Enhancement Of De-Duplication And Security In Cloud Using Chunks

Ms. MC. Marakatha valle[1], Ms. S. Purnambigai[2], Ms. M. Vanitha[3]
[1, 2, 3] Dept of Information Technology
[1, 2, 3] Saranathan College of Engineering, Tiruchirapalli-620012, Tamil Nadu, India.

Abstract- *Data de-duplication is one of the most important data compression technique which is used to remove duplicate copies of repeating data, and it has been widely used in cloud storage to increase the amount of storage capacity and to save bandwidth. The convergent encryption technique has been proposed to encrypt the data to provide data confidentiality while supporting de-duplication. This project addresses the problem of data de-duplication, to provide better data security. In this the uploaded file is divided into fragments, and reproduced the fragmented data in the cloud nodes. It makes use of graph T-coloring to prohibit attacker in locating the data by storing the fragments at certain distance apart. Furthermore, it makes use of computationally expensive methodologies to improve performance of primary storage systems and to decrease performance overhead.*

Keywords- De-duplication, Fragmented Data, Encryption Techniques, Data Security, Cloud Nodes.

I. INTRODUCTION

1.1 CLOUD COMPUTING

The cloud computing paradigm has reformed the usage and management of the information technology infrastructure. Cloud computing is characterized by on-demand self-services, ubiquitous network accesses, resource pooling, elasticity, and measured services. The characteristics of cloud computing make it a striking candidate for businesses, organizations, and individual users for adoption. The future aspects of computing as a utility is cloud computing, where cloud customers can remotely store their data into the cloud so as to enjoy the on-demand high-quality applications and services from a shared pool of configurable computing resources. The standards for connecting systems and software will need cloud computing. The privacy concerns are possessed by cloud computing because the service provider can access the data in the cloud at any time. The cloud data concerns certain space and crashes other data while interrupting it. The computing model brought may benefits including: the burden for storage management is relieved, global data access with independent geographical locations, and reduction in capital expenditure on hardware, software, and personnel maintenance.

1.2 DATA DE-DUPLICATION

Technique in Cloud backup and archiving applications to reduce backup window, improve the storage-space efficiency and network bandwidth utilization. Recent studies reveal that moderate to high data redundancy clearly exists in VM (Virtual Machine) enterprise and High-Performance Computing (HPC) storage systems. These studies have shown that by applying the data de-duplication has been demonstrated to be an effective technology to large-scale data sets HPC storage systems, can be achieved. For example, the time for the live VM migration in the Cloud can be significantly reduced by adopting the data de-duplication technology. Primary and secondary de-duplication systems can be further subdivided into inline and offline de-duplication systems. Inline systems de-duplicate requests in the write path before the data is written to disk. Since inline de-duplication introduces work into the critical write path, it often leads to an increase in request latency. On the other hand, offline systems tend to wait for system idle time to de-duplicate previously written data. Since no operations are introduced within the write path; write latency is not affected, but reads remain fragmented. The rationale is that the small I/O requests only account for a tiny fraction of storage capacity requirement, making de-duplication on them unprofitable and potentially counterproductive considering the substantial de-duplication overhead involved. However, previous workload studies have revealed that small files dominate in primary storage systems and are at the root of the system performance bottleneck. Furthermore, due to the buffer effect, primary storage workloads exhibit obvious I/O burstiness. From a performance perspective, the existing data de-duplication schemes fail to consider these workload characteristics in primary. The storage systems will miss the opportunity to address one of the most important issues in primary storage, that of performance.

1.3 DEDUP-METADATA CACHE

This is a fixed-size pool of small entries called content nodes, managed as an LRU cache. The size of this pool is configurable at compile time. Each content-node represents a single disk block and is about 64 bytes in size. The content-node contains the block's DBN and its fingerprint. In the prototype, use the checksum of the

block's contents as its fingerprint. Using a stronger fingerprint would increase the memory overhead of each entry by, thus leading to fewer blocks cached. Other than this effect, using MD5 is not expected to alter other experimental results. All the content-nodes are allocated as a single global array. This allows the nodes to be referenced by their array index instead of by a pointer. Each content node is indexed by three data structures: the fingerprint hash table, the DBN hash table and the LRU list.

