Data Deduplication in Cloud Storage

Pawar P.R.
Computer Department, Dr.D.Y.Patil School of Engineering and Technology, Savitribai Phule Pune University, India.

Aarti Waghmare
Computer Department, Dr. D.Y.Patil School of Engineering and Technology, Savitribai Phule Pune University, India.

ABSTRACT
Cloud storage has become so popular in a now days not because of its cost efficient on demand usage but because of provision of huge amount of storage on which user can easily outsource its data. Most of the surveys told that nearly half of consumed storage on cloud is occupied by duplicate files or data. Challenge in front of today's cloud services is the management of that huge increasing quantity of data. To make data management scalable, de-duplication is introduced. In deduplication, instead of storing multiple copies of data which are exactly identical, it keeps only one physical copy. Security and privacy are among top concerns for the public cloud environments. This Data confidentiality for users is achieved with Convergent Encryption instead of Traditional encryption.

Keywords
Cloud Storage, Data deduplication, convergent encryption, Proof of Ownership, Data Confidentiality.

1. INTRODUCTION
Cloud computing is emerging day by day due to its ability to provide cost efficient and on demand use of huge storage as well as resources. As huge amount of growth of contents and resources online, cloud storage looks forward for better utilization of storage resources. It is needed to avoid large amount of redundancy in storage because of duplicate data [2].

For that reason in cloud storage systems, there is a requirement, for reducing the amount of data that is to be transferred or stored becomes an important, for providing benefits for performance of applications, also storage costs and administrative issues. As a result, Data De-duplication [2] plays important role as it saves cost for cloud storage, by storing only a single copy of redundant data, and provide pointers to clients of that duplicate or redundant copies instead of storing actual copies of that data.

In cloud computing to make data management scalable, deduplication [2] has attracted more and more attention recently in the backup process. Deduplication stores and transfers only a single physical copy of identical data, in this way de-duplication saves both disk space as well as network bandwidth.

For recent cloud storage services biggest challenge is to handle exponentially increasing data. Amount of data is expected to reach 40 trillion gigabytes in 2020 [5], according to the analysis report of IDC. And deduplication came for the rescue. Data deduplication is a one kind of specialized data compression technique [3]. This technique is used for better storage utilization and data transfers in various networks which can reduce the number of bytes to be sent. Deduplication eliminates redundant data as it keeps only one physical copy.

For that reason, if a user wants to outsource or upload a file or block of data, cloud service provider checks weather the file or data is already uploaded previously, if yes, cloud provider will update the owner list of that file by adding user to it. Deduplication provides high space and cost savings, so many cloud service providers begun to adapt it. De-duplication is a technique to reduce storage space and used for scalable data management.

There are two types of de- duplication one is file-level de-duplication and another is block-level de-duplication. Where File-level de- uplication refers to the whole file whereas block-level de-duplication refers to the fixed or variable size data block.

With the help of various encryption techniques, de-duplication can be made secure. In case of traditional encryption, different users encrypt their original file or copy of data with their private keys, so for identical data copies though their content is exactly same as plain text, ciphertexts generated from different users are not identical so deduplication can't be applied for traditional encryption techniques.

Convergent encryption [4] provides a solution to the issue raised by traditional encryption by implementing data confidentiality. Convergent encryption, a cryptosystem that produces same identical ciphertext files from the same plaintext files, no matter what their encryption keys are [15]. It encrypts/decrypts a data with a convergent key. From original data copy hash is computed and key is formed or generated [4]. After convergent key generation and data encryption takes place with those keys, users keep the keys and send the ciphertext produced to the public cloud. Due to convergent encryption, identical file copies will generate the identical convergent key also it will generate identical ciphertext. Now de-duplication can be applied on the ciphertexts. At the time of downloading, the ciphertexts are decrypted with their convergent keys.

The original data copy or file of user is first encrypted with a convergent key and this convergent key is again encrypted by a master key which resides locally by each user. The master key can be used to decrypt the encrypted convergent keys and hence the encrypted files. So, user is required to store only master key and data about those private keys known as metadata. Before going to implementation it is required to understand basic concept and various types of data deduplication.

