ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

An Efficient Deduplication Approach to Maximally Detect and Eliminate Data Redundancy in Cloud Environment

Rohini S Mahajan, Prof. Rugraj Rajpurohit

Department of Computer Engineering, Alard College of Engg and Mgmt., Pune, India

ABSTRACT: Due to explosive growth of digital data, it has been very important to manage the space efficiently at lower costs. So nowadays, co-operations have started implementing data deduplication techniques in their systems. Data deduplication is a technique of reducing storage needs by eliminating redundant data. To attain the high data deduplication, an efficient deduplication approach i.e. DARE, a low-overhead deduplication-aware resemblance detection and elimination scheme is designed.
In this system, we perform deduplication on cloud. Firstly we fragment the file and then apply AES encryption algorithm to encrypt the chunks. This provides security. SHA-256 is used to check duplicate files. DARE performs deduplication at three levels i.e. root level, block (chunk) level and then adjacent chunks are checked. The DARE use Duplicate Adjacency resemblance Detection (dupadj) scheme. It considers any 2 data chunks to be similar (i.e. candidates for delta compression), only the non-similar data will be stored as deltas.
Our system achieves a higher throughput, by using existing duplicate adjacency information for resemblance detection and detects maximum duplication on data.

KEYWORDS*:* Data deduplication, delta compression, SHA-256

I. INTRODUCTION

The redundancy of the data at the cloud storage is increasing. Thus exploiting the duplicate data can help in saving the space. It helps in reducing the time too required for transferring data in low- bandwidth network.Data reduction is the process of minimizing the amount of data that needs to be stored in data storage environment. Data deduplication has become an important and economic way to remove the redundant data segments, thus alleviating the pressure incurred by large amounts of data need to store. Fingerprints are used to represent and identify identical data blocks when performing data deduplication.

To address this challenge, data deduplication technique is preferred. Data de-duplication techniques are widely used by storageservers to eliminate the possibilities of storing multiplecopies of the data. De-duplication identifies duplicate dataportions going to be stored in storage systems alsoremoves duplication in existing stored data in storagesystems. Hence yield a significant cost saving.There are two methods available for duplication checking such as:

1) File level duplication check.
2) Chunk level duplication check.

In first method, only the file with same name are removed from the storage whereas in second, the duplicate chunks of same files are removed and stores only one copy of them.

In this paper, we propose DARE, Deduplication Aware Resemblance detection and Elimination scheme. DARE combines two techniques i.e. data deduplication and delta compression to achieve high data reduction efficiency at low overhead.

II. REVIEW OF LITERATURE

In [1], authors described a network storage system called as Venti, for data archiving. In the system, a unique hash of a block's contents was used as a block identifier for read and write operations. Their approach enforced a write-once policy, prevented accidental or malicious destruction of data.
In[2], authors introduced a new encoding scheme for large data sets: those that are too large to encodemonolithically.REBL is a combination of compression and duplicate. REBL also encoded similarly to a technique based on delta-encoding, reducing overall space significantly.
In [3], authors proposed a chunking algorithm, Frequency-Based Chunking (FBC),for data deduplication. The FBC algorithm explicitly made use of the chunk frequency information from the data stream to enhance the data deduplication.
In [4], authors represented chunking algorithm. In the deduplication technology, data is broken down into multiple pieces called chunks and every chunk can be identified using unique hash identifier. These identifiers can be used to compare the chunks with earlier stored chunks and varied for duplication.

Motivation

1. File-level chunking or whole file chunking considers an entire file as a chunk, rather than breaking files into multiple chunks. In this method, only one index is created for the complete file and the same is compared with the already stored whole file indexes.
2. The limitation of the file chunking and FSC is Boundary Shift Problem. It occurs during data modification. When we add new data or one byte to a file, all subsequent blocks in the file will be rewritten and changes.
3. The rewritten blocks are likely to be considered as different from those in original file, even though most of the data in the file are unchanged. This problem is known as the boundary shift problem.
4. FSC has a disadvantage of low deduplication efficiency caused by the boundaries-shift problem. For example, if we insert a byte at the beginning of the file, all the chunk boundaries marked by FSC will be shifted and duplicate chunks will be changed.
5. Hash algorithm leads to hash collision.

