In-line Deduplication for Cloud storage to Reduce Fragmentation by using Historical Knowledge

Smitha .M. S, Prof. Janardhan Singh
Mtech Computer Networking, Associate Professor
Department of CSE, Cambridge Institute of Technology, Bangalore - 560036, India

***Abstract*—This Recovery and Backup system in which the process involves that copying and archiving of data on different cloud server, so that this data is used to recover the unique data, afterward a loss event. Purpose of backup is to recover data after its loss and to improve data from a past time. In backup systems, the fragments of every data file are physically distributed over multiple servers, which increase privacy of data files. We can use deduplication compressions techniques, but there is a need to improve such systems with respect to capacity of storage and to improve recovery of files. In this paper, we implement the erasure code, inline de-duplication checking and file restoring to improve the performance of backup system. To evaluate the performance of system, experiments are carried out on multiple files, ranging from 1KB to 100MB. Experimental results prove that the proposed system is better than existing one, in terms of minimized storage space capacity and data recovery.**

***Keywords*— Cloud computing, Backup systems, Fragmentation, Data loss recovery, Deduplication.**

I. INTRODUCTION

Cloud computing is a service also called as 'on-request computing'. Cloud computing is also called as Internet- founded computing and provides on-demand shared resources, data and user information to various computers and many other devices. It enables on-request access of shared resources for anywhere. Third-party data centers stores and processes users data and Cloud computing allows with their storage solutions provided to users and enterprises with various capabilities. Cloud computing technology shares resources to attain consistency and markets of scale on the utility style costing. Cloud computing is the broad concept covered with huge organization and multiple shared services. Cloud computing is an archetype for enabling appropriate, on request system access to a shared configurable computer resources such as services, storage networks, servers, applications etc. Resources are promptly examined and released with minimal management effort. Now a days cloud computing is a vastly claimed service or utility because of the benefits of high computing power, lower service cost, high performance, scalability, accessibility and availability.

The current information deduplication plans for essential data storage for example, inline deduplication and Offline-deduplication. Offline-Dedupe and iDedup are capacity based in that they concentrate on capacity of reserve funds and only choose the substantial request to deduplicate and sidestep for small demands. The little I/O asks for record for a modest fraction of the capacity limit. It is necessary to make deduplication unprofitable and probably counterproductive since the deduplication is overhead. Previous workload studies have noticed that minor records guidelines in essential storage frameworks (over half) and are at the cause of the framework. Cache Knowledge execution bottleneck is used to mitigate fragmentation and besides because of the cradle impact on essential stockpiling workloads show evident I/O burstiness. Considering performance, the current information deduplication plans disregard to consider workload qualities in essential storage frameworks. There is chance to face most imperative issues in essential storage of performance [8].

Erasure Coding has been widely used to provide high availability and reliability while introducing low storage overhead in storage systems. Erasure coding is a process of data protection from being corrupting and misplacing during processing in different applications. This method protects data from fragmented into segments, extended, encoded and avoid redundant data bits and stored at different locations or storage media system. The goal of erasure coding is to reconstruct corrupted data by using information that can be metadata about the data stored at in another array in the disk storage process. Erasure coding is beneficial with huge amounts of data and any applications or schemes. System should be tolerating failures, like data grids, disk array, distributed storage applications and archival storage, object storage. Currently erasure coding is used in object-based cloud storage. Here, we study regarding the previous related work done on the deduplication of backup storage in next section II, the implementation details and in the section III, in which we can see system architecture, different modules description, mathematical models details, algorithms and experimental setup. We have included expected results and overall conclusion of paper is in section V.

II. RELATED WORK

Restoration performance and efficient value of garbage collection get decreases when we stores data into sparse container. If restore cache size is small then restore performance of the out-of-order container decreases.

