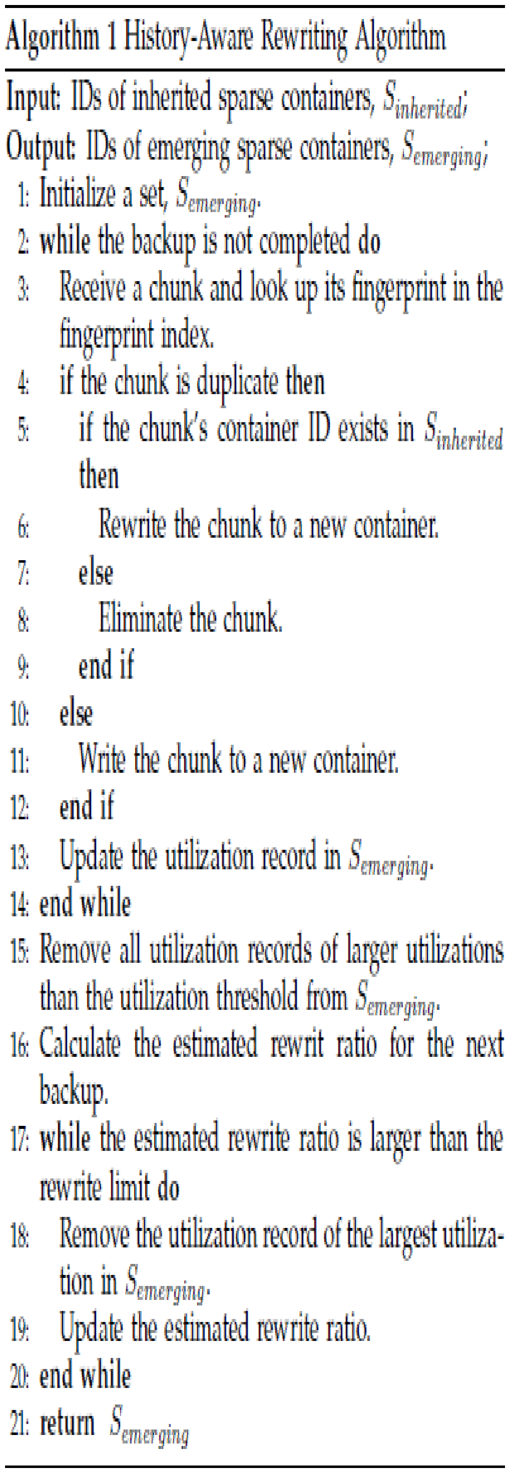
2. Literature Survey

Disk-based de-duplication storage has emerged as the new-generation storage system for enterprise data protection to replace tape libraries. De-duplication removes redundant data segments to compress data into a highly compact form and makes it economical to store backups on disk instead of tape. A crucial requirement for enterprise data protection is high throughput, typically over 100 MB/sec, which enables backups to complete quickly. A significant challenge is to identify and eliminate duplicate data segments at this rate on a low-cost system that cannot afford enough RAM to store an index of the stored segments and may be forced to access an on-disk index for every input segment. This paper describes three techniques employed in the production Data Domain de-duplication file system to relieve the disk bottleneck. These techniques include: (1) the Summary Vector, a compact in-memory data structure for identifying new segments; (2) Stream-Informed Segment Layout, a data layout method to improve on-disk locality for sequentially accessed segments; and (3) Locality Preserved Caching, which maintains the locality of the fingerprints of duplicate segments to achieve high cache hit ratios. Together, they can remove 99% of the disk accesses for de-duplication of real world workloads. These techniques enable a modern two-socket dual-core system to run at 90% CPU utilization with only one shelf of 15 disks and achieve 100 MB/sec for single-stream throughput and 210 MB/sec for multi-stream throughput.
This paper presents a set of techniques to substantially reduce disk I/Os in high-throughput deduplication storage systems. Our experiments show that the combination of these techniques can achieve over 210 MB/sec for 4 multiple write data streams and over 140 MB/sec for 4 read data streams on storage server with two dual-core processors and one shelf of 15 drives. We have shown that Summary Vector can reduce disk index lookups by about 17% and Locality Preserved Caching can reduce disk index lookups by over 80%, but the combined caching techniques can reduce disk index lookups by about 99%. Stream-Informed Segment Layout is an effective abstraction to preserve spatial locality and enable Locality Preserved Caching. These techniques are general methods to improve throughput performance of de-duplication storage systems. Our techniques for minimizing disk I/Os to achieve good de-duplication performance match well against the industry trend of building many-core processors. With quad-core CPU's already available, and eight-core CPU's just around the corner, it will be a relatively short time before a large-scale de-duplication storage system shows up with 400 ~ 800 MB/sec throughput with a modest amount of physical memory.