III. SOFTWARE REQUIREMENT SPECIFICATION

• Operating System	-	Windows XP/7.
• Programming Language	-	Java/J2EE.
• Software Version	-	JDK 1.7 or above.
• Application Server	-	Tomcat5.0/6.X
• Tools (IDE)	-	Eclipse.
• Scripts	-	JavaScript.
• Front End	-	JSP.
• Database	-	MySQL.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

IV. PROPOSED SYSTEM

A. SYSTEM ARCHITECTURE

In Fig.1 the User is responsible to select file or upload the data. The input can be file or image.

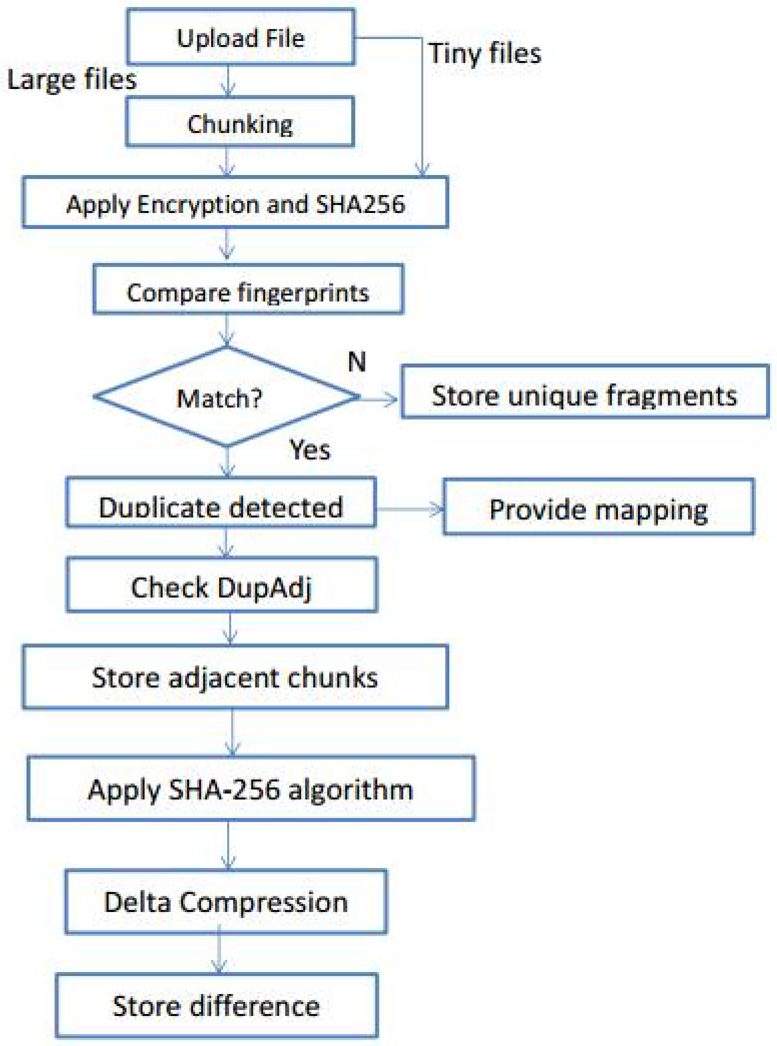


Fig.1 Proposed system flow diagram

Root level Deduplication:

At the root level, file level deduplication is performed. It checks its name and extensions. If it is notsame, it will be uploaded. Else, we cannot upload the file.
Once the file is uploaded successfully, chunk level deduplication will be performed.

Chunk level Deduplication:

The fragmentation algorithm splits the large file in small data chunks. If the file size if less than 4kb, it will not be fragmented. It will be considered as a chunk itself, instead file.

After fragments are formed, AES algorithm is applied to each chunk. This makes the chunk secure.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

We are using SHA-256 algorithm to generate fingerprint (hash value) for chunks. The reason for using SHA-256 hash function is that it isone way and collision-resistant hash function. The message digest size offingerprint produce by SHA-256 is 256 bit. SHA-256 generated fingerprint is used to detect duplicatechunks.