2. DATA DEDUPLICATION
2.1 An overview
Data de-duplication has many forms. Different organizations may use multiple de-duplication strategies. It is very essential to understand the backup and backup challenges, when selecting de-duplication as a solution. Data de-duplication has three types. **Compression**, used frequently for long time. Lately, **single-instance storage** has enabled the removal of redundant files from storage such as archives. Most recently, it is clearly seen the introduction of **sub-file de-duplication**.

2.1.1 Data Compression

Data compression does not recognize or eliminate duplicate file it just compress the given file by reducing the size of files. Data comp1ression works within a file to identify and re move empty space that appears as repetitive patterns. That de-duplication is local to the file and remains independent of other files and data segments within those files. Benefits of Data compression are limited though it has been available for many years, it becomes isolated to each particular file. For example, data compression cannot identify and remove duplicate files, but will independently compress each of the files.

2.1.2 Single-Instance Storage

Single-Instance Storage removes multiple copies of any file. Single-instance storage (SIS) environments can detect and eliminate redundant copies of identical files. As name suggests it keeps only single Instance or copy of data and pointers are created for all other users who own the same file. In Single-instance storage systems, content of files are checked to determine if the file to be uploaded on cloud is identical to an existing file or not. The number of files that are stored as unique at cloud, on the basis of file content in Single Instance system, there may be large amount of redundancy in that file or files. For example, a new date inserted into the title slide of a presentation of some files, this is very small amount of change in huge files but considered as different files to be stored without further de-duplication.

2.1.3 Sub-file De-Duplication

If redundant data exists in separate files not needed to be identical files, that redundancy can be avoided with the help of Sub-file Deduplication. Sub-file de-duplication detects redundant data within and across files which is not the case in SIS implementations. Using sub-file de-duplication, redundant copies of data are found and are removed—even after the duplicated data exist, within non identical files. As a result, sub-file de-duplication eliminates the storage of duplicate data across an organization. Though files are not identical, Sub-file data de-duplication has large amount of benefits, have data elements that are already recognized somewhere in the organization. Sub-file de-duplication implementation has two types. **Fixed-length** sub-file de-duplication uses fixed length of data to search for the duplicate data within the files. Fixed-length segments are simple in design, but miss many opportunities to discover redundant sub-file data. For example, when name of person is added to certain file's tile page—the whole content of the file will shift, resulting the failure of the de-duplication to detect equivalencies.

Means, small change or addition in file may cause non equivalencies. **Variable-length** implementations are usually not corresponding to segment length. Variable-length implementations match data segment sizes to the naturally occurring duplication within files, vastly increasing the overall de-duplication ratio (In the example above, variable-length de-duplication will catch all duplicate segments in the document, no matter where the changes occur).

So data deduplication is widely used in most of the organizations, which is also called as, data reduction, single-instance storage, capacity optimized storage and intelligent compression.

and removes all but keeps one copy which creates pointers to the file so that users could access the file as and when needed. [5].

Pointers and Reference

Pointers and references are provided for other data owners which want to access same file on cloud which is already stored or previously stored. [6].

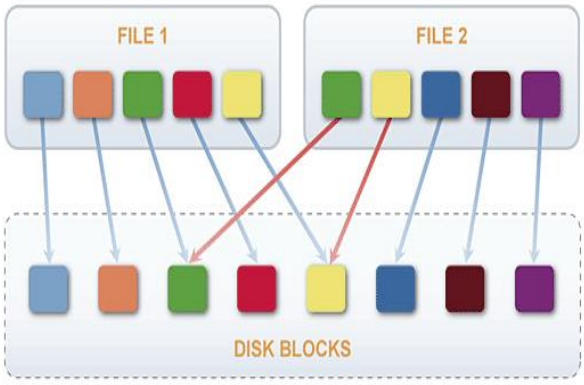

Fig. 1 Example of Data deduplication

Fig.1 shows de-duplication recognizes the redundant data

2.2 Data Deduplication Types

2.2.1. File-level de-duplication

File level deduplication checks certain attributes of given files against the specific index, if the file is unique, updating are done in It is commonly known as single-instance storage, if file is not unique then only a pointer corresponding to existing file that is stored references is updated. Only the single instance of file is saved instead of saving duplicate copies of files.

