IMPLEMENTING AN EFFICIENT DATA DEDUPLICATION FRAMEWORK FOR CLOUD STORAGE

Mrs. J Gnana Jeslin

Assistant Professor,
Research Scholar,
RMK College of Engineering and Technology,
Department of CSE,
Anna University,
Chennai.

Dr. P. Mohan Kumar,

Professor,
Department of CSE,
Sri Krishna College of Engineering and Technology,
Coimbatore

Abstract

As cloud computing dramatically increases, cloud service providers are under increased pressure to provide adequate data protection measures security. Cloud storage has become the recommended option to provide catastrophic recovery for users. In contrast to its convenience, customers are extremely worried that large-scale backup data would reduce storage costs. Data deduplication for backup storage is an effective approach. Current approaches for deduplicating cloud resources are not used to provide scalable backup service to cloud backup users and cannot fulfil the biased desire for various backup versions. For new versions of backup, consumers desire to minimize wait time by greater deduplicates and speed, but for older backups it is even more important to save storage costs. However, few studies take into account the problem of storage efficiency. This paper offers a new data deduction system with convergence encryption with blocking level data. The results of experiments demonstrate that the suggested system has an appropriate computational overhead and offers a potential option for data cloud storage and access scenarios in the real world.

***Keywords* – Cloud security, deduplication, ciphertext, cloud storage.**

1. Introduction

Duplication is a powerful method to avoid duplicate storage of data. In the subject of economic cloud computing, it has become a hot study topic. The current storage service is facing rapid data volume expansion and extra storage expenses resulting from the unintentional numerous needs for storage and backup. A recent Microsoft study reveals that around 68 percent of the data is duplicated. Duplication is a technique for finding the presence of a duplication and replacing it with a shared copy reference. Data from a file-level [2] or chunk-level [2] can be deduplicated,[4] and compression performance from this data is popular [5]. The latter is also popular.

The advantages of the deduplication of a Cloud Service Provider (CSP). First of all, the CSP may significantly lower its cost of storage by saving only one duplicate for each data. Second, the bandwidth of the network is retained by eliminating duplicate data transfer. Third, data administration costs are declining quickly and the CSP can give greater storage at the same rate. Concerning security and privacy, data must be kept on the cloud encrypted. However, it is far more difficult to output encrypted data to the cloud. Companies used to store low-cost backup data such drives and tape libraries.

Due to their fast and convenient disaster recovery capacity, more and more users decided to transfer the backup data into the cloud over the last several years, as the data scale has steadily expanded and cloud storage has grown. The variable price depending on the backup size does save a great deal of time by investing in early storage device for customers as well as the elastic storage capacity. The cloud backup business has therefore received increasing interest. However, it is a problem to manage increasing backup data and decrease storage expenses.

The backup data are generally cold and unavailable so that retention with minimal price and big capacity is acceptable, but at least at a slower speed of access. OSS is a sort of cloud-based storage service which can store and retrieve large volumes of data from around the world like Alibaba's OSS [1] and Amazon's S3[2]. OSS is particularly appropriate for storing unusual data such as backup information because to its incredibly low price and enormous storage capacity. While we still need to investigate additional ways to further cut storage costs, OSS delivers storage at a very cheap price. Long and ongoing user backup requirements tend to regularly upload latest file status to the cloud.

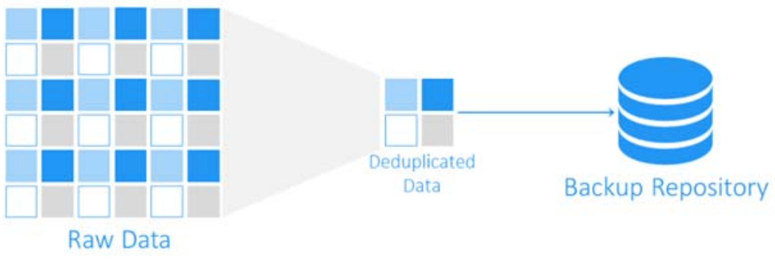

Fig 1. Overview of data deduplication process

For example, users of the database update the most recent data snapshots for fast catastrophe recovery once in a while. This leads to several backup copies being saved in the cloud, and incremental changes result in a large number of duplicates. In order to decrease data storage, data deduplication technologies can remove this redundant data. We thus design a cloud-based backup solution that benefits from cheap cost OSS storage and the deduction of data volume. Data deduplication is a widely known way of supporting large-scale storage backup systems. We note that the deduplication system may be measured by three key indicators: deduplication speed, restoration speed and deduplication ratio.