Duplicate adjacency:

To achieve data deduplication efficiency, fingerprints are compared. If fingerprint matches, data is considered as duplicate and will not be stored, instead link is provided.
If the fingerprint does not match with the existingfingerprint, then it is considered as the newfingerprint.
For non-duplicate chunks, our approach uses DupAdj i.e. it will check for adjacent chunks and store them in separate table. For those chunks, again SHA-256 algorithm will be applied.
The DupAdj detection: An input segment, it will traverse all the chunks by the aforementioned doubly-linkedlist to find the already duplicate-detected chunks.
For example consider the Fig.2:

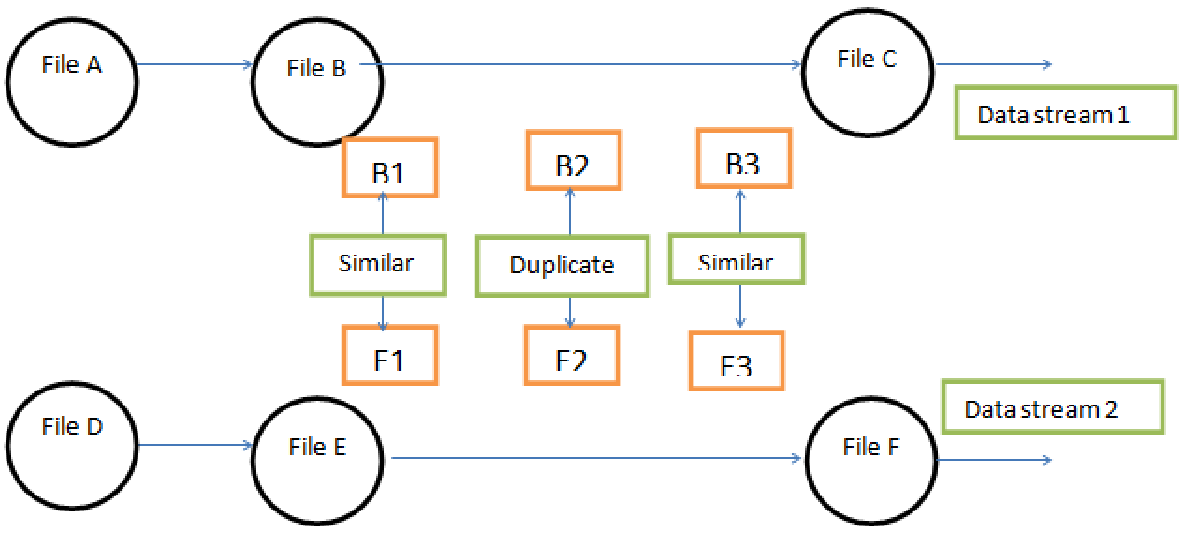


Fig.2 DupAdj Concept.

Fig.2 shows that if chunk B2 and E2 is duplicate, chunks around them will be considered potentially similar chunks(i.e.,B1&E1andB3&E3),and then migrate the similar chunks into one table in order to remove redundancy as much as possible.
If similar is detected, link will be provided, else will be considered for delta compression. Only will difference will be stored.

The non-similar data and deltas will be stored separately in tables.

Advantages of Proposed System

- Large-scale data reduction.
- Removing redundancy among information for likeness detection and finding the "sweet spot" for the super feature approach, similar data chunks has gained increasing attention in storage systems.
- Accurately detect the most similar candidates for delta compression with low overheads.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

B. ALGORITHM

Algorithm converting file into chunks
Input: File.
Output: Chunks.
1. If file is to be split go to step 2 else go to step 8.
2. Input is source path, destination path.
3. Size = size of source file.
4. Fs = Fragment Size.
5. NoF = number of fragments.
6. Fs = Size/Nof.
7. We get fragments.
8. End.

AES Algorithm:
The encryption process uses a set of specially derived keys called round keys. These are applied, along with other operations, on an array of data that holds exactly one block of data the data to be encrypted. This array we call the state array.Following are the steps of AES encryption for a 128-bit block:
1. Derive the set of round keys from the cipher key.
2. Initialize the state array with the block data (plaintext).
3. Add the initial round key to the starting state array.
4. Perform nine rounds of state manipulation.
5. Perform the tenth and final round of state manipulation.
6. Copy the final state array out as the encrypted data (ciphertext).

Secure Hash Algorithm (SHA-256)
The following steps are followed for SHA-256 algorithm:
1. Initialize hash values.
2. Initialize array of round constants.
3. Pre-processing.
4. Process the message in successive 512-bit chunks.
5. Compression function.
6. Add compressed chunk to the current hash value.
7. Produce the final hash value.