Proposed [1] History-Aware Rewriting algorithm (i.e. HAR) and Cache Aware Filter (i.e. CAF) to decrease the data fragmentation. Pervious historic information of old backup in backup systems is used to detect and reduction of sparse containers by HAR and CAF that takes benefit of restore cache knowledge to categorize the out-of-order containers that damages restore performance. CAF efficiently recognizes dominant out-of-order containers and complements HAR in datasets. Proposed Container Marker Algorithm (i.e. CMA) is used to discover valid containers rather than valid chunks, to condense the information of data (metadata) overhead of the garbage collection.

Author [2] proposed, a new indicator for dedupe scheme. Proposed scheme provides two-fold approach, first, a novel pointer for dedupe system called cache-aware Chunk Fragmentation Level (i.e. CFL) display and second discriminating duplication for improvement read performance. The CFL is consists finest chunk fragmentation parameter and cache-aware recent chunk fragmentation parameter. Selective duplication technique is activated if the current CFL becomes worse than the required one to boost read performance. Achieving performance at a reasonable level, proposed scheme guarantees required read performance of each and every data stream and also guaranteed an object system recovery time. Selective duplication is a major limitation; it needs additional memory size called in memory temporary container.

Author proposed [3] Migratory Compression-MC, a coarse grained method of data transformation to increase the efficiency of old compressors in advanced storage media systems. To improve compression factors, relocation of similar data chunks gathered into one unit, also compressing provides a clear improvement and migrated chunks return to their previous locations by decompression. Compared with traditional compressors, proposed scheme improves compression effectiveness by 11 percent to 105 percent. Deduplication process is eliminates duplicates in data and helps to improve the effective capacity of storage systems. Single-node raw capacity is cause to force users to resort to complex, capacity still mostly limited to tens or a few hundreds of terabytes.

Author proposed [4] progressive sampled new mechanisms that performs indexing, is used to removes scalability boundaries. Proposed scheme increases throughput, scalability and provide good deduplication efficiency.

Author proposed [5] three fold approaches, first they discuss sanitization requirements in the context of deduplicated storage, implemented a memory effective system for managing data based on perfect hashing, third they design neat deduplicated storage for data domain. Proposed approach mitigates I/O requirements and memory. Proposed scheme conflicts with a static fingerprint space required by perfect hashing and aim to support host writes during sanitization Data deduplication. Data deduplication has recently gained importance in the most of secondary storage and even some primary storage for the efficient storage purpose. Great significance gained by an input/output read performance of the deduplication storage system. Author introduced [6] a new technique called Chunk fragmentation level-CFL that is for better read performance. To maintain the indicator for the read performance of a data stream this proposed approach is very effective with deduplication storage.

Author proposed [7] the context based algorithm for rewriting selected few duplicates. Proposed algorithm improves restore speed of the latest backups and also increased resulting in fragmentation for older backups that are rarely used for restore. Proposed algorithm has a limitation that reduces write performance a little bit.

Author proposed [8], an iDedup i.e. inline deduplication solution for major workloads. Projected algorithm is roughly based on two keys. A perception from real dynamic world workloads i.e. spatial locality occurs in duplicated primary data and another second temporal locality occurs in the access patterns of duplicated data of backup. Proposed algorithm minimizes extra Input/Outputs and seeks

III. IMPLEMENTATION DETAILS

A. System Overview

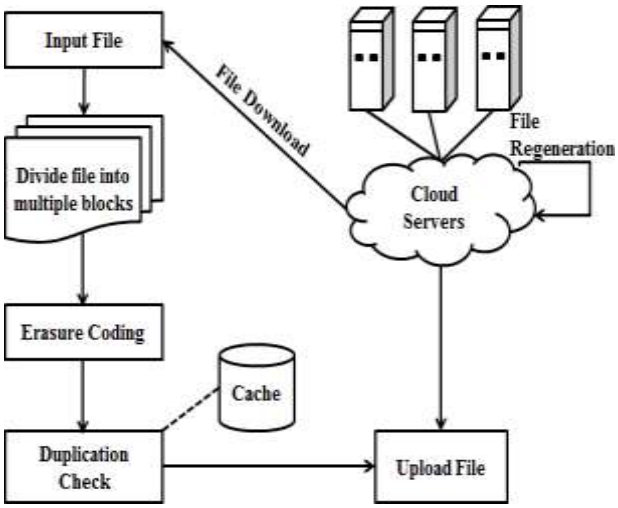

Fig.1. Architecture for In-line Deduplication System for Cloud storage.
Here, Cache gives historical knowledge to check duplicate data.