All three indicators are hard to perform best, thus most of the current work focuses just on one of those indicators. Some studies such as DDFS [3], SiLO [4], as well as Sparse Indexing [5] reach a compromise between the rate of deduplication and the deduplication ratio. Under order to recover the performance in improved physical settings but at the price of certain deduplication ratio, HAR[6], CBR[7] and Capping[8] restructure the chunks. Because companies keep backup data in their limited local storage, conventional options in the business are often backup ways to optimize the deductibility ratio[9]. Cloud storage allows user-friendly Storage expansion, which leads to some trading adjustments between three indications. Because the backup data and cloud storage are physically separated and access to OSS is considerably slower than the local disc, therefore increasing online deduplication and restoration efficiency in order to minimize the processing time.

We also notice that in the previous and new backup versions the users have a partial choice for indicators. The data value of older versions declines with time thus its storage cost may be expected to be cheaper, and the restoration procedure can be accepted by taking longer because they can be restored less frequently than new versions. Thus, the objectives of a cloud-based system are: the new version of a backup can rapidly be deduplicated and restored; for older versions, it can accurately deduct costs to save further, enabling a certain amount of efficiency sacrifice to be restored. This means that the encryption keys are in the vendors' hands and consumers should thus confide in the cloud service in full and think that it is the vendors.

To overcome this issue, many local software encryption solutions have been developed that give users with the option to execute encryption on their own by integrating with popular third-party cloud storage providers. However, due to the ever-increasing demand for data exchange, such systems that offer "secured storage" merely fail to satisfy many users' expectations. More advanced mechanisms, encrypted data exchange is necessary which are usually carried out via PKI technologies [1], where a sender first has to get a public recipient certificate before the encryption is conducted. However, the enormous volume of personal and confidential data presently available on the internet means that these mechanisms have substantial problems in regulating the main distribution process, managing computer overheads and meeting the demand for network bandwidth.

2. Literature Review

To address these issues, in 2003 Boneh and Franklin introduced the Identification-Based Encryption (IBE), which included the user identity as part of the public key. Goyal et al., Sahai and Waters presented an IBE-based attribute-based encryption system with the following characteristics: (1) senders encrypt the text on the basis of just certain unique information and so no explicit identification or number of recipients must be taken into consideration. This greatly reduces the processing overhead. (2) The message can only decrypt and therefore assure confidentiality of data, those users whose attributes comply with access rules. (3) The attacks of collision are avoided. (4) Availability policy is founded and hence expressive and flexible on the basic operations of both AND and OR.

As a result, ABE has been utilised currently in a wide spectrum of applications like encrypted audit loggers [5], pay-per-view TV [7]. Two versions of ABE, Ciphertex-Policy ABE (CP-ABE) and Key-Policy ABE, have been suggested (KP-ABE). In CPABE, only the encryption party defines the access policy connected to the ciphertext. In KP-ABE, however, the data owner has absolutely no say in who may decode the data and consequently has complete confidence in the key issuer. In the last decade, several ABE systems have been created, however few studies have taken the storage efficiency problem in ABE into account. The relevance of deduplication incentives has been noted by researchers.

In particular, some highlighted the need to encourage deduplication adoption and some utilised technological techniques to assure the performance of CSPs. Liu et al. pushed the CSP, the direct recipient of deductions, to increase storage quotas for the user whose information is deductible. Armknecht et al. suggested to limit the CSPs of ClearBox, enabling the users to confirm the conduct of the CSP. ClearBox offers the consumers of popular data with significant incentives, but it retains unpopular data.

