An Optimization of Backup Storage using Backup History and Cache Knowledge in Reducing Data Fragmentation for In-line De-duplication in Distributed

Abhijit Goswami[1], Nitin Shivale[2]

[1] *Student, Department of Computer Engineering, JSPM's, BSIOTR, Maharashtra, India*
[2] *Assistant Professor Department of Computer Engineering, JSPM's, BSIOTR, Maharashtra, India*

ABSTRACT

The chunks of data that are generated after the backup are physically distributed after deduplication in backup system, which creates a problem know as fragmentation. Basically fragmentation basically comes into sparse and out-of-order containers. The sparse container adversely affect the performance while restoring the database and garbage collection effectively , while the out-of-order container brings an adverse effect on the performance issue if the restore cache built is small. To overcome this fragmentation problem , we propose a method of History-Aware Rewriting algorithm (HAR) and also Cache-Aware Filter (CAF).HAR will gather the historical information in backup systems to define, identify and reduce sparse containers, and CAF acknowledges restore cache knowledge to find the out-of-order containers that impacts restore performance. CAF supports HAR in datasets where out-of-order containers are prominent. To get rid of metadata of the garbage collection, we exploit Container-Marker Algorithm (CMA) to gather valid containers instead of valid chunks. My output helps to prove how HAR significantly improves the restore performance.

Keyword: *Chunk, HAR, CMA, container, Container..*

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, deduplication technique is very well known and new techniques and research are being done to make this technique more optimized for future use. Data deduplication 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 deduplication we keep only one copy of data and eliminated redundant data and refer other data which are redundant to the same copy original copy. Deduplication can happen either in the file level system or can happen in the block level. In file level t 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 deduplication 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 deduplication 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 chunks within the garbage collection. Deduplication also can also happen at the block level, thus eliminate duplicate set of blocks of knowledge that occur in non-identical files.

.

2. LITERATURE SURVEY

A) **iDedup**: Latency-aware, Inline Data set Deduplication for Primary Storage in systems Deduplication technologies are more and more being deployed to scale back value and increase space efficiency in company knowledge centers. However, previous analysis has not applied deduplication techniques to the path which is requested for latency sensitive, primary workloads. This is often primarily thanks to the additional latency these techniques were introduced. Inherently, deduplication data that is present on the disk causes fragmentation [1] that may increases seek for sequent successive reads of a similar data thus, increasing latency. Additionally, deduplication knowledge needs additional disk IOs to access on-disk deduplication information. During this paper, we have a tendency to propose AN inline deduplication resolution, iDedup [2], for primary workloads, whereas minimizing additional IOs and seek. Our algorithmic rule relies on 2 key insights from real-world workloads: i) abstraction vicinity exists in duplicated primary knowledge; and ii) temporal vicinity[3] exists within the access patterns of duplicated data. Mistreatment the primary insight, we have a tendency to by selection deduplicated solely sequences of disk blocks.

B) Chunk Fragmentation Level: An Effective way to Indicate for Read Performance Degradation in Data deduplication has recently become very well-known thing in most auxiliary storage and even in some primary storage for the capability improvement purpose. Other than its write performance, browse performance of the deduplication[4] storage has been gaining quite importance with a good vary of its deployments. During this paper, we have a tendency to emphasize the importance of browsing in the performance in reconstituting a knowledge stream from its distinctive and shared chunks physically spread over deduplication storage.

C) Optimized Hybrid Inline and Out-of-Line Deduplication for Backup Storage in database Backup storage systems typically take away redundancy across backups via inline deduplication that works by referring duplicate chunks of the newest backup to those of existing backups. Inline deduplication [5] reduces restore performance of the newest backup owing to fragmentation [6], and complicates deletion of terminated backups owing to the sharing of knowledge chunks. Whereas out-of-line deduplication addresses [7] the issues by forward pointing existing duplicate chunks to those of the newest backup, it introduces extra I/Os of writing and removing duplicate chunks. We have a tendency to style and implement RevDedup, associate degree economical hybrid inline present in data set and out of-line deduplication system for backup storage.

2.1 Problem Definition

1) While restoring and garbage collection the efficiency reduces considerably because of data fragmentation in deduplication. The fragmentation are divided broadly in 2 categories: out of-order and sparse container

2) Sparse containers determine the maximum restore performance, while out-of-order containers determine the restore performance under limited restore cache. So an efficient algorithm need for identifies and help in rewriting the containers which are sparse using the exploiting method of historical information & reduce the negative impacts of out-of-order containers.

