ISSN:0975-9646

Reducing Fragmentation for In-Line Deduplication Backup Storage and Cache Knowledge in Cloud

D. Shanmugasundari[1], K. Anbumathi[2]

[1]*M.phil, Research Scholar, Bharathiyar Arts and Science College for women.*
[2]*Assistant Professor, Department of Computer Science, Bharathiyar Arts and Science College for women.*

Abstract-Deduplication is process of elimination of repeated data in storage system and it is powerful technique to reduce storage costs. Fragmentation mainly caused by replication from existing backups of the same backup set, so the duplicated copies cannot be deleted from garbage collection. Cloud storage is a model of data storage where the digital data is stored in logical pools. Cloud back up refers to backing up data to a remote cloud based server. To reduce the fragmentation, we propose History-Aware Rewriting Algorithm (HAR). HAR algorithm checks the data container to avoid overflow of data. A fragmentation approach is to store the lists in the blank areas of containers, which on average is half of the chunk size. Container Marker Algorithm (CMA) provides id .Each ID is paired with a backup time, and the backup time indicates the dataset's most recent backup that refers to the container.

Index Terms – Cloud storage, Deduplication, Fragmentation problem, Restore performance.

1. INTRODUCTION

Deduplication is the paradigm in modern backup systems because of its enormous ability of improving storage efficiency. Deduplication based backup system divides a backup stream into variable sized chunks of data and identifies each chunk by its ID. An ID index is used to map stored chunks to their physical addresses. The containers are the basic unit of read and write operations. During a backup, the chunks that need to be written are aggregated into containers to preserve the spatial locality of the backup stream. During a restore, the backup stream is reconstructed according to the data. The containers serve as the prefetching unit depends upon space.

Deduplication is process of automatic elimination of duplicate data in storage system and it is most effective technique to reduce storage costs. Deduplication effects predictably in data fragmentation, because logically continuous data is spread across many disk locations. Fragmentation mainly caused by duplicates from previous backups of the same back upset, since such duplicates are frequent due to repeated full backups containing a lot of data which is not changed. Systems with in-line deduplicate data intends to detects duplicates during writing and avoids storing them such fragmentation causes data from the latest backup being scattered across older backups.

The data deduplication operation inevitably introduces some amount of overhead and often involves multiple processes at the solution level, including compression .This means that the choice of where and how deduplication is carried out can affect the speed of a backup process. The deduplication process can be applied during ingest or in post-processing. It can also occur at the destination end of a backup operation or at the source or in a hybrid mode where part of the process occurs on the target and part in software on a backup server.

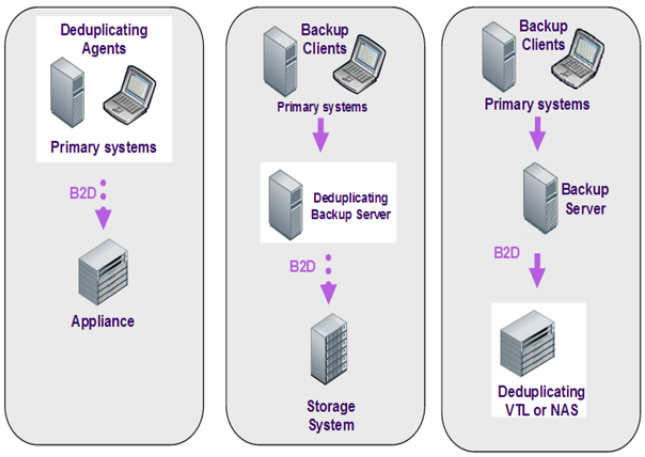

Fig 1.1Deduplicationfor Backup and Recovery

2. PROBLEM FORMULATION

This section consists of security model in cloud environment. Two kinds entities will be involved in this deduplication system, including the user and the storage cloud service provider (S-CSP). Both client-side deduplication and server-side deduplication are supported in our system to save the bandwidth for data uploading and storage space for data storing.

• ***User*:** The user is an entity that wants to outsource data storage to the S-CSP and access the data later. In a storage system supporting deduplication, the user only uploads unique data but does not upload any duplicate data to save the upload bandwidth.

•***S-CSP:*** The S-CSP is an entity that provides the outsourcing data storage service for the users. In the deduplication system, when users own and store the same content, the S-CSP will only store a single copy of these files and retain only unique data.

3. CLOUD ENVIRONMENT

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly.

Cloud computing is a practical approach to experience direct cost benefits and it has the potential to transform a data center from a capital-intensive set up to a variable priced environment.