2.2.2. Block-level de-duplication

In Block-level data de-duplication, file is being divided into segments blocks or chunks, those chunks of data are checked for redundancies or previously stored information. The size of the chunk which needs to be checked varies from vendor to vendor. Each chunk is assigned with an identifier, by using hash algorithm for example – it generates a unique ID to that particular block. The comparison takes place between that identifier and particular unique Id. If ID is already present, it indicates that data is processed before. Therefore only a pointer reference is saved to the previously stored data. If the ID is new and does not exist, then that block is unique. The unique chunk is stored and the unique ID is updated in the Index. Some will have fixed block sizes, while some others use variable block sizes likewise few may also change the size of fixed block size for sake of confusing. Block sizes of fixed size may vary from 8KB to 64KB but the main difference with it is the smaller the chunk, than it will be likely to have opportunity to identify it as the duplicate data. If less data is stored than it obviously means greater reductions in the data that is stored. The only major issue by using fixed size blocks is that in case if the file is modified and the de-duplication result uses the same previously inspected result than there will be chance of not identifying the same redundant data segment, as the blocks in the file would be moved or changed, than they will shift downstream from change, by offsetting the rest of comparisons.

2.3 How deduplication works?

Data de-duplication compares the data i.e. usually blocks or files and eliminate the redundant data copies that are already

present in datasets. It removes the files that are not unique. The process consists of following steps

2.3.1. Firstly divide the input data into chunks or blocks.

2.3.2. Hash value for each of the block need to be calculated.

2.3.3. The values that are got is used to determine whether the blocks of same data is already stored.

2.3.4. Replace the redundant data with the reference or pointers to the block that is already in database**.**

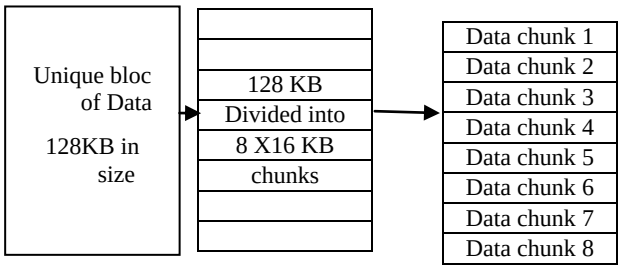

Fig. 2 Divisions of blocks according to chunks

As shown in figure 2 data is divided into data chunks. For every data chunks hash numbers are created as shown in figure 3. After the data is chunked, an index is created from the results, and the redundant data can be found and eliminated. Only a one copy of every chunk is stored in storage.

Once the data is divided into chunks, by the results that is obtained index can be created and the redundant data that will be found is removed. Only a single copy of every chunk will be stored. Data de-duplication process can be implemented in the number of different ways. The duplicate data can be eliminated b y simply comparing the files with each other and removing the data that is no longer needed and old**.**.

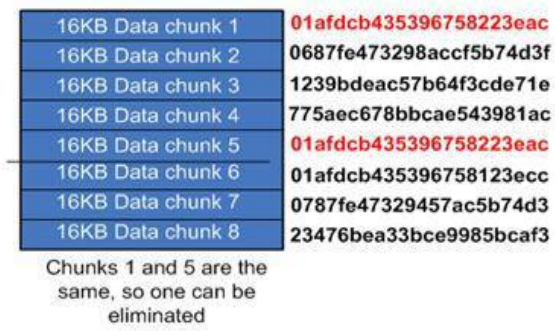

Fig.3 Hash number generation for data chunks

Once the data is divided into chunks, by the results that is obtained index can be created and the redundant data that will be found is removed. Only a single copy of every chunk will be stored. Data de-duplication process can be implemented in the number of different ways. The duplicate data can be eliminated b y simply comparing the files with each other and removing the data that is no longer needed and old.

