Analysis on Reducing Fragmentation for In-line Deduplication Backup Storage via Exploiting Backup History and Cache Knowledge

Rajashekhar Kothapeta 1 , Assistant Professor
Christu Jyothi Institute of Technology & Science, Jangaon
Rudru Divya Vani 2 , Assistant Professor
Christu Jyothi Institute of Technology & Science, Jangaon

Abstract

In backup systems, the chunks of each backup are physically scattered after deduplication, 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.

I.Introduction

Cloud computing resources (hardware and software) use a network (usually distributed as a service on the Internet). The name schemes come from the general use of the code, such as abstract complex cloud infrastructure. Cloud computing is the longest service assigned to customer data, software and accounts. Cloud computing has Internet resources management services for third parties available for hardware and software. Usually forward software applications and servers access the high-end network computer services.

The economic portfolio does not perform billions of military tens and the research facilities, application-based consumer consumers will provide personal information such as computation cloud computing, or high-performance computing power, with the use of application-based consumer per second, data or large power, computer games storage, attractive

Generally PC technology running servers is a low cost with links to separate data through processing commercial use requirements for large groups of cloud computing networks. It shared the basic information systems with the big pool of IT connected. Often, they use virtualization techniques to boost the power of cloud computing.

II.Existing System

In backup systems, the chunks of each backup are physically scattered after deduplication, 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.

III.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 deduplication ratio. HAR outperforms the state-of-the-art work in terms of both deduplication 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 deduplication 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 deduplication ratio without decreasing the restore performance. Note that CAF can be used as an optimization of existing rewriting algorithms.

System Architecture:

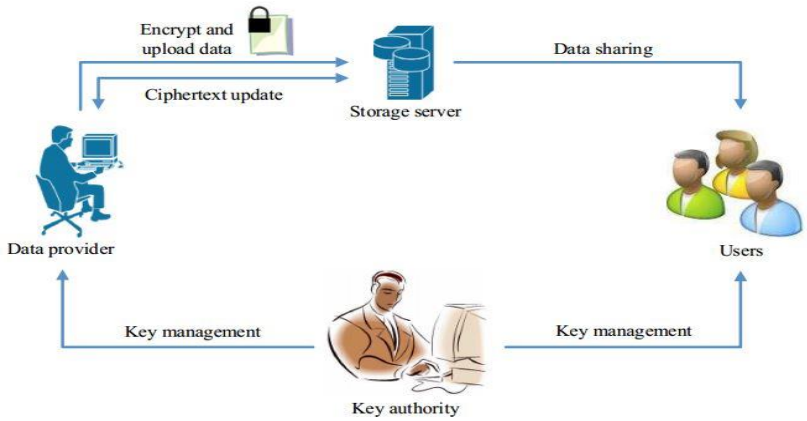

Figure 3.1. System architecture

Encryption, Data AB System to provide secret sharing and Backward mystery. Moreover, the way All data can be shared and encrypted Privacy has proven more. However, it brings a new one Challenges. Then decoding process reencrypt note Involved secret key users Information that public data exchanges, Government is new attacks. Normal, use Secret keys should only be limited to normal Special keys Generator (PKG) Their communications (Alice).

Decryption, it is appropriate to fix the update Use the secret key constantly from time to time. Another challenge comes from the ability. Update Shared text, data encrypted from data provider You often have to carry out the procedures Download-decoder is again encrypted.

We can provide formal definitions for RS-IBE And its corresponding security model; and backward/forward secrecy simultaneously.

We prove that the security of the proposed. Scheme in the standard model, under the decisional ℓ-Bilinear Diffie-Hellman Exponent (ℓ-BDHE) assumption. In addition to security, this system will reduce the time complexity and provide a better performance.

MODULES IMPLEMENTATION

Architecture:

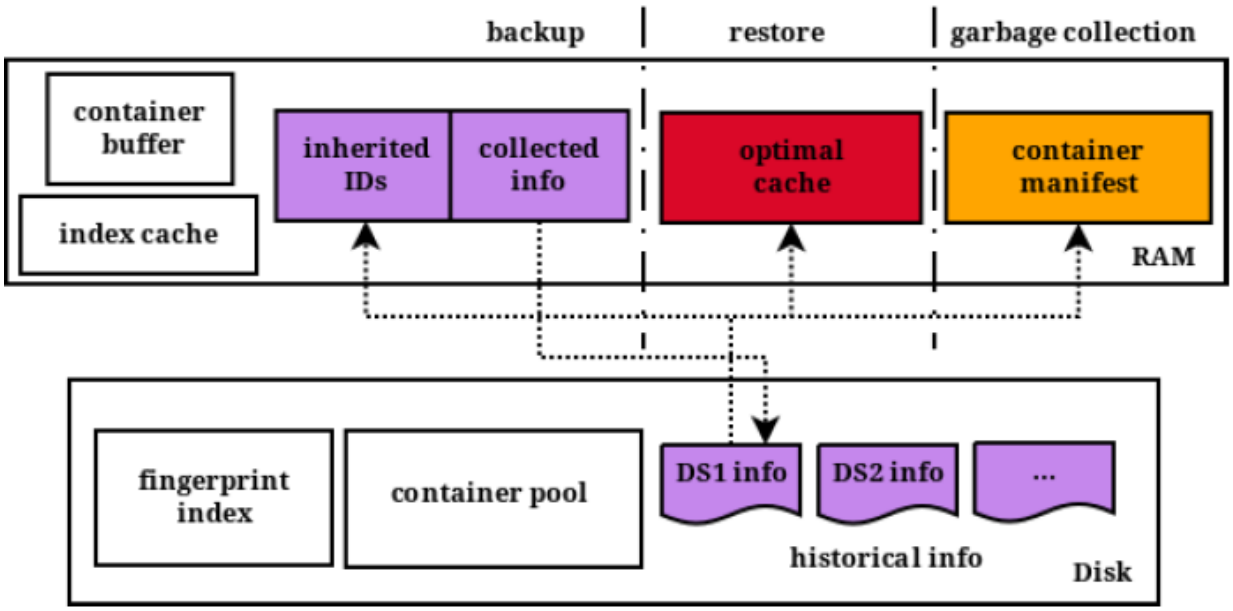

Figure.3.2. Modules Architecture

- **Storage system**

Cloud storage is a model of data storage where the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store end user, organization, or application data

- **Data Deduplication**

Multiple backups, the chunks of a backup unfortunately become physically scattered in different containers, which is known as fragmentation. The negative impacts of the fragmentation are two-fold. First, the fragmentation severely decreases restore performance. The infrequent

restore is important and the main concern from users Moreover, data replication, which is important for disaster recovery, requires reconstructions of original backup streams from deduplication systems and thus suffers from a performance problem similar to the restore operation. Second, the fragmentation results in invalid chunks (not referenced by any backups) becoming physically scattered in different containers when users delete expired backups. Existing garbage collection solutions first identify valid chunks and the containers holding only a few valid chunks (i.e., reference management Then, a merging operation is required to copy the valid chunks in the identified containers to new containers Finally, the identified containers are reclaimed. Unfortunately, the metadata space overhead of reference management is proportional to the number of chunks, and the merging operation is the most timeconsuming phase in garbage collection We observe that the fragmentation comes in two categories of containers: sparse containers and out-of-order containers, which have different negative impacts and require dedicated solutions.

- **Chunk fragmentation**

The fragmentation problem in deduplication systems has received many attentions. iDedup eliminates sequential and duplicate chunks in the context of primary storage systems. Nam et al. propose a quantitative metric to measure the fragmentation level of deduplication systems , and a selective deduplication scheme for backup workloads. SAR stores hot chunks in SSD to accelerate reads. RevDedup employs a hybrid inline and out-of-line deduplication scheme to improve restore performance of latest backups. The Context-Based Rewriting algorithm (CBR) [17] and the capping algorithm (Capping) are recently proposed rewriting algorithms to address the fragmentation problem. Both of them buffer a small part of the on-going backup stream during a backup, and identify fragmented chunks within the buffer (generally 10-20 MB).

For example, Capping divides the backup stream into fixed-sized segments (e.g., 20 MB), and conjectures the fragmentation within each segment. Capping limits the maximum number (say T) of containers a segment can refer to. Suppose a new segment refers to N containers and N > T, the chunks in the N □T containers that hold the least chunks in the segment are rewritten. Reference management for the garbage collection is complicated since each chunk can be referenced by multiple backups. The offline approaches traverse all fingerprints (including the fingerprint index and recipes) when the system is idle. For example, Botelho et al. [14] build a perfect hash vector as a compact representation of all chunks. Since recipes need to occupy significantly large storage space , the traversing operation is timeconsuming. The inline

approaches maintain additional metadata during backup to facilitate the garbage collection. Maintaining a reference counter for each chunk is expensive and error-prone . Grouped Mark-and- Sweep (GMS) uses a bitmap to mark which chunks in a container are used by a backup.