3) The two main aspects for efficient algorithm are restoring performance and out-of-order. They are basically calculated using sparse container and restore cache.

4) So we need to developed better algorithm for identifies and utilize historical information to developed sparse container and try to overcome out-of-order container

5) Along with that we need to design security aspects data access management for better deduplication feasible. To overcome these issues, we proposed a Secure Authorized Deduplication backup storage with reducing fragmentation via exploiting backup history and cache knowledge using Convergent Encryption.

2.2 Proposed Methodology

2.2.1 Convergent encryption

Convergent secret writing provides information confidentiality in deduplication. A user tries to derive a convergent key from every original information copy and encrypts the data copy with the help of convergent key. Additionally, the user conjointly tag for the data copy, specified the tag are accustomed find duplicates. Here, we try to assume that the correctness of the tag holds property, i.e., if 2information copies area unit constant, then their tags area unit constant. To find repeating copies, the user initial sends the tag to the server aspect to envision if the identical copy has been already keep.

2.2.2 HAR Architecture:

We have pool of container which can provide a storage service on disk. We can use fingerprint indexes for this. The disk is useful for keeping the finger print index, and hot part which is in memory. For writing the chunk we use container buffer which is present inside memory The dataset are assigned with the unique ID, say DS1.The transaction or historical data we have are stored on disk which consists of ID for e.g. DS1 the transactional or historical data are divided into 3 main parts: Sparse container of HAR for inherited IDs, the optimal replacement cache and CMA for container manifest. 2.2.3 History-Aware Rewriting Algorithm Whenever the backup starts, HAR holds the IDs of sparse container which are inherited to build in-memory S inherited structure. Whenever backup start, HAR rewrites all duplicate chunks whose container IDs exist in S inherited.

1). Along with that HAR also maintains a structure in built knows as S emerging to monitor or housekeeping the container referred by backup and maintains their utilization factor.

2) S emerging is used to record the utilization factor and each dataset or record consists of utilization factor of each container.

3) Once the backup is done HAR uses the technique to get the record of higher utilization from S emerging. S emerging has the list of all emerging container which are sparse.

4) S emerging can be directly sent to disk as the size of S emerging is small due to our second observation

5) Hence from the observation we can come to know that if the value of chunks is more than they are rewritten in next backup. It would hamper the performance and cause bottleneck issues.

6) To mitigate this effect HAR set the limit as 5% for rewriting, ties would avoid too much rewrite on future backup. HAR makes use of limit defined for rewriting for segregating too many sparse containers in S emerging

7) HAR helps in estimating the rewrite ratio for the coming backup. Specifically it calculates the size for the chunks which are rewritten for each emerging sparse container.

8) This is done using the help of the utilization factor that is calculated from container size. The rewrite ratio is later calculated as the total of all estimated size dividing by the current backup size, which may be give us approx. value of rewrite ratio for the coming backup or next backup.

9) If the value which is estimated for rewrite ratio crosses the predefined value rewrite limit, HAR helps in removing the S emerging with highest utilization and directly jump to step1

10) Else HAR helps in replacing the IDs which are old inherit sparse container with their IDs of the emerging sparse container present in S emerging. The result generated as S inherited which helps to provide as the S inherited for next backup.

2.2.4 Optimal Restore Cache

To reduce the side effects of out-of-order containers on restore performance, we tend to implement Beladys best replacement cache. Implementing the optimal cache (OPT) has to apprehend the long run access pattern. We will collect such data throughout the backup, since the sequence of reading chunks throughout the restore is simply constant because the sequence of writing them throughout a backup. Once a chunk is processed through either elimination or over-writing its Container ID, its Container ID is understood. We tend to add access record in the information that is collected .Every access record will solely hold a Container ID. Consecutive accesses to the identical container will be incorporated into a record. This part of historical data will be updated to disks sporadically, and so wouldn’t consume a lot of memory The entire sequence of access records will consume hefty memory once out-of-order containers are dominant. Forward every container is accessed fifty times intermittently and also the average utilization is five hundredth, the entire sequence of access records of a one TB stream consumes over a hundred MB of memory. Rather than checking the entire sequence of access records, we can take a help of slide window to look into a fixed-sized a part of the long run sequence, as a near-optimal theme. The memory foot-print of this near optimal theme is thus delimited.

