Survey on Data Deduplication for Cloud Storage to Reduce Fragmentation

Reshma A. Fegade
ME Computer Student of
D.Y.P.I.E.T. Pimpri,
Pune India.

R. D. Bharati
Faculty of Computer Department
D.Y.P.I.E.T. Pimpri,
Pune,India.

ABSTRACT
Data Deduplication is an important technique which provides better result to store more information with less space. Cost and maintenance of Information backup storage system for major enterprises can be minimized by storing it on Cloud Storage. Data redundancy between different kinds of data storage gets minimal by utilizing data deduplication method. By giving each application differently and storing the associated information distinctly the overall disk usage can be enhanced to a great level. Cloud backup systems uses data deduplication to eliminate duplicate chunks that are present in multiple files. The duplicate chunks are substituted with the references to already present chunks through deduplication, without storing it again on cloud storage. The successive chunks are actually stored in scattered form in backup system in numerous segments (the *storage unit of cloud).*

Keywords
Cloud Backup, Data deduplication, Fragmentation.

1. INTRODUCTION
Cloud computing is a computing model which uses the hosted web services and delivers them over Internet (Wide area network) making use of standard Internet Protocols. Cloud as a term comes from the use of cloud symbol generally used in network diagrams representing a section of the Internet. In these diagrams, the cloud gets the details from some part of the network and presents a view focused on the web services provided by the cloud part of the network [5].

At present, backing up and archiving of data in the cloud is becoming very popular due to its flexibility and ease of usability it provides to the end customers. Cloud backup providers need to offer their services at a convincing price point, appreciate strong margins, and still offer reasonable Service Level Agreements (SLAs). Cloud typically uses high performance disks and tapes to store the data. While tape has a high mount and seeks time overhead when compared to disk, it is considerably cheaper, has a much longer projection life, lower bit error rate, and is more energy efficient for long term archiving. Furthermore, technologies like the Linear Tape File System (LTFS) make large tape library farms increasingly easier to use in an online fashion [2].

Some cloud backup systems are designed under a thin cloud assumption that the remote data center only provides minimal interfaces (i.e., uploading and downloading complete files). Cumulus, Brackup, YuruBackup and Duplicity are this kind of system. The thin cloud design ensures that the systems are able to back up data to almost any remote storage. In order to improve the backup speed and reduce the storage space, the backup system uses data deduplication, and delta compression due to their salient features of data compression performance.

The deduplication process is a backup process, in which the input chunks of the data are detected for duplication and duplicate chunks are not be written to the cloud. Instead, the system only keeps references to the stored chunks. Using deduplication, only new chunks of data will be written to segments and uploaded to the cloud. Thus, deduplication improves storage utilization and saves backup time for thin cloud based backup systems [4].

The Storage Networking Industry Association (SNIA) formed a special interest group in 2010 called Cloud Backup Recover & Restore (BURR) to focus on interoperability, explanations, best performs, and principles requirements for cloud backup, recover and restore. One of the cloud BURR user desires is that any cloud backup system should be able to provide fast retrievals locally. However, limited wide area network (WAN) bandwidth, poses a challenge on cloud backup services that have to transmit large datasets while adequate the requirements of the ever shrinking backup windows and recovery time objectives [5].

2. RELATED WORK
Because of chunk fragmentation in cloud backup systems using deduplication, the restore performance becomes poorer when the version number propagates. Though, existing defragmentation schemes are poor to identify fragments in cloud backup. Rongyu Lai et al. proposed NED to identify fragments in cloud backup by accurately identifying fragmented chunks in every backup. The experimentation results states NED successfully recovers the restore performance at the cost of deduplication ratio [4].

Y. Nam et al. proposed deduplication process which implanted with some kind of chunk fragmentation optimization. It associates each data stream with its own open container in the memory, so that the unique chunks from different streams can be stored into different chunk containers [7].

Fu, Min, et al. proposed History-Aware Rewriting algorithm (HAR) and Cache-Aware Filter (CAF). HAR utilizes historic information in backup systems to specifically recognise and decrease sparse containers, and CAF exploits restore cache data to identify the out-of-order containers that hurt restore performance. CAF well complements HAR in datasets where out-of-order containers are dominant. To reduce the metadata overhead of the garbage collection, they further proposed a Container-Marker Algorithm (CMA) to identify effective containers instead of valid chunks [1].

Young Jin Nam et al. proposed a deduplication scheme that guarantees required read performance of each data stream while accomplishing its write performance at a practical

level, ultimately being can be guarantee a objective system recovery time. For this, they first propose an indicator called cache aware Chunk Fragmentation Level (CFL) that evaluations degraded read performance on the fly by taking into account both entering chunk information and read cache effects. Author also displays a strong association between this CFL and read performance in the backup datasets. In order to assurance claimed read performance stated in terms of a CFL value, they propose a read performance improvement scheme called selective duplication that is initiated whenever the current CFL becomes worse than the claimed one. The key idea is to thoughtfully write non-unique (shared) chunks into storage together with unique chunks unless the shared chunks exhibit well in spatial areas. They enumerate the spatial area by using a selective duplication threshold value. Their experiments with the actual backup datasets determine that the proposed scheme achieves requested read performance in most cases at the realistic cost of write performance [8].

Author describes LBFS, a network file system intended for low-bandwidth networks. Users usually run network file systems on LANs or campus-area networks with 10 MB per seconds or more allocated bandwidth. Over slower, wide-area networks (WAN), data transfers saturate blockage links and causes undesirable delays. Collaborative programs restricts and does not responds to user input data during file I/O process, batch commands can perform process for usual execution time, and other less hostile network applications suffer lack of bandwidth allocation. Users must specify employ different methods to achieve over LAN they would use the file system [6].

3. LEVELS OF DATA DEDUPLICATION

3.1 File Level

File Level also normally stated to as single-instance storage (SIS), file-level data deduplication relates a file to be backed up or archived with those already stored using checking its qualities against an index. If the file is matchless, it is stored and the index is updated; if not, only a pointer gets pointed to the existing file which is stored in backup system. The effect is that only one case of the file is saved and successive copies get replaced with a "stub" that has pointer which points to the original file. File-level methods can be less effective than block-based deduplication. A normal change within the file can cause the overall file to be saved again. PowerPoint presentation file can have somewhat as simple as the label page altered to reflect a new presenter or date this will cause the same file to be saved again. Changed blocks between one version of file and next version can be saved using Block based Deduplication. Range of reduction ratios between is 5:1 or lesser. To reduce capacity between ranges 20:1 to 50:1 of stored data uses Block based deduplication [3].

3.2 Chunk Level (or Block Level)

Chunk level data deduplication functions on the substitute file level. The file is usually divides into segments chunks or blocks that is observed for differences of redundancy and earlier stored information. Block-level data deduplication fragments data streams into blocks, examining the blocks to define if each has been encountered before (characteristically by creating a digital signature or unique identifier via a hash algorithm for each block). If the block detected as unique content then get stored in to disk and its unique identifier is updated in an index; else, file pointer get projected to the original and unique block is stored. By changing repetitive blocks with smaller pointers somewhat than storing the block again, Space of disk is getting saved [3].

3.3 Byte Level

Exploring data streams at the byte level is alternative methodology for deduplication. By carrying out a byte-by-byte comparison between new data streams and formerly stored ones, a high level of accuracy can be provided. A deduplication product which uses this technique have one thing in common i.e. it's possible that the received backup data stream has been seen earlier, so it is revised to see if it matches similar data received previously. Products leveraging a byte-level methodology are commonly "content aware," which states that vendor has done some reverse engineering of the backup data to recognize how to retrieve information parameters such as the file name, file type, date/time stamp, etc. This technique decreases the aggregate of computation required to determine by comparing unique and duplicate data. This methodology normally occurs for post-process execution on backup data when the backup process has finished. In Backup jobs, to complete full disk performance, also require a spare of disk cache to achieve the deduplication process. The deduplication process is restricted to a backup storage stream from a particular backup set and not applied globally on backup sets [3].

4 DEDUPLICATION TYPES

4.1 Offline Deduplication

The signatures of fixed-size chunks updated to the backup storage and send these signatures to the server that shares duplicate chunks asynchronously. The index of unique signatures is stored on the Storage Area Network and it has two versions. One is structured to support sequential I/O and spatial locality. The second is indexed by partial bits for enabling random searches. Copy-on-write is used to guarantee that chunks are not changed and the signatures are valid, otherwise it will lead to data corruption. References are stored in different metadata structures used for garbage collection. Deduplication is regulated to within a specific file set that is a subsection of the global file system. This policy allows separate file sets for applications with different performance norms, as some may not allow the performance consequence introduced by deduplication. And performing effective lookup and update operations [9].

4.2 Inline Deduplication

Deduplicating data before it's retained in to disk refer to inline deduplication. This relates to post-process deduplication, also called asynchronous deduplication, which examines and reduces data after it has been stored to disk. Performing deduplication in an in-line manner needs costly lookups in the write path, which can enact a significant overhead in I/O latency. On the other side, offline deduplication may introduce extra reads from the storage, it requires more storage space, and increases concurrency issues, and increases the complexity of the deduplication process. These problems driven to the development [9].

5 PROCESS OF DATA DEDUPLICATION

In the data deduplication process, there are four main steps in chunk level data deduplication, chunking, fingerprinting, index lookup and writing.

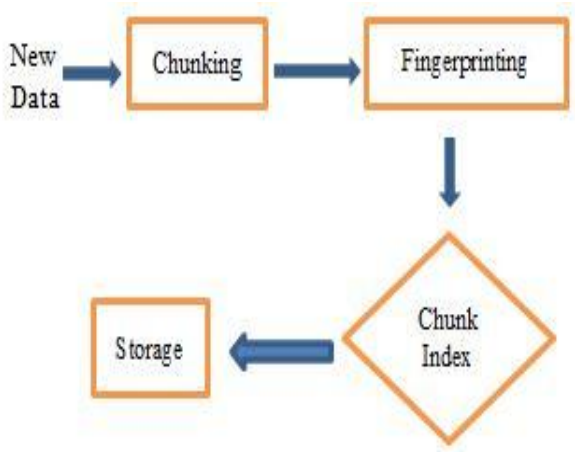

Fig 1: Deduplication Process (Chunk level) [5].

5.1 Chunking

In the chunking stage, data is divided into chunks of non-corresponding data blocks. The size of the data blocks are of two types one is fixed and another is variable size depending on the chunking technique used. The Fixed Size Chunking (FSC) process is used in the situation of fixed data blocks, while the common technique is used to produce variable sized chunks is Content Defined Chunking (CDC) [5].

5.2 Fingerprinting

Cryptographic hash functions (e.g., SHA-1) can be used as a fingerprint to calculate each chunk which gets produced from the chunking phase [5].

5.3 Index Lookup

A lookup table (chunk index) is formed and it contains the fingerprints for each unique data chunk. A lookup operation is executed for each fingerprint generated in chunking step to decide whether the current chunk is unique or not. If the fingerprint is found in the lookup table, it implies that the data chunk is not unique and vice a versa. The fingerprint is thus place in the table and the chunk is updated to the data store in writing step [5].

5.4 Writing

All unique data chunks from backup files are written or updated to the data store. Each chunk stored on the backup storage using chunk-based deduplication has a unique fingerprint in the chunk index [5].

To decide whether a given chunk is a duplicate or not, the fingerprint of the incoming data chunk is first search in the chunk index. Presence of a matching fingerprint (i.e., hash) in the index shows that a duplicate copy of the incoming chunk already present in the index (i.e., has been stored previously) and the system consequently only needs to store a reference to the existing data. If there are no unique chunks found, then the incoming chunk is unique and is stored on the backup storage system and its fingerprint put in the chunk index consequently. Deduplication can be performed either 'inline' as the data is incoming the storage system/device in real time, or as a 'post-process' after the data has been stored. Inline data duplication uses less storage space as the identical data is detected in real time before the data is stored [5].

6 CLOUD V/S TRADITIONAL BACKUP

Table 1. Cloud V/S Traditional Backup[4]

Constraint	Cloud Backup	Traditional Backup
User	Single	Enterprise
Size of Backup	Less	More
Bandwidth Allocation	Low Bandwidth	High Bandwidth
Storage Type	Cloud Server	Data Server
Restore	Segment	Container
Access	More Flexible	Less Flexible
Examples of System	Dropbox, Brackup	Hydrastor, Symantec

Table 1 shows the difference between cloud backup and traditional backup. In traditional backup, chunks are transferred to the client after being acquired from the containers in the data servers in restore process. If a data server is running many jobs at a time, reading data from disk will become a block in traditional backup. Since thin cloud system services interface of reading a complete file (i.e., segment), data is read in the client after the segments are transferred from the cloud in cloud backup. Overall, the speed of reading data from disk is much faster than that of translating data through Wide Area Network. Thus, the bottleneck of restore process in cloud backup is the segment moving process [4][9].

7 CONCLUSION

We have discussed deduplication technique and related factors which can be used for efficient deduplication process so that space utilization in backup storage system can be reduce effectively. Chunk level Deduplication process can be used for minimizing size at block level. By comparing Cloud and traditional backup storage system we concluded that for individual backup storage system cloud is better and gives more mobility as compare to traditional backup.

8 REFERENCES

[1] Fu, Min, et al. "Reducing Fragmentation for In-line Deduplication Backup Storage via Exploiting Backup History and Cache Knowledge."

[2] Gharaibeh, Abdullah, et al. "CloudDT: efficient tape resource management using deduplication in cloud backup and archival services." *Proceedings of the 8th International Conference on Network and Service Management*. International Federation for Information Processing, 2012.

[3] http://viewer.media.bitpipe.com/1019054049_245/1240950275_886/FalconStor_sDataBackup_Final.pdf

[4] Lai, Rongyu, et al. "A Near-Exact Defragmentation Scheme to Improve Restore Performance for Cloud Backup Systems." *Algorithms and Architectures for Parallel Processing*. Springer International Publishing, 2014. 457-471.

[5] Mkandawire, Stephen. "Improving Backup and Restore Performance for Deduplication-based Cloud Backup Services." (2012).

[6] Muthitacharoen, Athicha, Benjie Chen, and David Mazieres. "A low-bandwidth network file system." *ACM SIGOPS Operating Systems Review*. Vol. 35. No. 5. ACM, 2001.

[7] Nam, Youngjin, et al. "Chunk fragmentation level: An effective indicator for read performance degradation in deduplication storage." *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*. IEEE, 2011.

[8] Nam, Young Jin, Dongchul Park, and David HC Du. "Assuring demanded read performance of data deduplication storage with backup datasets." *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*. IEEE, 2012.

[9] Paulo, João, and José Pereira. "A survey and classification of storage deduplication systems." *ACM Computing Surveys (CSUR)* 47.1 (2014): 11.