Youn and Chang discovered the egotistical acts of data holders due to the lack of incentives in deductions. They examined the drawbacks of deduplication as the first data consumer and suggested several possible offsetting measures. They said that reducing the storage service cost of the first data consumer can assist, but did not offer a genuine mechanism. Jin et al. have found the presence of autonomous data holders who simply benefit from deduplication and stop participating in verification of data ownership. They offered a way to split the data storage costs for all holders. But the CSPs are not benefiting from this incentive mechanism. They felt that the optimum price approach would be determined by market competition among CSPs. In releasing an unjust price plan for data proprietors, Miao et al. observed the selfish conduct of CSP.

They have implemented a paid incentive to S-DEDU and have charged the users at a deductible rate. The payments' structure is like that when the stock fee payable to CSPs is shared across all holders with the same deducted data. Liang et al. [20] officially demonstrated, however, that this incentive is unable to ensure CSP profitability. To address this difficulty, the discount is limited to the overall number of information holders and other system characteristics. They have proven themselves to be individual, encouraged and lucrative and resilient through their discount-based incentive structure. Liang et al [26] have further studied the viability of their suggested economic structure in the C-DEDU of an uniform discount and personalized reduction.

They also examined how data holder's privacy may be preserved throughout the creation of an incentive system. However, because of the variation in the deduplication scheme architecture, the economic models cannot be directly applied to the H-DEDU scenario. The implementation of game theory in the solution of complex issues and the taking of optimum choices encourages interdisciplinary collaboration Many survey articles showed the enormous potential of game theory to deal with computer science security and privacy concerns. Game theoretical analysis provides significant aid to eliminate egoism and to promote acceptance of the plan, thereby assuring the long-term growth of a scheme.

The theoretical way of analyzing how cars exchange resources to improve network performance was utilised by Yu et al. Nan et al. presented a distributed bandwidth assignment process based on Game Theory in wireless multimedia social networks that successfully prevent participants' egoistic behaviors. Then, a cooperative game dealt with resource and equitable award allotments. Researchers have developed a distributed work programme based on game theory that can eliminate the selfish conduct of all entities and produce optimal social results. Chunking splits backup data into tiny pieces so that more duplicates may be identified. Fixed chunking is easy, however because of the border shift problem it has a poor deduplication ratio.

The influence of limit shift, which is the dominant chunking approach to obtain a high deduplication ratio, can be eliminated using content-defined chunking (CDC). The CDC from Rabin is frequently used, but time consuming is the calculation of Rabin hatch. Gear and FastCDC employ a simple hash to decrease the cost of computing, which is almost equal to the Rabin-based CDC, but substantially accelerated the CDC process. After the chunking, a cryptographically safe hash signature is used to calculate each fingerprint. When the same fingerprints are present, two chunks are recognized as duplicates. In order to recognize duplicates, the fingerprint index is an essential component of the duplication systems. It is viewed as the bottleneck in a large-scale deductive system since its size overflows with the exponential expansion of backup data thus it cannot be stored in the memory.

Many works are concerned with preventing fingerprint-lookup bottlenecks on the disc. The physical location is used to expedite deductions using DDFS [3], ChunkStash[12] and Sampled Index. When an identical chunk is identified, they can scan the whole container in the cache to find other copies. DDFS [3] and ChunkStash hold all index fingerprints that can accurately deduct. Indexing Sparse [5], SiLO [4] and Extreme Binning are techniques used for indexing a logical location. Indexing Sparse optimizes the usage of fingerprints in memory and uses champions to deduplicate the most representative fingerprints. SiLO and Extreme Binning take similarity detection to loosen the overhead RAM for indexing through the usage of the same similarity as single on-disk index access for the file or segment (a collection of chunks). DeFrame examines the compromise ratios between deductibility, RAM overhead indexes, and restoration performance, which provide good consideration to the design of the deductibility index.