2.2.5 Container-Marker Algorithm:

Existing garbage pickup schemes have faith in merging thin containers to reclaim invalid chunks within the containers. Before merging, they need to spot invalid chunks to work out utilizations of containers, i.e., reference management. Existing reference management approaches area unit inevitably cumbersome owing to the existence of huge amounts of chunks. HAR naturally accelerates expirations of thin containers and therefore the merging is not any longer necessary. Hence, we'd like to not calculate the precise utilization of every container n we tend to copy the Container-Marker algorithmic program (CMA) to with efficiency confirm that containers area unit invalid. CMA assumes users delete backups during a FIFO theme, during which oldest backups area unit deleted initial.

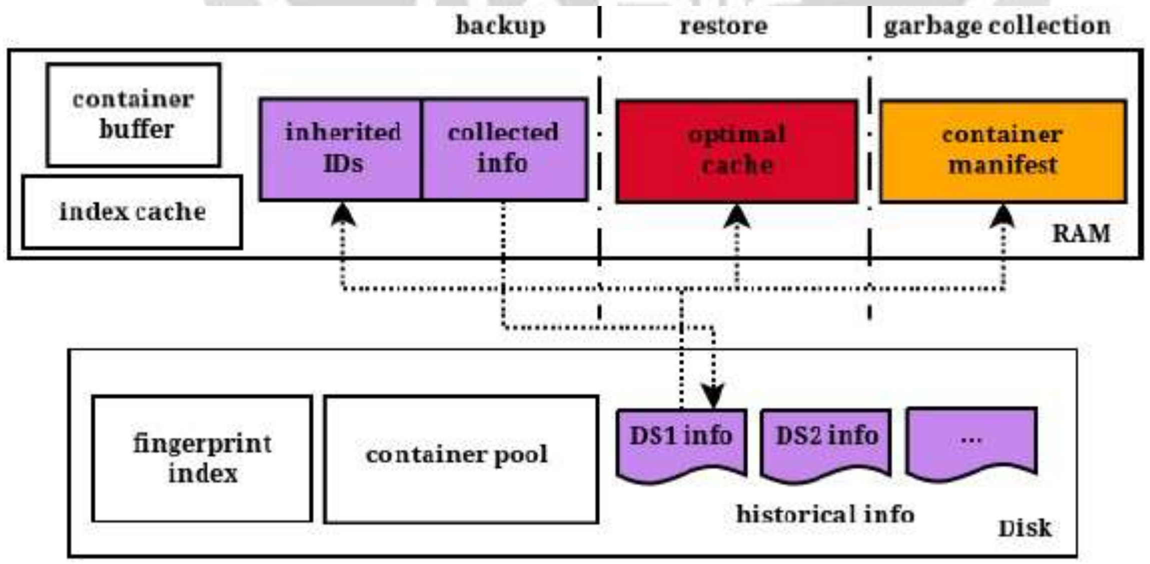
3. SYSTEM ARCHITECTURE

The main purpose of our model is to provide Security to Authorized Deduplication in daily backup and helping it to reduce the fragmentation via exploiting backup history and cache knowledge. Deduplication comes with many issues like security, privacy of user. The security measures taken in deduplication are not good enough. Specially the old measures to protect the data against encryption. The earlier method encrypt with the own key making it impossible to access. A new method knows as Convergent encryption used widely for making deduplication possible. Hence hash value is used to protect the data copy. After the generation of key the user keeps the key and sends the cipher text to backup system. As the key is derived from the main data the duplicate will have the same data which would prove out to be easier to access the data in feasible way. A new protocol is introduced for user data to identify the duplicate data is the copy of main file hence need not upload the same in the server. A convergent key can be used to decrypt the data which is encrypted in the server. This key can be downloaded from the server itself. This encryption can help to prevent the unauthorized access of data in the server for the user

1) User Access: User creates account in the database with his username, password, name, email, mobile no and so on. This data are used as user access management. The user needs to qualify for the data access.

2) File Uploading and its encryption: User need to upload files in the backend. Encryption of file is done using convergent Encryption. Convergent encryption algorithm is used to follow data confidentiality while making Deduplication feasible. It encrypts/decrypts a data set with a convergent key, which is produced by applying the cryptographic hash value of the content of the data copy. After the generation of the key and database encryption, users will have the keys and send the cipher text to the backup storage. Since the encryption operation is produced from the data content, identical data copies will produce the same convergent key and thus we will have the same cipher text.