1.4 SELECT-DE-DUPE PERFORMANCE

Full-Dedupe, iDe-dup and Select Dedupe all use the fixed cache partition that allocates equal spaces to the index cache and read cache. Shows the response-time performance of the different de-duplication schemes normalized to that of the Native system, driven by the three traces. Full-De-dupe degrades the Native system performance for the homes trace, indicating that directly applying data de-duplication to primary storage systems may introduce extra performance overhead. iDe-dup improves the performance of Native system slightly, which indicates that capacity-oriented de-duplication schemes are not effective in improving system performance. The web-vm and homes traces respectively, but outperforms the Native system for the mail trace. The read performance degradation is caused by the read amplification problem for the web-vm and homes traces. Since the mail trace has a significant percentage of fully redundant write requests, the positive effect of reducing a large number of write requests far overshadows the less serious read amplification problem, thus improving the read performance. iDe-dup achieves comparable read performance to the Native system by only de-duplicating redundant large I/O requests to reduce the HDD seek overhead. In contrast, Select-Dedupe consistently outperform the Native system in read performance by for the web-vm, homes and mail traces, respectively.

Select-Dedupe improve the read performance indirectly by reducing the write traffic and avoiding the read amplification problem as much as possible. The significant number of reduced write requests in Select-De-dupe greatly shortens the length of the disk I/O queue and relieves its pressure, thus allowing the read requests to be serviced more quickly. We also plot the normalized storage capacity used by the different schemes. Full-Dedupe saves the largest amount of storage capacity among all the de-duplication-based schemes because it de-duplicates all redundant write data, which is not the case for iDe-dup and Select-Dedupe. Select-Dedupe achieve comparable or better capacity savings than the capacity-oriented de-duplication scheme iDe-dup, especially for the mail trace. This is because, while iDedup only de-duplicates large I/O requests, Select- de-duplicates both large and small I/O requests. When the small I/O requests become a major part of the total requests, the capacity saving is also increased accordantly.

1.5 INDEXING AND DUPLICATE ELIMINATION

As the index update process incorporates information about recently modified blocks recorded in the write logs, in addition to detecting hash matches that indicate potential duplicates, it also performs the actual COW sharing operations to eliminate these duplicates. The duplicate elimination process must be interleaved with the index scanning process because the results of block content verification can affect the resulting index entries. In order to update the index, a host sorts the recent write records by hash and traverses this sorted list of write records in tandem with the sorted entries in the index. A matching hash between the two indicates a potential duplicate, which is handled differently depending on the state of the matching index entry. Overview of all possible transitions a matching index entry can undergo, given its current state. When DEDE detects a potential duplicate, it depends on the file system's compare-and-share operation, described to atomically verify that the block's content has not changed and replace it with a COW reference to another block. Based on user-specified policy, this verification can either be done by reading the contents of the potential duplicate block and ensuring that it matches the expected hash, or by reading the contents of both blocks and performing a bit-wise comparison. If the latter policy is in effect, hash collisions reduce DEDE's effectiveness, but do not affect its correctness. Furthermore, because hashes are used solely for finding potential duplicates, if SHA-1 is ever broken, DEDE has the unique capability of gracefully switching to a different hash function by simply rebuilding its index. The content verification step can be skipped altogether if a host can prove that a block has not changed; for example, if it has held the lock on the file containing the block for the entire duration. The write record was generated and no write records have been dropped, while this is a fairly specific condition.

1.6 RUN-TIME EFFECTS OF DE-DUPLICATION

DEDE operates primarily out of band and engenders no slowdowns for accessing blocks that haven't benefited from de-duplication. It can also improve file system performance in certain workloads. Further reducing the working set size of the storage array cache. For access to de-duplicated blocks, however, in-band write monitoring and the effects of COW blocks and mixed block sizes can impact the regular performance of the file system. Unless otherwise noted, all of our measurements of the run-time effects of de-duplication were performed using IOmeter in a virtual