3. Proposed System

Several typical deduplication systems information stored and conduct activities in the same machine, limiting the number of jobs a node may complete. Moreover, if any of their resources are insufficient, it is superfluous to upgrade computation and stock at the same time. The cloud scenario is intrinsic to decoupling calculation and storage. By saving cloud data, the system should store any capacity, and distribute computer resources flexibly in order to manage dynamic backup or to restore workloads. Due to its elastic expansion characteristics, the system is more economical. The solution circumstance is that customers are required to continuously back up data for full volume. There are gradual changes between different versions, therefore there are a number of duplications between nearby versions.

The system is built to make this pattern the best possible. This essentially removes duplication across versions and creates a solution for every version of the backup. The file's recipe shows the chunks in the file sequence. The history version information may be utilised to identify duplication by utilizing the recipes content. The system may also preserve files using the file recipe when restored a backup. Our technology tries to provide consumers with scalable and rapid online deduplication and restoration services. This enables us to create the Node process node. No state is saved by Node, that enable the user to dynamically create numerous Nodes to cover the demands of individual users, and to do backups and restoration operations rapidly.

Separate jobs on different Nodes can therefore be done in simultaneously. Because of continuous reference to the fingerprint database, deduction methods generally have substantial performance costs. These activities are particularly expensive in the cloud environment, as the index is set on OSS. Thus, Node becomes a lighter process by preventing frequent access to cloud data. Duplicates may be recognized quickly, eliminating many OSS access, by finding a historical or comparable version for each backup file, and using the similarity and location in the file. Moreover, it can also be made easier by historical facts. With regard to restoration performance, given the fact that sections of the backup are physically distributed over time, particularly for new back-up versions.

However, it is much easier to retrieve fresh backup versions, and the users expect to restore them quickly. We think that improving the physical location for new versions is not an acceptable option by compromising its deduplication ratios. Because the restoration requirements are uncertain for those versions and in many situations a version must not be restored. It is not smart thus to overlay all data versions. The approach presented enhances the efficiency of new versions in two different ways. First, to make efficient, online restore we create a restore cache with full restore information and LAW-based prefetching. For two reasons, we suggest a Next node.

Firstly, the deduction rate must be high and storage costs must be saved. Gnode increases the deduplication efficiency by additionally refining the results using an universal fingerprint index, thereby achieving accurate deduplication of any backup files as a result of rapid deduplication on the Node. In addition, reverse deduplication is done to correctly delete duplicates in the previous versions in order to preserve the restoration efficiency of the new version against impairment. In order to make the system better for new version data, Next node will also optimize the efficiency of restoration of the new versions by adjusting its physical storage structure with slick container compactness by transferring the data layout in old versions to new versions, thereby enhancing their location and reducing the storage efficient in new versions.