Hash collisions are potential problem with de-duplication. The piece of data when it receives the hash number, the number is compared with the index of other already existing hash numbers.[8] Some of the algorithms can be used to find the hash numbers for example, SHA, MD5 [9] then it is encrypted using the AES algorithm and then data is out sourced to the cloud environment.[10]

3. Data confidentiality and depulication

3.1 Traditional Encryption

Data duplication, which has proved to be beneficial for data storage, some security issues arises due to it. As users sensitive data can be under attack of both insiders, person who belongs to same encryption, while providing data confidentiality, is incompatible with data deduplication [1]. Specifically, in case of traditional encryption different users encrypt data using their respective private keys [15]. Thus, even though files are exactly identical their ciphertext will be different and deduplication cannot be done as after traditional encryption generated files are not exactly same, it means no scope for deduplication.

3.2 Convergent Encryption

Convergent encryption [8] has been proposed to take care of data confidentiality while making deduplication possible and with feasible manner. A data copy is encrypted with a *convergent key*. That convergent key is formed by computing the hash value of original content of data copy. When key is generated from data copy and data is encrypted with that convergent key, users stores the keys and upload the generated ciphertext to the public cloud. There will be same convergent keys for identical data copies of files which generate same ciphertext, as encryption process is deterministic and it is derived from data content. In order to prevent unauthorized access, a secure protocol is also needed which is known as Proof of Ownership (PoW)[11]. In this protocol, user must provide the proof that he/she owns the same file when a duplicate file is found, on the basis of response/verify manner. After the proof is submitted, users are provided with pointer to access that file and no need to upload that file. Then with the help of that pointers user can download the file from public cloud or server, file downloaded is always encrypted and only data owner of that file can decrypt it with the help of respective convergent keys.

4. IMPLEMENTATION

Data confidentiality for users in process of deduplication is provided by Convergent encryption [4], [10]. A user derives a convergent key from original files and data copies which is hash either SHA1 or MD5 and encrypts the file or data with generated convergent key. User also derives a *tag* from original data copy; this tag will be used for duplicate check or detection. Clearly, if two data copies are the identical, it means their tags are identical. User first sends a tag to public cloud for checking whether the given tag is previously stored or not. Both convergent key and the tag are derived separately, and thus being independent, tag cannot be used to achieve the convergent key thus confidentiality of data remains uncompromised. The encrypted file or data copy and its tag will be stored on the public cloud or server side. For defining a convergent encryption scheme there are four primitive functions as:

- $KeyGen_{CE}(M) \rightarrow K$ this function is for generating convergent key form M which is original file or copy of data to a convergent key *K*;
- $Enc_{CE}(K,M) \rightarrow C$ this function symmetrically encrypts original file or data copy taking convergent key *K* and data M itself as parameters and ciphertext *C* is formed;

- Dec_{CE} $(K,C) \rightarrow M$ is this function uses the decryption algorithm that takes both the ciphertext C and the convergent key K as parameters and then recovers the original file or data copy M
- TagGen(M)$\rightarrow T(M)$ this is the tag generation algorithm, tag is generated with original file or data copy M .

4.1 Proof of ownership

With This protocol or (PoW) [11] enables users can give proof of their ownership on file or data copies to public cloud. PoW is implemented as follows, it has two modules which are implemented on interactive basis, prover and verifier. Prover is user and verifier is public cloud where data resides. The verifier derives a short value $\phi(M)$ from a data copy M. To prove the ownership of the data copy M, the prover needs to send ϕ' to the verifier such that $\phi' = \phi(M)$. The PoW normally based on the threat model, where an attacker knows who have or who owns file, he does not know the entire file. PoW ensures that user holds entire file rather than some information about that file. Some important requirements that constrain the solution in setting are discussed next:

4.1.1. Public hash function

To support various clients or users which are called as cross-user deduplication, for all clients there must use the same or common procedure for identifying duplicate data or files. Since at every client, procedure is implemented, it is public for attacker which can learn it easily. **Bandwidth constraints**: There should be enough bandwidth efficiency for protocol which will run between the client and server. Also, it must consume less bandwidth than the size of the data or file.