machine stored on a volume of an EMC storage array. Overhead of In-Band Write Monitoring since DEDE's design is resilient to dropped write log entries, if the system becomes overloaded, we can shed or defer the work of in-band hash computation based on user-specified policy. Still, if write monitoring is enabled, the hash computation performed by DEDE on every write IO can represent a non-trivial overhead. To understand the worst-case effect of this, we ran a write-intensive workload with minimal computation on a virtual disk. Table 1 shows that these worst case effects can be significant. However, because VMware ESX Server offloads the execution of the IO issuing path code, including the hash computation, onto idle processor cores, the actual IO throughput of this workload was unaffected. Backup data also tends to be well-structured and presented to the backup system in sequential streams, whereas live file systems must cope with random writes. Many CAS-based storage systems, including address data exclusively by its content hash. Write operations return a content hash which is used for subsequent read operations. Applying this approach to VM disk storage implies multi-stage block address resolution, which can negatively affect performance. Furthermore, since data is stored in hash space, spatial locality of VM disk data is lost, which can result in significant loss of performance for some workloads. DEDE avoids both of these issues by relying on regular file system layout policy and addressing all blocks by filename; offset tuples, rather than content addresses. DEDE uses content hashes only for identifying duplicates. Both NetApp's ASIS and Microsoft's Single Instance Store use out of band de-duplication to detect duplicates. The detected live file system in the background is similar to DEDE. SIS builds atop NTFS and applies content addressable storage to whole files. To using NTFS filters to implement file-level COW-like semantics. While SIS depends on a centralized file system and a single host to perform scanning and indexing, Farsite builds atop SIS to perform de-duplication in a distributed file system. Farsite assigns responsibility for each file to a host based on a hash of the file's content. Each host stores files in its local file system, relying on SIS to locally de-duplicate them. However, this approach incurs significant network overheads because most file system operations, including reads, require cross-host communication and file modifications require at least updating the distributed content hash index.

II. EXISTING SYSTEM

The data outsourced to a public cloud must be secured. Unauthorized data access by other users and processes (whether accidental or deliberate) must be prevented. Moreover; the probable amount of loss (as a result of data leakage) must also be minimized. A cloud must ensure throughput, reliability, and security. A key factor determining the throughput of a cloud that stores data is the data retrieval time. In large-scale systems, the problems of data reliability, data availability, and response time are dealt with data replication strategies. The existing data de-duplication schemes for primary storage, such as iDedup and Offline- De-dupe are capacity oriented in that they focus on storage capacity savings and only select the large requests to de-duplicate and bypass all the small requests. The rationale is that the small I/O requests only account for a tiny fraction of the storage capacity requirement, making de-duplication on them unprofitable and potentially counterproductive considering the substantial de-duplication overhead involved.

2.1 DRAWBACKS OF EXISTING SYSTEM

Placing replicas data over a number of nodes increases the attack surface for that particular data. For instance, storing m replicas of a file in a cloud instead of one replica increases the probability of a node holding file to be chosen as attack victim, from 1/ n to m /n, where n is the total number of nodes.

III. PROPOSED SYSTEM

A proxy-based storage system designed for providing fault-tolerant storage over multiple cloud storage providers. The system can interconnect different clouds and transparently stripe data across the clouds. Propose the first implementable design for the functional minimum-storage regenerating (FMSR) codes. Our FMSR code implementation maintains double-fault tolerance and has the same storage cost as in traditional erasure coding schemes based on RAID-6 codes, but uses less repair traffic when recovering a single-cloud failure. In particular, we eliminate the need to perform encoding operations within storage nodes during repair, while preserving the benefits of network coding in reducing repair traffic.

3.1 FMSR

FMSR (Functional Minimum-Storage Regenerating) codes provide the same fault-tolerance and bandwidth as traditional regenerating codes, but use less repair traffic by enabling uncoded repair, therefore typically saving money because the data transfers faster.

3.2 ADVANTAGES OF PROPOSED SYSTEM

The proposed CDC scheme ensures that even in the case of a successful attack, no meaningful information is revealed to the attacker. We ensure a controlled replication of

the file fragments, where each of the fragments is replicated only once for the purpose of improved security. To improving the performance of primary storage systems and minimizing performance overhead of de-duplication.