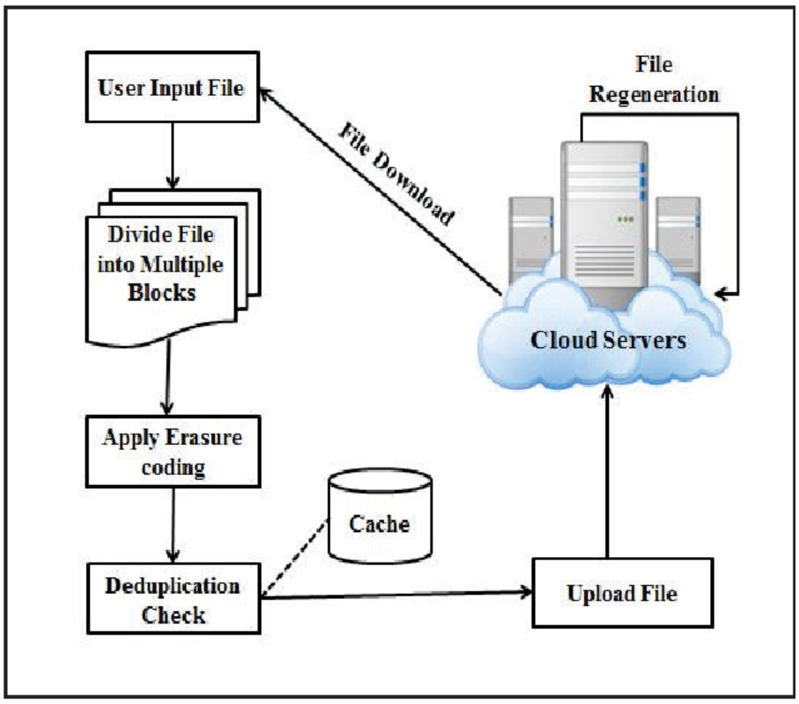

Fig 2. Proposed system architecture

All Next node actions are carried out offline, regardless of online duplication and performance. At initially it isn't cheap to access a piece from the storage, particularly from a distant OSS. The container as a fundamental storage and access device of backup data is a typical option. The remaining non-duplicate chunks are added in fixed-size containers and are kept on OSS while duplicate chunks are removed. The container repository also preserves the metadata of every container in order to acquire a chunk, which maintains the condition and the offset of each chunk and the percentage of stalled chunks. The storage of the container results in a physical location.

As a container is a group of physical bits which can be positioned nearby in the backup file, other bits in the container are probably also accessible once a chunk is reached. The number of accesses to OSS may therefore be reduced every time a container is accessed. Recipe is indeed the data model describing the pieces of a file system logically. A recipe contains chunk records, and each chunk records are kept as a quadruple, representing the chunk's fingerprint, container identification, the chunk dimension as well as the number of times that the chunk has been validated in the historical versions of the file to be duplicates. The recipe contains a logical location. Usually chunk sequencing of two backup versions are identical, given the incremental changes in the backup files. So, we employ the so-called segment structure to deduct the property.

A number of chunks are a segment in the backup file. The chunk of the recipe then generates the matching chunk records. On this basis, we may predict that between two close backup versions there are numerous identical parts. A set of indices is built for the recipe for each file to rapidly match the comparable segments and identify the part of its section. We extract several typical fingerprints as samples from the recipe index for each segment and map them to the offset of their segment recipe. The same index holds the fingerprints representative for every file that are used to discover comparable files. Overall likeness of the complete set depends heavily on the similitude of two alterations. A set of fingerprints may be viewed as a file, thus if two files share certain typical fingerprints, they are seen as comparable. History or similar file detection.

The newest version of the backup is initially searched by file path and file name for each input backup file. It's straightforward and effective to check the file name. It doesn't always match, though, because users alter file names sometimes. If so, the input file will be ticked and sampled, and a probable historical version or similar file is retrieved by searching the comparable index file using the sampling fingerprint. For many deduplication tasks, we apply a simple random sample approach that picks mod $R = 0$ for the fingerprints in a segment where R is a value that is changeable for sampling controlling. The whole input file that has not matched the name is unworkable, because all the pieces of a big file are impossible to save in the memory.

Therefore, just the header portions may be used to sample huge files. The Node will collect a recurrence index from the identified file if the historical version or comparable file is discovered. All chunks are considered as non-duplicate for files without historical versions and comparable files. Perfecting and deduplicating related segments. The input file will be chopped and sampled after you have selected the historical version's recipe Index or a comparable file. The technique of sampling is the same as previously used. The recent index is searched for the comparable section for each sampled chunk. If you have a chunk with the identical fingerprint, prefund and add it into the deduplication cache. When one sampled chunk is recognized, more chunks will also emerge due of the logical location in this segment, with high likelihood.

A range of duplicate pieces may be effectively filtered utilizing this feature in the area. The metadata of a chunk is created during the process, including fingerprint, size, container ID and duplicate times. Based on the chunks sequence in the input file, several chunks are packed into one segment in a following file. The unduplicated is saved in the new container once a segment is processed. When a container exceeds the maximum capacity, it is persisted immediately in the container shop on OSS. This segment recipe, that is attached to the recital in the recurrent after containers survive, is made up of the metadata of all chunks in the segment. The sampled chunks' fingerprints and indeed the offset of the segmental recipe are maintained and finally included in the recipes index.