Server constraints: The server typically has to handle a Large number of files are to be handled by server in some Petabytes of data nearly. Server or public cloud stores very small data about each file in RAM, instead of storing entire files. Data or original files are stored on some secondary storage space having access time large. For every upload request, it is not affordable for server to retrieve the files each time from secondary storage. The solution is server or public cloud stores only that information which is useful to check correctness of the claim of user that they hold the file. For this server does not require to fetch the file from secondary storage for verification.

4.12 Client constraints

There should be clear proof form client that he holds the required file. This is challenging as the file may too large to fit in local memory for storage [14], [11].

4.13 Identification Protocol

It has two phases: first is Proof and second is Verify. In Proof, prover should submit some distinct attributes which only he can posses like credit card number etc. The input of the prover/user is his private key. This private key should not be shared with anyone else. The verifier which is public cloud can verify identity from input of public information related to *private key. T*hus verifier either accepts or decline and on that basis it is decided that proof is passed or not passed. Some identification protocols are certificate-based, identity-based identification etc. [12], [13].

5. CONCLUSION

Study of Data deduplication and its types can be very helpful for cloud storage, which need very convenient and cost efficient storage system. Notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. Several new de-duplication constructions are also presented and need of deduplication, various types of deduplication, how deduplication works, and how deduplication can be made compatible with encryption with the help of Convergent Encryption instead of traditional encryption where for identical data different ciphertexts are produced. Also, Focus is thrown on Proof of Ownership protocol for users to verify their identities. With all above mentioned entities one can implement data deduplication for duplicate check and PoW to prove user's identity using various functions across the public cloud environment. Due to deduplication, client is not charged for duplicate storage also security analysis demonstrates that proposed schemes are secure in terms of insider and outsider attacks specified in the proposed security model.

5. REFERENCES

[1] A Hybrid Cloud Approach for Secure Authorized Deduplication Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou *No. 6, June 2014*

[2] P. Anderson and L. Zhang, ''Fast and Secure Laptop Backups with Encrypted De-Duplication,'' in Proc. *USENIX LISA, 2010, pp. 1-8.*

[3] Secure Deduplication with Encrypted Data for Cloud Storage Pasquale Puzio (SecludIT, France & EURECOM, France), Refik Molva (Institut Eurecom, France), Melek Önen (EURECOM, France) and Sergio Loureiro (SecludIT, France), March, 2013.

[4] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer, ''Reclaiming Space from Duplicate Files in a Serverless Distributed File.

[5] Reclaiming Space from Duplicate Files in a Serverless Distributed File System, John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, Marvin Theimer Microsoft Research {johndo, adya, bolosky, dansimon, theimer}@microsoft.com

[6] *David Geer, "*Reducing *the* Storage Burden via Data Deduplication.computer.org, December 2008.

[7] http://www.computerworld.com/article/2474479/data-center/data- eduplication-in-the-cloud-explained--part-one.html

[8] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication", IEEE Transactions on Parallel and Distributed Systems,Volume:PP,Issue:99,Date of Publication :18.April.2014.

[9] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.

[10] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 491–500. ACM, 2011.

[11] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature

schemes. *J. Cryptology*, 22(1):1–61, 2009.

[12] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002.

[13] Shai Halevi1, Danny Harnik2, Benny Pinkas3, and Alexandra Shulman-Pel : Proofs of Ownership in Remote Storage Systems , eg2 1IBM T. J. Watson Research Center, 2IBM Haifa Research Lab, 3Bar Ilan University

[14] On Secure De-Duplication Using Convergent Encryption Key Management. *Ms. Madhuri A. Kavade1, Prof. A.C.Lomte2*