IV. SYSTEM ARCHITECTURE

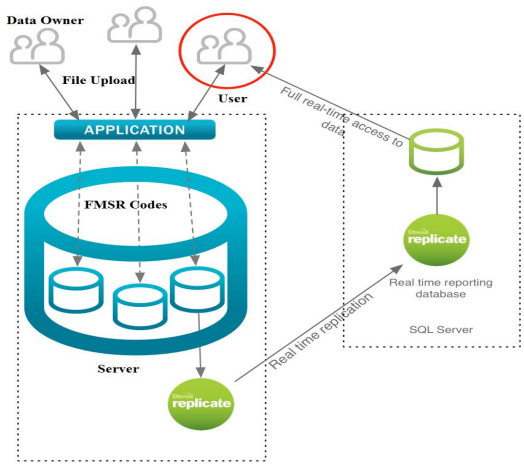

Fig.5.1 De-duplication Architecture.

V. MODULES

5.1 CLOUD ACCOUNT CREATION

The Login Form module presents site visitors with a form with username and password fields. If the user enters a valid username/password combination they will be granted access to additional resources. The login is categorized into three types

1) Data Owner
2) User

Data owner can upload the file. He/ She has the privileges to delete the uploaded file. User can perform keyword search on those files.

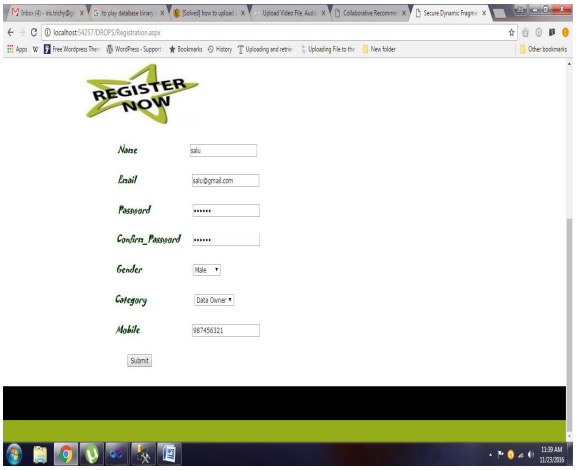

Fig.5.1 Cloud Account Creation.

5.2 DATA DE-DUPLICATION

Data De-duplication involves finding and removing of duplicate data's without considering its fidelity. Here the goal is to store more data with less bandwidth. Files are uploaded to the CSP and only the Data owners can view and download it. De-duplication is a technique where the server stores only a single copy of each file, regardless of how many clients asked to store that file, such that the disk space of cloud servers as well as network bandwidth are saved. For example, a server telling a client that it need not send the file reveals that some other client has the exact same file, which could be sensitive information in some case.

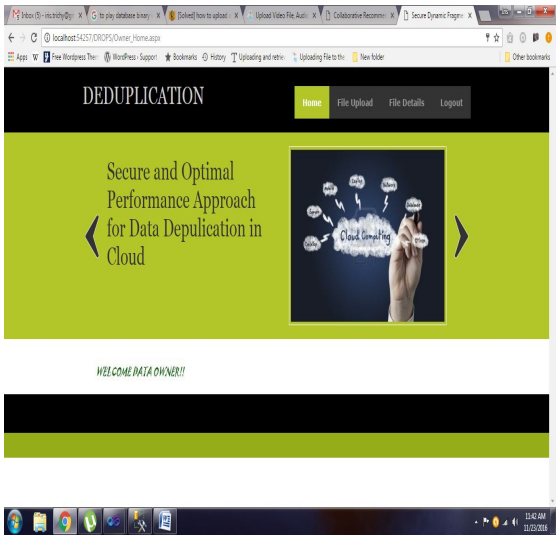

Fig.5.2 Uploading user data to cloud.

5.3 SELECT DE-DUPE

In this module, the uploaded file is encoded using Base64 Encoding, and the file content is stored in the server. Here the duplication check is processed, if the encoded file

content and the content already stored in the server matches, then the file is marked as duplicate and it tells the owner that the content already exists in the cloud server.

Fig.5.3 De-duplication of user data and creation of chunks.

5.4 ENCRYPTION & DECRYPTION

Encryption and decryption provides data confidentiality in de-duplication. A user (or data owner) derives a convergent key from the data content and encrypts the data copy with the convergent key. In addition, the user derives a tag for the data copy, such that the tag will be used to detect duplicates. Here, we assume that the tag correctness property holds, i.e., if two data copies are the same, then their tags are the same.