The Cloud iDedup (Cloud Inline Deduplication) system consists of cloud server and multiple users. This system is applicable for business applications. The existing system performs de-duplication checking of files store on containers. The system used hash code and cache knowledge for de-duplication checking of file or block. But system has drawback that it required large storage space for backup. To overcome this drawback, our proposed system uses erasure code and it is applicable on cloud [1].

The proposed system can be explained using following modules:

User Registration-login

When new user enters the system, needs to be successfully registered on cloud server for authentication. For this registration, user should provide username and password. After successful login and authentication, cloud server provides access to users to use facilities provided by server.

Input File Fragmentation

After successful login, user wants to store their file on cloud server in distributed manner. For this purpose, input files are subdivided into several fragments. These fragments are store on multiple servers on cloud storage, which are accomplished by single main cloud.

Erasure Code

Erasure code is a technique in which data get protected by scattering into fragments, extended and coded (that can be encoded or encrypted) with duplicated data pieces and further stored among the set of diverse locations of storage system. The basic target of this technique is to permit stored data that becomes corrupted because of any natural or artificial failure at some point in the storage process to be recreated by using metadata which is stored to another place in the stored array.

Inline De-duplication Checking

This module check the duplication file storing on server. That is when this module receives fragment to store on server, it check its hash value in cache. If hash is not available in cache, it uploads the fragment on server, otherwise discard that fragment. This module decreases the memory wastage, because only unique files are store on server and removes the duplicated files.

File Upload

In this module, user can upload the file on cloud. For this, file is fragmented and store on multiple servers in distributed manner. This process maintains the security and privacy of data stored on multiple servers and reduce the chances of data corruption.

File Download

User can also download the files distributed on multiple servers. If anyone fragment of the file is loss or corrupted, can be recovered in file restore module. This corrupted file can be restoring

B. Algorithm

Input: File

Output: Upload File/Download File

Chunking Preprocess and Hash Generation:

1. chunk_size=(File_size)/4.

2. Generate hash values using SHA-1 and store it into database.

3. Check Deduplication:

-Check Hash values into the client side database.

-If hash values of input values are same as pervious stored hash values in the database.

-Then file is duplicated.

-Discard File.

-Update File Name.

-Else Store File into cloud server.

Algorithm states that historical information can be used in the current duplication detection process to detect duplicate data. History Aware Rewriting Algorithm also gives utilization records which stores current backup records of duplicated data that will be referred in further backup deduplication process [1].

C. Mathematical Model Set theory

System (S) is denoted as S = {C,B,U,D}

1) Client-

C = {C1, C2, C3, ……..,Cn}

Here, C is the set of client and C1, C2, C3,....., Cn are the number of clients.

2) Input files-

I = {I1, I2,….,In}

Here, I is the set of all input file, which will be store on cloud server.

3) Divide file into Block- B = {B1,B2, B3,….,Bn}

Here, B is the set of block and B1, B2, B3,....., Bn are the number of blocks.

4) Apply Erasure code-

E = {E1,E2,…,En} Here, E is the erasure code of all file blocks.

5) Inline deduplication check-

De-duplication check for each block.

6) Upload file into cloud server-

If the block is not duplicated then store it on cloud server.

D. Experimental Setup

The proposed System is developed on Java platform using jdk 1.8 (i.e. Java Framework) on Microsoft windows. The Netbeans version-8.0 is a tool which is used for development. This system is not restricted or strictly depended on any hardware to run; any standard machine is adept to run this application. This proposed system supports datasets contains files.

IV RESULT AND DISCUSSION

Data Set:

In this work we have used multiple files. The Input Files are of various sizes varying from 1 KB to 100MB. Following table I describe the Storage space required for specific file size.