System Modules

The system contains following modules:
1. User.
2. Admin.
3. Cloud server.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

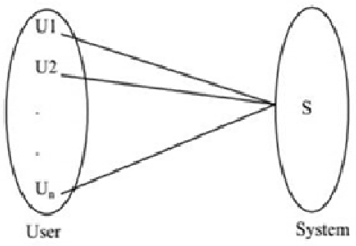
(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

V. MATHEMATICAL MODEL

A] MAPPING DIAGRAM



Where,
U1, U2….Un= Users.
S = System

B] SET THEORY

Login:
Let S be the data deduplication system.
S= {L, F, Enc(F), F_p}.
Where,
s = Start of the program.
Log in with webpage.
Where,
L = Login
UN = User name
PWD = Password
To access the facilities of system such as store and retrieve the files on cloud server, user has to log into system.

Chunking:
Before storing the files on cloud, files are divided into chunks.
The data chunking process is denoted by F.
F = {FC1, FC2….FCn}

Encryption:
Each chunk is encrypted before storing on private server by using AES algorithm to provide the security over data.
Enc (Fc) =CFC
Where,
C is the cipher text of chunk of file F, (FC).

Hash value generation phase:
SHA-256 hash function from family of cryptographic hash functions is implemented on the data chunks C.
F denotes chunks on which SHA-256 is applied.

So the set of fingerprint can be represented as.
Fp = {Fp1, Fp2... Fpn}
The duplicate chunks are identified on the basis of the fingerprint.

Deduplication Checking:
If H (New chunk) == H (Old n chunks [])
If the chunk is duplicate, do not store it instead provide link.
Else store it.

VI. CONCLUSION

DARE using DupAdj exploits the duplicate-adjacency information to find similar chunks data Delta compression is used to further store only deltas and utilize the storage space efficiently. DARE can be applied on files like word, pdf and .txt as well as images.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere and pro-found thanks to all our teachers Prof. Priyadarshani Kalokhe (HOD) and Prof. Archana Jadhav (PG Coordinator) and Prof. Rugraj Rajpurohit (Project Guide). I would like to thank our college for the boost that it has provided. I heartily thank for all their help and valuable time.

REFERENCES

[1] S. Quinlan and S. Dorward, “Venti: A new approach to archival storage,” in Proc. USENIX Conf. File Storage Technol., Jan. 2002, pp. 89–101.
[2] P. Kulkarni, F. Douglis, J. D. LaVoie, and J. M. Tracey, “Redundancy elimination within large collections of files,” in Proc. USENIX Annu. Tech. Conf., Jun. 2012, pp. 59–72.
[3] G. Lu, Y. Jin, and D. H. Du, “Frequency based chunking for data de-duplication,” in Proc. IEEE Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst., Aug. 2010, pp. 287–296.
[4] Venish and K. Siva Sankar, “Study of Chunking Algorithm in Data Deduplication ”
[5] Meister, J. Kaiser, and A. Brinkmann, “Block locality caching for data deduplication,” in Proc. 6th Int. Syst. Storage Conf., 2013, pp. 1–12.
[6] Proceeding of the International Conference on Soft Computing System, ICSCS , Volume 2.
[7] T. Meyer and W. J. Bolosky, “A study of practical deduplication,” ACM Trans. Storage, vol. 7, no. 4, p. 14, 2012.
[8] El-Shimi, R. Kalach, A. Kumar, A. Ottean, J. Li, and Sengupta, “Primary data deduplication-large scale study and system design,” in Proc. Conf. USENIX Annu. Tech. Conf., Jun. 2012, pp. 285–296.
[9] P. Shilane, M. Huang, G. Wallace, and W. Hsu, “WAN optimized replication of backup datasets using stream-informed delta compression,” in Proc. 10th USENIX Conf. File Storage Technol., Feb. 2012, pp. 49–64.
[10] R. C. Burns and D. D. Long, “Efficient distributed backup with delta compression,” in Proc. 5th Workshop I/O Parallel Distrib. Syst.,Nov. 1997, pp. 27–36.
[11] W. Xia, H. Jiang, D. Feng, and Y. Hua, “Silo: A similarity-locality based near-exact deduplication scheme with low RAM overhead and high throughput,” in Proc. USENIX Conf. USENIX Annu. Tech. Conf., Jun. 2011, pp. 285–298.
[12] F.Douglis and A.Iyengar, “Application- specific delta-encoding via resemblance detection,”in Proc.USENIX Annu.Tech.Conf., General Track, Jun.2003, pp.113–126.