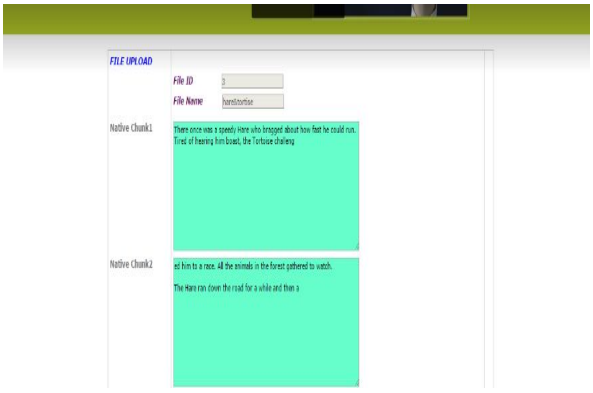

Fig.5.4 Encryption of user data using convergent key.

5.5 SWAP

Once the key request was received, the sender can send the key or he can decline it. With this key and request id which was generated at the time of sending key request the receiver can decrypt the message. Based on Access Monitor information, the Swap module dynamically adjusts the cache space partition between the index cache and read cache. Moreover, it swaps in/out the cached data from/to the back-end storage.

5.6 FILE UPLOAD& FRAGMENTATION

This module is available to the data owner category. In this module the data owner can upload the files. Each file has a unique id. The data owner should mention the name of the file while uploading. The file owner encrypts his files and outsources the cipher texts to the server. The server validates the outsourced cipher texts and stores them for the owner. The file owner can specify the fragmentation threshold in terms of either percentage or the number and size of different fragments.

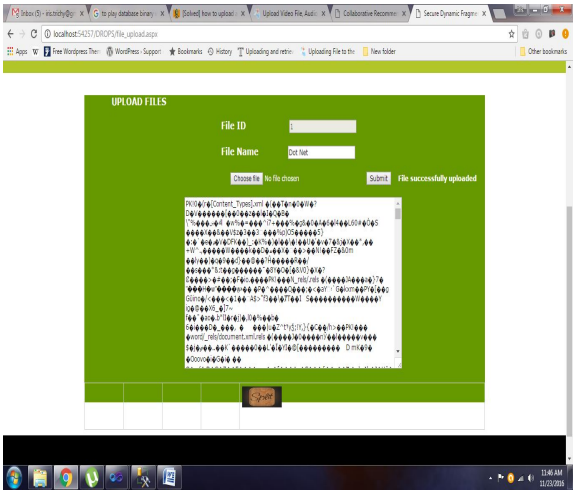

Fig.5.5 File uploading and fragmentation of user data.

VI. PSEUDOCODE

Input: base_chunk, src; input_chunk, tgt;
Output: output_chunk, otc;

```
function COMPUT(src,tgt)
        otc=empty
slink=Chunking(src)
        tlink=Chunking(tgt)
        sindex=INITMatch(slink)
        str=tlink;
        len=size(str)
while(str!=NULL)
do
    pos=FINDMatch(src,sindex,str,len)
    if(pos<0)
    then
        otc+=Instruction(insert str,len)
        output_chunk

else
     otc+=Instruction(copy pos,len)
```

```
      chunk
end if
str=(str->next)
len=size(str)
end while
end function

function INITMATCH(Slink)
          str=slink
          pos=0
          sindex=empty
     while(str!=empty)
      do
          f=dedup(key,str)
          sindex[hash[f]]=pos
          pos+=size(str)
          str=str->next
      end while
          return sindex
end function

function dedup (fileHashKey , data )
isUnique = false
if ( existInBloomFilter(fileHashKey ) )
if( !isDuplicateInCache(fileHashKey ) )
isUnique = true
 else
isUnique = true
 end if
if ( isUnique )
saveInCache ( fileHashKey )
sm.setBufferedData( fileHashKey , data )
end if
end function
```

VII. RELATED WORKS

7.1 SECURE DE-DUPLICATION WITH ENCRYPTED DATA FOR CLOUD STORAGE

ClouDedup, a secure and efficient storage service which assures block level de-duplication [1] data confidentiality at the same time using convergent key encryption [2] added with block level key management. Architecture of ClouDedup proposes to prevent well known attacks against convergent encryption by embedding a user authentication mechanisms and access control mechanisms. Thus, a server encryption is applied on top of convergent encryption performed by user. For each data segment a signature is linked to it, and need to be verified for retrieving it. To deal with block level key management a metadata manager(MM) has been added to architecture.MM uses file table- to store meta data about file, pointer table-to manage storage and a signature table- to store meta data about signature for meta data management.