- **performance evaluation**

Four datasets, including Kernel, VMDK, RDB, and Synthetic, are used for evaluation. Their characteristics are listed in Table 1. Each backup stream is divided into variable-sized chunks via Content-Defined Chunking . Kernel, downloaded from the web is a commonly used public dataset [. It consists of 258 consecutive versions of unpacked Linux codes. Each version is 412:78 MB on average. Two consecutive versions are generally 99% identical except when there are major revision upgrades. There are only a few self-references and hence sparse containers are dominant. VMDK is from a virtual machine installed Ubuntu 12.04LTS, which is a common use-case in real-world .

We compiled source code, patched the system, and ran an HTTP server on the virtual machine. VMDK consists of 126 full backups. Each full backup is 15:36 GB in size on average, and 90-98% identical to its adjacent backups. Each backup contains about 15% self-referred chunks. Out-of-order containers are dominant and sparse containers are less severe. RDB consists of snapshots of a Redis database The database has 5 million records, 5 GB in space. We ran YCSB to update the database in a Zipfian distribution. The update ratio is of 1% on average. After each run, we archived the uncompressed dump.rdb file that is the on-disk snapshot of the database. Finally, we got 212 versions of snapshots. There is no self-reference and hence sparse containers are dominant. Synthetic was generated according to existing approaches We simulated common operations of file systems, such as file creation/deletion/modification. We finally obtained a 4:5 TB dataset with 400 versions. There is no self-reference in Synthetic and sparse containers are dominant.

History-Aware Rewriting Algorithm

Algorithm 1 History-Aware Rewriting Algorithm

Input: IDs of inherited sparse containers, $S_{inherited}$;
Output: IDs of emerging sparse containers, $S_{emerging}$;

```
Initialize a set, S_emerging.
while the backup is not completed do
    Receive a chunk and look up its fingerprint in the fingerprint index.
    if the chunk is duplicate then
        if the chunk's container ID exists in S_inherited then
            Rewrite the chunk to a new container.
        else
            Eliminate the chunk.
        end if
    else
        Write the chunk to a new container.
    end if
    Update the utilization record in S_emerging.
end while
Remove all utilization records of larger utilizations than the utilization threshold from S_emerging.
Calculate the estimated rewrit ratio for the next backup.
while the estimated rewrite ratio is larger than the rewrite limit do
    Remove the utilization record of the largest utilization in S_emerging.
    Update the estimated rewrite ratio.
end while
return S_emerging
```

At the beginning of a backup, HAR loads IDs of all inherited sparse containers to construct the in-memory Sinherited structure During the backup, HAR rewrites all duplicate chunks whose container IDs exist in Sinherited. Additionally, HAR maintains an in-memory structure, Semerging (included in collected info in Figure 3), to monitor the utilizations of all the containers referenced by the backup. Semerging 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 Semerging. Semerging then contains IDs of all emerging sparse containers. In most cases, Semerging can be flushed directly to disks as the Sinherited of the next backup, because the size of Semerging is generally small due to our second observation. However, there are two spikes in . A large number of emerging sparse containers indicates 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 Semerging. (1) HAR calculates an estimated rewrite ratio (defined as the size of rewritten data divided by the backup size) for the next backup.

IV.References

[1] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage,"2013.

[2] G. Anthes, "Security in the cloud," Communications of the ACM, 2010.

[3] S. Ruj, M. Stojmenovic, and A. Nayak, s"Decentralized access control with anonymous authentication of data stored in clouds" 2014

[4] X. Huang, J. Liu, S. Tang, Y. Xiang, K. Liang, L. Xu, and J. Zhou, "Cost-effective authentic and anonymous data sharing with forward security" 2014.

[5] C. Gentry, "Certificate-based encryption and the certificate revocation problem," 2003.

[6] V. Goyal, "Certificate revocation using fine grained certificate space partitioning," 2007.

[7] J. M. G. Nieto, M. Manulis, and D. Sun, "Forward-secure hierarchical predicate encryption,"2013.

[8] K. Liang, J. K. Liu, D. S. Wong, and W. Susilo, "An efficient cloud based revocable identity-based proxy reencryption scheme for public clouds data sharing," 2014.

[9] D.-H. Phan, D. Pointcheval, S. F. Shahandashti, and M. Strefler, "Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts," 2013.

[10] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," 2000.