TABLE I
STORAGE SPACE REQUIRED FOR FILE

File Size	Existing System	Proposed System
10KB	20KB	14KB
20KB	40KB	28KB
30KB	60KB	42KB
40KB	80KB	56KB
50KB	100KB	70KB

For example, 10KB file requires 20KB and 14 KB storage space in existing and proposed system respectively. We have considered five different files from 10 KB to 50 KB which we can store on multiple cloud servers. Existing system can store those files up to 20KB whereas proposed system can minimized file storage space from 20 KB to 14 KB and further for other files. Here, we are taking backup of the file on multiple servers but not storing duplicate file on same server.

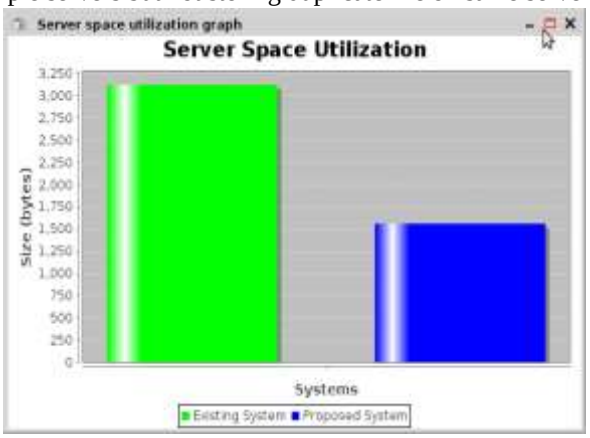

Fig. 2. Represents the comparison of existing and proposed system. Proposed system utilizes less space to store fragments, compare to existing system and also it provides same security level. Where X co-ordinates are file size in KB and Y co-ordinates represents storage space.

CONCLUSION

To maintain the efficiency and to avoid data corruption in data storage backup system are challenging tasks. Storing data fragments on multiple servers reduces the chances of data loss but these data fragment storage on multiple servers for data backup increases storage space. To save the storage space, our proposed system implements erasure coding technique, this can restore the corrupted data files if any fragment is loss or corrupt. Also to reduce the computation cost, system makes use of cloud servers to store the data, as cloud server has its own advantages; security, low cost, high availability, etc. To evaluate the performance of proposed system, the experiment carried out on dataset having multiple files. The file size varies from 1 kb to 100 mb. The experimental result shows that, our system is better than existing system in terms of storage space, cost, and availability of data.

ACKNOWLEDGMENT

We authors are thankful to the all researchers and publication houses as they have made their research information accessible for us. We are also thankful to the review examiner for their corrective ideas and suggestions. We are also thankful to the college faculty and authorities for providing the required infrastructure and helpful support. Lastly, we would like to prolong a sincere thanks to friends and family members.

REFERENCES

[1] Fu, Min, et al. "Reducing Fragmentation for In-line Deduplication Backup Storage via Exploiting Backup History and Cache Knowledge." *IEEE Transactions on Parallel and Distributed Systems* 27.3 (2016): 855-868.

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

[3] Lin, Xing, et al. "Migratory compression: Coarse-grained data reordering to improve compressibility." *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*. 2014.

[4] Lin, Xing, et al. "Migratory compression: Coarse-grained data reordering to improve compressibility." *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*. 2014.

[5] Botelho, Fabiano C., et al. "Memory efficient sanitization of a deduplicated storage system." *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*. 2013.

[6] 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.

[7] Kaczmarczyk, Michal, et al. "Reducing impact of data fragmentation caused by in-line deduplication." *Proceedings of the 5th Annual International Systems and Storage Conference*. ACM, 2012.

[8] Srinivasan, Kiran, et al. "iDedup: latency-aware, inline data deduplication for primary storage." *FAST*. Vol. 12. 2012.

[9] Zhu, Benjamin, Kai Li, and R. Hugo Patterson. "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System." *Fast*. Vol. 8. 2008.

[10] Li, Jin, et al. "A hybrid cloud approach for secure authorized deduplication."*IEEE Transactions on Parallel and Distributed Systems* 26.5 (2015): 1206-1216.