7.2 I/O DEDUPLICATION: UTILIZING CONTENT SIMILARITY TO IMPROVE I/O PERFORMANCE

Koller R, et al. [2] explained about the duplication of data in storage systems is becoming increasingly common. We introduce I/O De-duplication, a storage optimization that utilizes content similarity for improving I/O performance by eliminating I/O operations and reducing the mechanical delays during I/O operations. I/O De-duplication consists of three main techniques: content-based caching, dynamic replica retrieval, and selective duplication. Each of these techniques is motivated by our observations with I/O workload traces obtained from actively-used production storage systems, all of which revealed surprisingly high levels of content similarity for both stored and accessed data. Evaluation of a prototype implementation using these workloads revealed an overall improvement in disk I/O performance of across these workloads.

7.3 DUPLESS: SERVER AIDED ENCRYPTION FOR DE-DUPLICATED STORAGE

DupLess Server-Aided Encryption for De-duplicated Storage provide simple storage interface. Cloud Storage provider like Dropbox, Mozy, and other providers can use de-duplication technology to save space by storing single copy of data. Message lock encryption is used to resolve the problem of clients encrypt their file however the saving is lock. Dupless is used to provide secure De-duplicated storage as well as storage resisting brute-force attacks. Clients encrypt under message-based keys obtained from a key-server via an oblivious PRF protocol in dupless server. It allows clients to store encrypted data.

7.4 WEAK LEAKAGE –RESILIENT CLIENT SIDE DE-DUPLICATION OF ENCRYPTED DATA IN CLOUD STORAGE

With the tremendous growth in data and number of users for cloud storage providers, data de-duplication becomes more important. De-duplication helps to store a unique copy of redundant data which results in reduce storage space and low bandwidth consumption. But the de-duplication result in new privacy and security challenges. We propose a new Secure Client Side de-duplication which provides secure de-duplication at client side with the help of convergent encryption and Merkle tree

7.5 A STUDY ON DATA DEDUPLICATION IN HPC STORAGE SYSTEMS

Kaiser J, et al. [4] created about the de-duplication is a storage saving technique that is highly successful in enterprise backup environments. On a file system, a single data block might be stored multiple times across different files, for example, multiple versions of a file might exist that are mostly identical. With de-duplication, this data replication is localized and redundancy is removed by storing data just once, all files that use identical regions refer to the same unique data. The most common approach splits file data into chunks and calculates a cryptographic fingerprint for each chunk. By checking if the fingerprint has already been stored, a chunk is classified as redundant or unique. Only unique chunks are stored. This paper presents the first study on the potential of data de-duplication in HPC centers, which belong to the most demanding storage producers.

VIII. CONCLUSION

In the analysis phase, the focus is on the data flow diagrams and system architecture and subsequently on further description of the proposed system. In design phase forms will be designed according to its function. Later on, we discuss about the system implementation and testing.

IX. FUTURE ENHANCEMENT

It is strategic to develop an automatic update mechanism that can identify and update the required fragments only. The aforesaid future work will save the time and resources utilized in downloading, updating, and uploading the file again. Moreover, the implications of TCP in cast over the proposed methodology need to be studied that is relevant to distributed data storage and access.

REFERENCES

[1] Pasquale Puzio, Refik Molva , Sergio Loureiro Cloudedup: Secured De-duplication with encrypted data for Cloud Storage. In *FAST'12*, Feb. 2012.
[2] Ricardo Koller, Raju Rangaswami: I/o de-duplication: utilizing content similarity to improve i/o performance**.** In *USENIX ATC'09*, Jun. 2009.
[3] Mihir Bellare, Sriram Keelveedhi, Thomas Ristenpart. Dupless: server aided encryption for de-duplicated storage. *ACM Transactions on Storage*, 10(2):1–22, 2014.
[4] Jia Xu, Ee-Chien Chang, Jianying Zhou. Weak leakage – resilient client side de-duplication of encrypted data in cloud storage. In *SOSP'95*, Dec. 1995.
[5] Kaiser J, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel. A Study on Data De-duplication in HPC Storage Systems. In *SC'12*, Nov.2012.