4. Results and Discussion

The proposed system is used in a network of seven cloud elastic computer services (ECS), each of which has a 16-core and 64GB input Intel Xeon 2.50 GHz CPU. We note that skip chunking increases considerably deduplicate performance. In particular, after implementing a skip chunking, Rabin-based CDC improves 2-fold speed and also increases its output by 1.5 times.

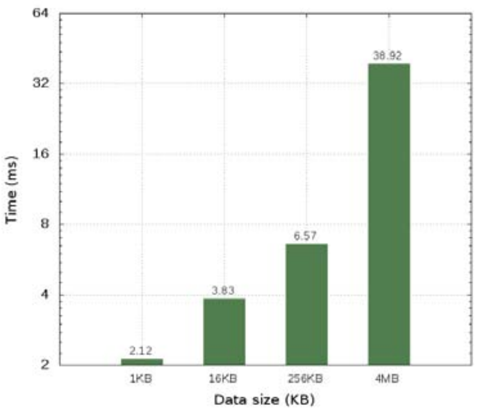

Fig 3. Deduplication time analysis

Based on observation, the CPU is the obstacle during deduplicating, historically conscious skip chunking may minimize the byte byte-byte computer overload and therefore save a great deal of CPU time and speed up deductibility. The CPU costs for CDCs are decreased to around 2 percent when skipping is used. Cut-offs do not have harm to the deduplication ratio and are deductible in the same way as Rabin-based CDC and FastCDC are achieved. For the evaluation of the effect of skipping we utilize files of varying duplication ratio in S-DB. We note that perhaps the improvement in strength is linked to the duplicate ratio, and a greater duplication ratio is achieved, as it has more consecutive replica chunks, thus it is much more likely to succeed.

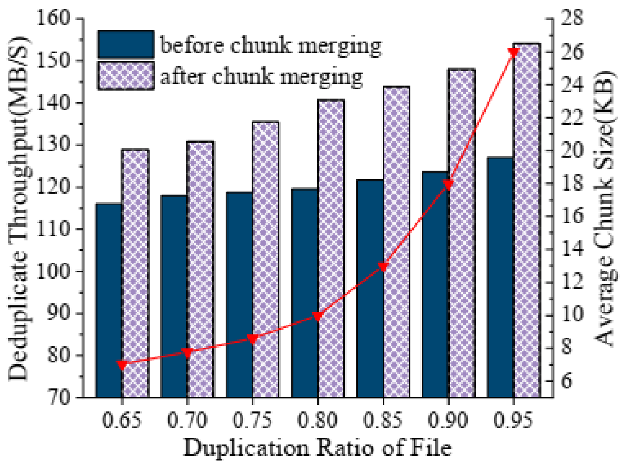

Fig. 4. Deduplication throughput

Deduplication performance may be shown to rise as the amount of the chunk increases and stabilizes after 32 kb. The deduplication proportion deteriorates, and when the chunk is bigger than 16 KB the negative trend grows steeper. History-aware merging of the chunk is therefore recommended to adjust dynamically to a suitable chunk size that might compromise the deduction speed and the deduplication rate. In order to check the historically-witting combination, we adjust the file duplication ratio, and the starting chunk size is 4KB. History-conscious chunk fusion can obviously enhance duplicate output.

The increase is more than 20% for the file with a duplication ratio of 0.95, ranging from 125MB/s to 155MB/s at the price of a deduction ratio of just 0.9%. But the file having a lower replication ratio has a lower profit and a greater loss ratio for the chunk merger. The difference is because files having a high duplication proportion combine more super-stuffs, which causes the average chunk size to be big and the big chunks speed the deduplication process following the merger. In additionally, a high duplication ratio implies that the information is less likely to be updated, hence for any file with a high duplication ratio, the deductions resulting from the super-chunk modification are also minimal.