3) Fragmentation reducing in the system to overcome the Fragmentation, we have introduced History-Aware Rewriting algorithm (HAR) and Cache-Aware Filter (CAF). HAR exploits historical information in backup history systems to mark correctly and reduce sparse containers, and CAF exploits restore cache knowledge to understand the out-of-order containers that will impact restore performance.



SYSTEM ARCHITECTURE

4. ALGORITHM

```
Let S is the Whole System Consist of
History Aware Algorithm
Input: S inherited IDs of inherited sparse container
Output: Semerging IDs of emerging sparse container
1) First you need to initialize Semerging
2) Using while unless the backup is done
3) Need to receive the chunk of the data produced
and lookup for its fingerprinting in the
fingerprint index
4) if the chunk is duplicate
5) if the container ID exists in S inherited then
6) Using hashing technique
7) Rewrite the chunk to a new container
8) else
9) Need to eliminate the chunk data
10) end if
11) end
12) write the chunk to new container
13) end if
14) update the utilization record in Semerging
15) end while
16) Remove all utilization record of larger
utilization
17) calculate the estimated rewrite ratio for the
next backup
18) while the estimated rewrite ratio is larger than
the rewrite limit do
19) remove the utilization record of the largest
utilization in Semerging
20) Update the estimated rewrite ratio
21) end while
22) return Semerging
```

5.MATHEMATICAL MODEL

Let S, be a system such that,
S = fs,M,K,Ci,ni,Sg
Where,
S- Proposed System
s- Initial state at Time T=0.
M - Data copy
K - Convergent key
C - Cipher text
T (M) - tag of data copy
Ci - set of containers
ni - the number of inherited sparse containers
S - Container size
X- Input of System.
-Chunks stream of backup B
Y- Output of System.
-List of Sparse container.
T- Set of steps to be performed from sparse
container detection to garbage collection
CM=fC1,C2,Cig container manifest consist of Ids
of sparse container from previous backup
Take CM as input

```
Receive a new chunk Ci
If Duplicate then
If ID is equal to ant Id in CM=fIdsg then
Re-write the chunk into new container
Else
Eliminate the chunk
End if
Else
Write chunk to new container

End if;
Sid<-update CM
```

Garbage collection by sub-system
Once Container marking is executed successfully
we get Final List of Sparse container
Sid=fc1,c2,,cig
This final list is input to this subsystem
1) The IDs list Present here are of invalid
Containers.
2) The space by invalid containers are reclaimed

7. PERFORMANCE ANALYSIS

System Boundaries: Predefined utilization threshold, such as 50%, the container is considered as a sparse container for the backup. For evaluating the performance of system we consider here two factors average utilization of a backup. The average utilization of a backup exhibits its theoretically maximum restore performance with unlimited restore cache. We calculate the utilization via dividing the backup size by the total size of actually referred containers. Existing system Algorithm shows average utilization for around 10 backup is in between 55 % to 65%. Proposed system Algorithm shows the average utilization for around 10 backups is expected between 80% to 90%.

8. CONCLUSION

There is considerably decrease in restore and garbage collection efficiencies because of fragmentation in deduplication 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 deduplication ratio. HAR helps to optimize both deduplication ratio and restore performance. As to reduce deduplication in hybrid scheme we involved CAF to reduce deduplication ratio in hybrid scheme. We can adapt CAF for optimizing the rewriting algorithms. Container-Marker Algorithm (CMA) sis 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

9. REFERENCES

[1]K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti, “iDedup:Latency-aware, inline data deduplication for primary storage,” in Proc. USENIX FAST, 2012.
[2] Y. Nam, G. Lu, N. Park, W. Xiao, and D. H. Du, “Chunk fragmentation level: An effective indicator for read performance degradation in deduplication storage,” in Proc. IEEE HPCC,2011.
[3] F. Guo and P. Efstathopoulos, “Building a highperformance deduplication system,”in Proc. USENIX ATC, 2011.
[4] J. Wei, H. Jiang, K. Zhou, and D. Feng, “MAD2: A scalable highthroughput exact deduplication approach for network backup services,” in Proc. IEEE MSST, 2010
[5] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, “Sparse indexing: large scale, inline deduplication using sampling and locality,” in Proc. USENIX FAST, 2009.
[6] D. Meister and A. Brinkmann, “dedupv1: Improving deduplication throughput using solid state drives (SSD)” in Proc. IEEE MSST, 2010.
[7] M. Kaczmarczyk, M. Barczynski, W. Kilian, and C. Dubnicki“
Reducing impact of data fragmentation caused by in-line
deduplication,”in Proc. ACM SYSTOR, 2012.