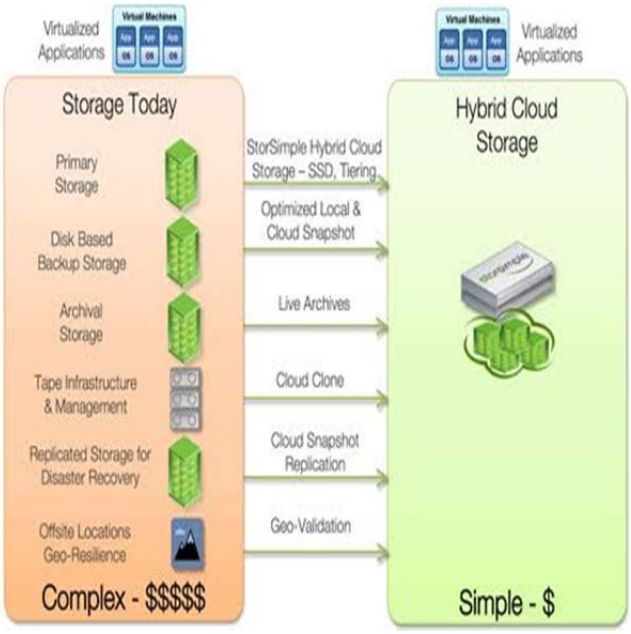

Fig 1.2 Cloud Storage

The idea of cloud computing is based on a very fundamental principal of reusability of IT capabilities. The difference that cloud computing brings compared to traditional concepts of "grid computing", "distributed computing", "utility computing", or "autonomic computing" is to broaden horizons across organizational boundaries.

4. CONTAINER MARKER ALGORITHM

The design of CMA is used to efficiently determine which containers are invalid. CMA assumes users delete backups in a FIFO scheme, in which oldest backups are deleted first. The FIFO scheme is widely used, such as Drop box that keeps data of latest 30 days for free users. CMA maintains a container manifest for each dataset. The container manifest records ids of all containers related to the dataset. Each ID is paired with a backup time, and the backup time indicates the dataset's most recent backup that refers to the container. Each backup time can be represented by one byte, and let the backup time of the earliest non deleted backup be 0. One byte suffices differentiating 256 backups, and more bytes can be allocated for longer backup retention time. Each container can be used by many different datasets. For each container, CMA maintains a dataset list that records ids of the datasets referring to the container. A possible approach is to store the lists in the blank areas of containers, which on average is half of the chunk size. After a backup is completed, the backup times of the containers referenced by the backup are updated to the largest time in the old manifest plus one. CMA adds the dataset's ID to the lists of the containers that are in the new manifest but not in the old one. If the lists are corrupted, we can recover them by traversing manifest of all datasets Hence, CMA is fault-tolerant and recoverable.

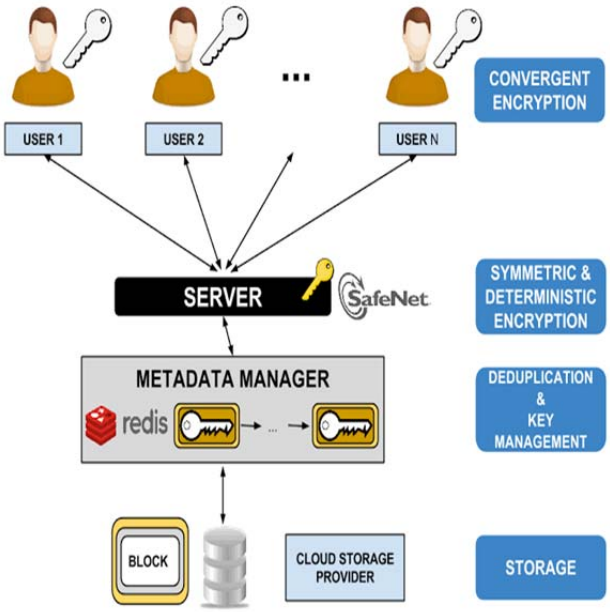

Fig 1.3 Identification Of Authenticated User

Owner Registration:

In this module an owner has to upload the files to the cloud server, and then the user is to register first. Then the user only can able to do it. For that he needs to fill the details in the registration form. These details are maintained in a database. In this module, they should login by giving their emailed and password.

User Registration And User Login:

In this module user wants to access data which is stored in a cloud, he/she should register their details first. These details are maintained in a Database. If the user is an authorized user, he/she can download the file by using file id which has been stored by data owner when it was uploading. Owner can permit access or deny access for accessing the data. So users can able to access their account by the corresponding data owner. If owner does not allow, user can't able to get the data.

File Upload:

In this module Owner uploads the file (along with meta data) into database, with the help of this metadata and its contents, the end user has to download the file. The uploaded file was in encrypted form, only registered user can decrypt it.

5 THE FRAGMENTATION PROBLEM

In this methodology, we divide a file into fragments, and replicate the fragmented data over the cloud nodes. Each of the nodes stores only a single fragment of a particular data file that ensures that even in case of a successful attack, no meaningful information is revealed to the attacker. This methodology does not rely on the traditional Cryptographic techniques for the data security. We show that the probability to locate and compromise all of the nodes storing the fragments of a single file is extremely low. The cloud manager system upon receiving the file performs: fragmentation, first cycle of nodes selection and stores one fragment over each of the selected node, and second cycle of nodes selection for fragments

replication. The cloud manager keeps record of the fragment placement and is assumed to be a secure entity. To handle the download request from user, the cloud manager collects all the fragments from the nodes and re-assembles them into a single file. Afterwards, the file is sent to the user. User can download a file on users dashboard show option that file can downloadable. The fragmentation decreases the efficiencies of restore and garbage collection in deduplication-based backup systems.