3. Proposed System

We propose History-Aware Rewriting algorithm (HAR) and Cache-Aware Filter (CAF). HAR exploits historical information in backup systems to accurately identify and reduce sparse containers, and CAF exploits restore cache knowledge to identify the out-of-order containers that hurt restore performance. CAF efficiently complements HAR in datasets where out-of-order containers are dominant. To reduce the metadata overhead of the garbage collection, we further propose a Container-Marker Algorithm (CMA) to identify valid containers instead of valid chunks. Our extensive experimental results from real-world datasets show HAR significantly improves the restore performance by 2.84-175.36at a cost of only rewriting 0.5-2.03% data. Propose a hybrid rewriting algorithm as complements of HAR to reduce the negative impacts of out-of-order containers. HAR, as well as OPT, improves restore performance by 2.84-175.36 at an acceptable cost in de-duplication ratio. HAR outperforms the state-of-the-art work in terms of both de--duplication ratio and restore performance. The hybrid scheme is helpful to further improve restore performance in datasets where out-of-order containers are dominant. To avoid a significant decrease of de-duplication ratio in the hybrid scheme, we develop a Cache-Aware Filter (CAF) to exploit cache knowledge. With the help of CAF, the hybrid scheme significantly improves the de-duplication ratio without decreasing the restore performance. Note that CAF can be used as an optimization of existing rewriting algorithms.

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 01
January 2018



```
Algorithm 1 History-Aware Rewriting Algorithm
Input: IDs of inherited sparse containers, $S_{inherited}$;
Output: IDs of emerging sparse containers, $S_{emerging}$;
 1: Initialize a set, $S_{emerging}$.
 2: while the backup is not completed do
 3:   Receive a chunk and look up its fingerprint in the
      fingerprint index.
 4:   if the chunk is duplicate then
 5:     if the chunk's container ID exists in $S_{inherited}$
        then
 6:       Rewrite the chunk to a new container.
 7:     else
 8:       Eliminate the chunk.
 9:     end if
10:   else
11:     Write the chunk to a new container.
12:   end if
13:   Update the utilization record in $S_{emerging}$.
14: end while
15: Remove all utilization records of larger utilizations
    than the utilization threshold from $S_{emerging}$.
16: Calculate the estimated rewrit ratio for the next
    backup.
17: while the estimated rewrite ratio is larger than the
    rewrite limit do
18:   Remove the utilization record of the largest utiliza-
      tion in $S_{emerging}$.
19:   Update the estimated rewrite ratio.
20: end while
21: return $S_{emerging}$
```

figure 1: HAR Algorithm.

At the beginning of a backup, HAR loads IDs of all inherited sparse containers to construct the in-memory Inherited structure During the backup, HAR rewrites all duplicate chunks whose container IDs exist in inherited. Additionally, HAR maintains an in-memory structure, emerging (included in collected info in Figure 3), to monitor the utilizations of all the containers referenced by the backup. Emerging is a set of utilization records, and each record consists of a container ID and the current utilization of the container. After the backup concludes, HAR removes the records of higher utilizations than the utilization threshold from. Emerging then contains IDs of all emerging sparse containers. In most cases, Emerging can be flushed directly to disks as the Inherited of the next backup, because the size of Emerging is generally small due to our second observation. However, there are two spikes in. A large number of emerging sparse containers indicate that we have many fragmented chunks to be rewritten in next backup. It would change the performance bottleneck to data writing and hurt the backup performance that is of top priority. To address this problem, HAR sets a rewrite limit, such as 5%, to avoid too much rewrites in next backup. HAR uses the rewrite limit to determine whether there are too many sparse containers in Emerging. (1) HAR calculates an estimated rewrite ratio (defined as the size of rewritten data divided by the backup size) for the next backup.

4. Conclusion

There is considerably decrease in restore and garbage collection efficiencies because of fragmentation in de-duplication backup systems. Fragmentation can be categorized in 2 forms as sparse container and out-of order container. Sparse basically tell us about the maximum restore and out-of-order tells us about restore performance. HAR help us to identify and rewrite sparse with the knowledge of history. We also came to the conclusion that an optimal caching scheme which is optimal and hybrid algorithm act as a complementary to HAR for reducing the impact of out-of-order case. HAR and OPT helps to optimize the restore performance in de-duplication ratio. HAR helps to optimize both de-duplication ratio and restore performance. As to reduce de-duplication in hybrid scheme we involved CAF to reduce de-duplication ratio in hybrid scheme. We can adapt CAF for optimizing the rewriting algorithms. Container-Marker Algorithm (CMA) is introduced to identify valid containers instead of valid chunks. CMA is bounded by the number of containers; it is more cost effective than the number of chunks.

5. References

1.*B. Zhu, K. Li, H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system", Proc. USENIX FAST, 2008.*
2.*C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, M. Welnicki, "HYDRAstor: A scalable secondary storage.", Proc. USENIX FAST, 2009.*
3.*A. Muthitacharoen, B. Chen, D. Mazières, "A low-bandwidth network file system", Proc. ACM SOSP, 2001.*
Access at ACM
4. *S. Quinlan, S. Dorward, "Venti: A new approach to archival storage", Proc. USENIX FAST, 2002.*
5. *M. Lillibridge, K. Eshghi, D. Bhagwat, "Improving restore speed for backup systems that use inline chunk-based deduplication", Proc. USENIX FAST, 2013.*

6. Y. Nam, G. Lu, N. Park, W. Xiao, D. H. Du, "Chunk fragmentation level: An effective indicator for read performance degradation in deduplication storage", Proc. IEEE High Performance Comput. Commun., pp. 581-586, 2011.
View Article Full Text: PDF (249KB)
7. Y. J. Nam, D. Park, D. H. Du, "Assuring demanded read performance of data deduplication storage with backup datasets", Proc. IEEE MASCOTS, pp. 201-208, 2012.