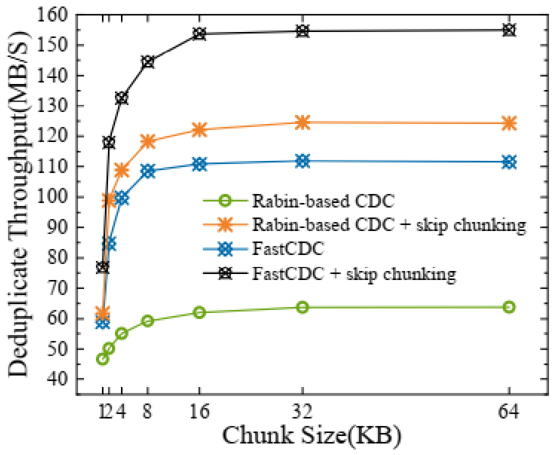

Fig.5. Comparative analysis of throughput

Restoration with two high efficiency objectives and minimal use of OSS bandwidth is optimized. Therefore, to show our optimization, we utilize the restoration and reading container per 100MB. In order to assess the effect of full view cache and spare container compaction, we constantly backup 25 versions and then restored them under various cache sizes to deactivate LAW-based prefetching. Because sparse containers are rare before this, restoration efficiency depends on the cache restores capacity to solve large-scale containers and reference chunks. Three restore-caches have an influence from partially expanded views.

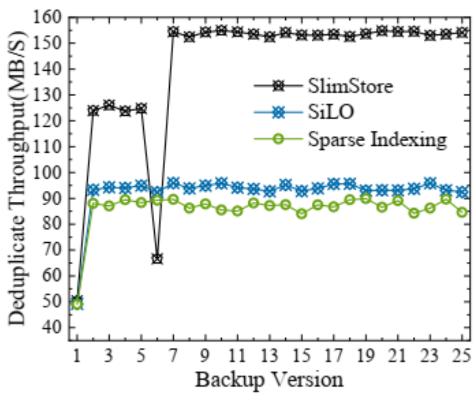

Fig. 6. Backup structure

Finally, we evaluated the way in which measurement measures are impacted by data owner's operational costs and summarized the results that linearly decrease the number of iterations with the increase. Intuitive, greater data owner operating expense equals less service, as has been shown. An online probability will emerge as a direct method to improve the usefulness of a data owner. But the costly operating costs cannot still be fully offset. The increasing online likelihood encourages the desire of data holders to participate; consequently, the deduplication rate increases and the CSP may save more space and provide more savings. The utilities curves of the data holders and CSP show that there is no negative effect of this growth and that the only participant obviously affected is the data owner whose utilities have directly to do with the value of the operating cost.

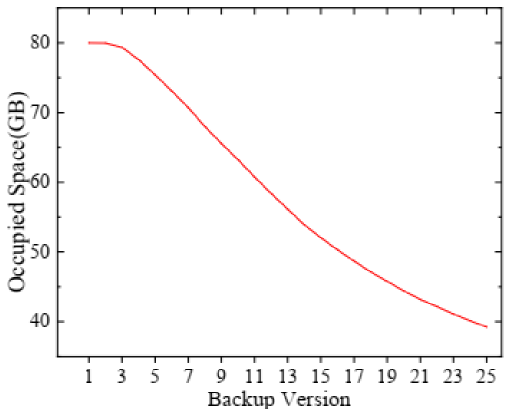

Fig.7. Space variation analysis

We pick the "customer-side" data duplication technique which is used to analyse customer device detection duplicates. The time the input files of the varied sizes are checked duplicate. It illustrates the time to deduplicate a 5 MB file processed with different chunking sizes. Although it is apparent that the processing of tiny chunks is faster, a substantial deterioration in performance is caused by an increase in the number of duplication controls. However, the improvement in storage efficiency is favorable with the lower piece size. In actuality, the chunk size choice does not have an absolute norm. The process of deduction of data needs to some extent correlate overhead performance, while smaller chunk sizes increase the space efficiency. We have to balance efficiency and storage efficiency with demand. Usually, a file has a better probability of being deduplicated with a bigger amount of data kept by the cloud storage provider. Significant reductions can therefore be realized in transmission time.