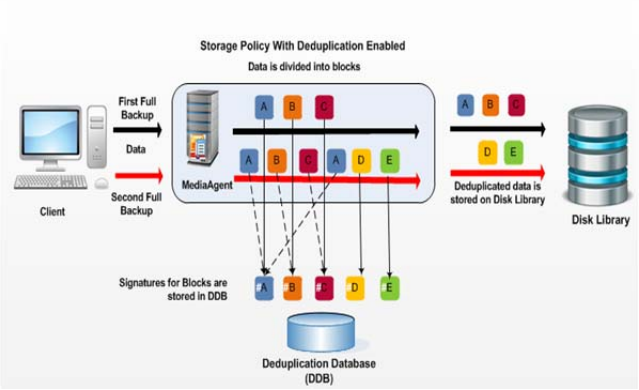

Fig 1.4 Deduplication Data Fragmentation

Our deduplication storage under investigation, like others consists of a deduplication appliance and underlying storage. The deduplication appliance resides between a host and the underlying storage While the deduplication is under our full control, the underlying storage is not (no information/control on data layout). Each data stream is a byte stream representing a series of backup dataset. The write of an incoming data stream to the underlying storage begins by segregating the stream into variable-size chunks by identification of accurate data. A deduplication process subsequently eliminates any duplicated copies of chunks called shared chunks each of which has been stored already in the underlying storage. Any remaining chunks after the deduplication are called unique or deduplicated chunks. Chunk deduplication for multiple incoming data streams is fulfilled in a FIFO manner. Finally, in a memory fixed-size container becomes full of the unique chunks, it is stored into the underlying storage as a large write. Our deduplication storage has a distinct container for a different incoming data stream as in. Reading a data stream retrieves both unique and shared chunks from the underlying storage, whereas the write stores only the unique chunks. We have two read performance related concerns:

(1) Shared chunks are likely to have been physically dispersed over different containers in the underlying storage called chunk fragmentation.

(2) Writes of concurrently incoming data streams may not exploit an expected performance benefit of large consecutive writes. In-storage containers for each data stream are desired to be physically allocated or consecutive, so that its stream read can retrieve the chunks stored in the containers sequentially.

For deduplication with a single data stream, we can easily assure the consecutiveness of the in-storage containers.

6 HISTORY AWARE REWRITING ALGORITHM

The rewriting algorithm find the data which it is existed in sparse container 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.

7. CONCLUSION

The hybrid cloud storage is useful to further improve increase 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 two algorithm such as container marker algorithm and history aware rewriting algorithm to exploit backup history and cache knowledge. With the help of CMA, the hybrid scheme significantly improves the deduplication ratio without decreasing the restore performance. Note that CMA can be used as an optimization of existing rewriting algorithms. The ability of HAR to reduce sparse containers facilitates the garbage collection. It is no longer necessary to offline merge sparse containers, which relies on chunk-level reference management to identify valid chunks. We propose a CMA that identifies valid containers instead of valid chunks. Since the metadata overhead of CMA is bounded by the number of containers, it is more cost-effective than existing reference management approaches whose overhead is bounded by the number of chunks.

The over backup cloud data and establish a variety of privacy requirements. Among various multi-keyword semantics, we choose the efficient similarity measure of coordinate matching that is as many matches as possible to effectively capture the relevance of outsourced documents to the query keywords, and use inner product similarity to quantitatively evaluate such similarity measures of deduplication. .Alongside Data Deduplication there are advances in network and disk technology that provide the very high data transfer rate, lower disk cost and higher disk IO rate.

REFERENCES

[1] F. Guo and P. Efstathopoulos, "Building a high Performance deduplication system," in Proc. USENIXATC, 2011.

[2] H.P. Shilane, N. Garg, and W. Hsu,"Memory efficient sanitization of a deduplicated storage system," in Proc.USENIX FAST, 2013.

[3] 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, 2013.

[4] M. Kaczmarczyk, M. Barczynski, W. Kalian, and C.DubNicki,"Reducing impact of data fragmentation caused by in-line deduplication in Proc. ACMSYSTOR, 2012.

[5] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti, "iDedup: Latency-aware, inline data deduplication for primary storage," in Proc. USENIX FAST, 2012.

[6] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file in Proc. USENIX FAST, 2011.

[7] M. Lillibridge, K. Eshghi, and D. Bhagwat, "Improving restorespeed for backup systems that use inline chunk-based deduplication," in Proc. USENIX FAST, 2013.

[8] (2010).How to force a garbage collection of the deduplicationFolderN [Online]. Available: http://symantec.com/business/support/index?page=//content/&id=TECH129151

[9] (2010). Restoring deduped data in deduplication systems.[Online].Available:http://searchdatabackup.techtarget.com/feature/Restoring-deduped-data-in deduplication-systems