5. Conclusion

This article describes a cloud-based deduplication platform that offers large-scale multi-version storage deduplication and recovery services online. It quickly reproduces and repairs new backup versions while making sure that deduplication efficiency reduces storage costs. Several strategies for improving its efficiency are offered. Each of these techniques is rather easy in isolation. What is new is that these concepts are designed and combined in an effective and cohesive system that achieves the aims of design. Experimental findings show that the system performs very fast deduplication and restore and may effectively decrease storage costs by eliminating duplicating data.

References

[1] Cao, S. Liu et al., (2019):Sliding look-back window assisted data chunk rewriting for improving deduplication restore performance," in FAST, pp. 129–142.
[2] Cao, H. Wen et al., (2018):ALACC: accelerating restore performance of data deduplication systems using adaptive look-ahead window assisted chunk caching, in FAST, pp. 309–324.
[3] Fu, D. Feng et al., (2016):Reducing fragmentation for in-line deduplication backup storage via exploiting backup history and cache knowledge," IEEE Trans. Parallel Distrib. Syst., **27(3),** pp. 855–868.
[4] Fu, D. Feng et al., (2015):Design tradeoffs for data deduplication performance in backup workloads, in FAST, pp. 331–344.
[5] Jin, L. Wei, M. Yu, N. Yu, and J. Sun, (2013):Anonymous deduplication of encrypted data with proof of ownership in cloud storage," in 2013 IEEE/CIC International Conference on Communications in China (ICCC). IEEE, pp. 224–229.
[6] Kulkarni and Shanbhag U. V, (2015), An existence result for hierarchical stackelberg v/s stackelberg games," IEEE Transactions on Automatic Control, **60(12)**, pp. 3379–3384.
[7] Liang and Z. Yan, (2019):A survey on game theoretical methods in human–machine networks,Future Generation Computer Systems, vol. 92, pp. 674–693.
[8] Liang, Z. Yan, and R. H. Deng, Game theoretical study on client-controlled cloud data deduplication, Computers & Security, vol. 91, p. 101730, 2020.
[9] Manshaei, Q. Zhu, T. Alpcan, T. Bacs¸ar, and J.-P. Hubaux, (2013):Game theory meets network security and privacy, ACM Computing Surveys (CSUR), **45(3),** pp.25.
[10] Nam, G. Lu et al.,(2011):Chunk fragmentation level: An effective indicator for read performance degradation in deduplication storage, in HPCC, pp. 581–586.
[11] Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, (2010):A survey of game theory as applied to network security," in 2010 43rd Hawaii International Conference on System Sciences. IEEE, pp. 1–10.
[12] Yan, X. Liang, W. Ding, X. Yu, M. Wang, and R. H. Deng, (2019):Encrypted big data deduplication in cloud storage,in Smart Data: State-of-the-Art Perspectives in Computing and Applications, CRC Press, , **4**, pp. 63–92.

Authors Profile

Mrs. J. Gnana Jeslin received her B.E. degree in computer science and engineering from Anna University, Chennai, India, in 2010. She received her M.E. degree in computer science and engineering from Sathyabama University, Chennai, India, in 2012. Her interests include neural networks, Artificial Intelligence and cloud data security. Currently working as Assistant Professor in the Department of Computer Science and Engineering, RMK college of Engineering and Technology, Anna University, Chennai, India.

Dr. P. Mohan Kumar has completed his undergraduate in Computer Science and Engineering in Manonmaniam Sundaranar University in the year 2001. He completed his postgraduate in Computer Science and Engineering in Anna University in the year 2004. He got his Ph. D in Computer Science and Engineering from Sathyabama University in 2012. He has teaching experience over 18 years. He has published more than 30 research papers in national and international journals. He is currently working as Professor of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore