ISSN: 2057-5688

Privacy Enable block level Data Deduplication method for cloud data centers

[1]CHINNAMEDA NAGARAJU, [2]A. CHENNAKESAVAREDDY

[1]PG Scholar, Dept. of MCA, Newton's Institute of Engineering, Guntur, (A.P)

[2]Associate professor, Dept. of CSE, Newton's Institute of Engineering, Guntur, (A.P)

Abstract: The ever-expanding IT industry has resulted in a meteoric rise in the need for remote data storage. According to the EMC Digital Universe research 2012, global storage has already reached 2.8 trillion GB, and by 2020, it is expected to have increased to 52,47GB per user. Because clients upload data without being aware of the material accessible on the server, data redundancy is one of the primary issues in storage scarcity. The "National Survey on Data Centres Outages" conducted by the Ponemon Institute found 18% duplicate data. This problem is addressed by using the idea of data deduplication, in which each file is given a dynamically generated hash identification based on its contents. Clients are given a reference to the original file if they attempt to store a copy of one that already exists. As a result, data deduplication is useful for locating and eliminating duplicates of files stored in server farms. Because of this, several of the most well-known cloud storage companies, including Amazon, Google, Dropbox, IBM Cloud, Microsoft Azure, Spider Oak, Waula, and Mozy, have implemented data deduplication. In this research, we contrasted the standard practise of File-level deduplication with our own suggestion of Block-level deduplication for remote servers. We put the two deduplication methods to the test on a small dataset in-house and found that the suggested Block-level deduplication method outperformed the File-level method by 5%. In addition, we anticipate that taking into account a larger dataset with more users operating in a comparable area would further increase performance.

Keywords: DE deduplication, Data Centre, Cloud Computing, Cloud Server Provider.

I. INTRODUCTION

Because of its higher performance in compute, enormous storage, and web access, cloud computing is now the best option for establishing highly secure and scalable systems in any environment. Digital storage services are offered by several well-known organisations, like

Amazon, Apple, and Google, in the form of hardware, computational, and software resources with desirable characteristics such as flexibility, availability, and low cost. The EMC Digital Universe Study estimates that by 2020, the average individual would have access to 5247 GB of data, despite the fact that only 0.5% of that amount was analysed in any way in 2012.

Because customers upload data without verifying whether or not the identical file already exists, data redundancy is one of the primary issues in storage shortage. In their National Survey on Data Centre Outages [15], the Ponemon Institutes found that 18% of the data was redundant. To address the potential data storage shortage, cloud service providers are implementing many solutions all at once, including data deduplication. Data deduplication has been hailed as a potentially game-changing approach to reducing the burden on data centres while simultaneously increasing efficiency by eliminating redundant copies of data. To solve the duplicate data issue in data centres, a data deduplication system can find and remove duplicate copies of data. All authorised users will have access to the single, stored copy. The amount of space required for data storage and transmission may be drastically reduced by using this technology. As a result, numerous well-known cloud storage providers have started using data deduplication [10], including Amazon, Google, Dropbox, IBM Cloud, Microsoft Azure, Spider Oak, Waula, and Mozy.

The most typical method of data deduplication technology recognises duplicate files or blocks by hashing them using a cryptographic-hash string (SHA-1, SHA-256, etc.). When two files have the same contents, the hash algorithm will produce the same hash string.

The deduplication system will enable the user to upload a file once a cryptographic-hash value has been produced for it and stored in a record table.

Now, suppose another user also uploads the identical file and uses the hash function to get the crypto-hash value. The system now checks the newly received crypto-hash value against a stored hash table of previously analysed data blocks. If two users have the same cryptographic-hash key, the new user will be given access to the old file and the data file will not be saved to the storage facility. Therefore, by keeping just one version of each file in a data centre, a great deal of space may be saved.

II. METHODOLOGY OF THE SYSTEM

1. The System Model

Figure 4 depicts the overarching design of our proposed system, while the sub-diagrams below it show the various steps taken by the various components whenever a new client or uploader tries to save data on the server. After that, the new client goes through a series of steps that are represented graphically.

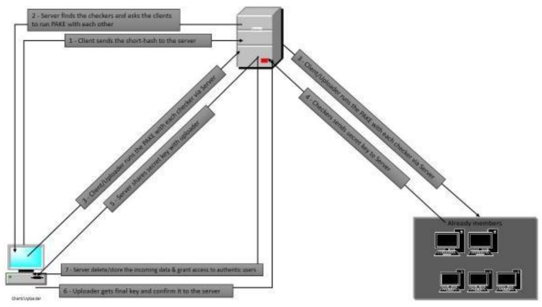

2 - Server finds the checkers and asks the clients to run PAKE with each other
1 - Client sends the short-hash to the server
3 - Client/Uploader runs the PAKE with each checker via Server
5 - Server shares secret key with uploader
3 - Client/Uploader runs the PAKE with each checker via Server
4 - Checkers sends secret key to Server
Already members
7 - Server delete/store the incoming data & grant access to authentic users
6 - Uploader gets final key and confirm it to the server
Client/Uploader

Figure 1: General architecture of our proposed system

2. Block-level duplication system design

Our Block-level deduplication methods offer a client-side, cross-user, and single-server deduplication scheme, with the former two ensuring the privacy of users' files while the latter preventing brute-force attacks common in proxies and additional servers by eliminating the need for them.

Furthermore, we have limited the number of times a user may try to upload a file by utilising the PAKE protocol, which deters malicious attempts from any curious client.

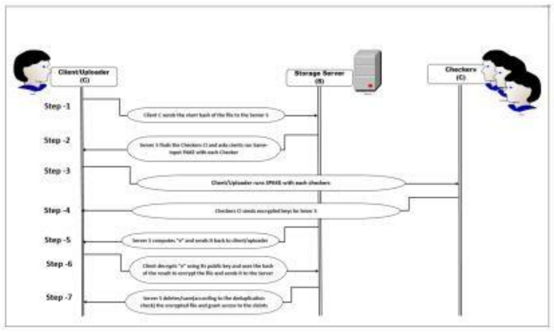


Figure 1: Interaction diagram for proposed deduplication model

III. SYSTEM MODULES

A. Eliminating Redundancies

File-level deduplication and block-level deduplication are the two major kinds of deduplication methods.

When doing deduplication at the file level, the system first calculates a hash value for the whole file and then queries a hash table to see if there are any duplicates. By contrast, Block-level deduplication is more space-efficient since it breaks the file into smaller pieces (called "chunks") and uses hash values to determine whether or not duplicates already exist in a hash table.

Deduplication may be broken down into two distinct types depending on its location: those that occur on the server and those that occur on the client. The client is not involved in the deduplication process in server-side deduplication, which occurs automatically once a file is uploaded to the system, but is actively involved in client-side deduplication, which begins with the client sending the server the hash value of the file to be uploaded so that the server can compare it to its own hashing table. If the server already has the file in its hash database, it will tell the client not to upload it and will provide them the key to access the existing file. Otherwise, the client will submit the whole file as a hash value. According to the research we just presented, client-side deduplication is the best option for reducing network traffic.

B. Types of Encryptions

The encryption method plays a crucial part in the deduplication system in guaranteeing the user's privacy at all times. Most of the studies cited in the aforementioned literature use a convergent file encryption technique. If two clients deliver the same file, the convergent approach will provide identical plaintexts, allowing us to identify the replicated file in the system. When we encrypt a file, it becomes illegible during transmission, but when it is received and decrypted, the original file is restored.

C. Cryptography: Hash Tables

It is assumed that two files are identical if the same cryptographic hash function is applied to them and they both generate the same output checksum. Hashing, also known as a cryptographic hash function, is an algorithm used to verify the authenticity and replication of data. Common cryptographic hashing methods include Message Digest 5 (MD-5), Secure Hash Algorithm 1 (SHA-1), and SHA-256; we will utilise SHA-1 throughout the study. To prevent any potential recurrence in hash keys, we have taken precautions while employing the short-hash (sh) cryptographic algorithm. Hashing, as contrast to encryption, is a one-way procedure in which we just construct a checksum that is representational of the file and cannot be reverse-engineered.

D. PAKE, or Password-Authenticated Key Exchange

PAKE is a well-known cryptographic mechanism that ensures the safety of the user's file and restricts access to the file to only authorised users. PAKE is a safe protocol even at low entropy, allowing clients to execute PAKE-protocol to exchange file accessing keys if the server discovers two similar short hashes. For instance, if a user is uploading a file and receiving a message from the server that the same file already exists in the storage after sending the hash value, the server will allow the new and old clients having the same file to exchange their private file key, for accessing the file, via the PAKE protocol. In order to swap access with the new user, the server needs the old client to be available in order to do the requisite number of checks. To prevent data from being lost, the technique employs a rate-limiting approach to reset the access key for new clients after a certain number of PAKE iterations. In order to maximise the efficiency of the deduplication process, the server additionally validates the short hash of the accessible data subset.

E. Rate Capping

Because a hacked server may operate as either an uploader or a checker to guess certain predictable files, its existence cannot be disregarded in the data centre's paradigm. The server may take on the role of either the uploader attempting to upload the file or the checker responding to PAKE requests from the uploader. As a result, we have suggested a "Rate Limiting Strategy" to restrict the uploader's PAKE runs and the checkers' PAKE answers in an effort to prevent brute force attacks. For each given file, there are two types of Rate Limits: RLu, which stands for the limit set for an uploader of type C, and RLc, which stands for the limit set for a checker of type Ci.

In this research, we found that keeping RLU = 70 but setting RLC = 30 was the optimal option for maximising security while minimising the system's overhead.

F. Threshold Randomization

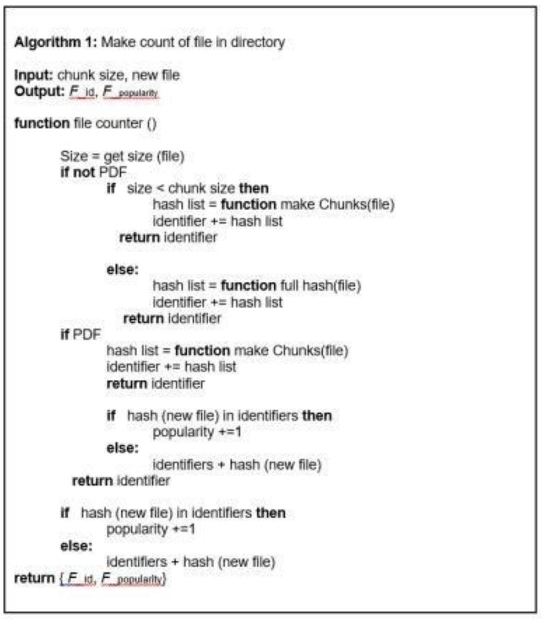
We propose the concept of a "Randomised threshold" to explain how to avoid having multiple copies of a recently added file using precious disc space and network traffic. The number of duplicate files in the system is calculated by comparing the random threshold TF (usually TF => 2) with the counter Tc for each file. Now, if TC > TF, server S notifies the uploader that the file is already present in the system and that further uploading the file is unnecessary, resulting in storage and bandwidth savings thanks to client-side de-duplication.

In the second scenario, server-side de-duplication maintains just storage space and requests an upload from the client if a duplicate file is detected but TC TF. However, Server S does not save the file since it is already accessible.

G. Checkerboard Choice

The server will select the checkers based on their popularity whenever it receives an upload request from the client in the form of a short hash (sh). The most popular client is the one who has performed the most checks on the new files. The checkers with the highest PAKE run engagement are then chosen by the server to act as checkers for the same file. In order to guarantee that PAKE is performed prior to uploading the file, the server will assign random checkers to the client if no matching file is detected.

In addition, we have outlined the functionality of some of our framework's essential features in the form of an algorithm, which we explain below. Using the relent function specific to the file type, the simulator in Algorithm 1 reads every file in the directory and its subdirectories, then calculates the matching crypto-hash.



```
Algorithm 1: Make count of file in directory

Input: chunk size, new file
Output: F_id, F_popularity

function file counter ()

    Size = get size (file)
    if not PDF
        if  size < chunk size then
            hash list = function make Chunks(file)
            identifier += hash list
          return identifier

        else:
            hash list = function full hash(file)
            identifier += hash list
          return identifier
    if PDF
        hash list = function make Chunks(file)
        identifier += hash list
        return identifier

        if  hash (new file) in identifiers then
            popularity +=1
        else:
            identifiers + hash (new file)
  return identifier

    if  hash (new file) in identifiers then
        popularity +=1
    else:
        identifiers + hash (new file)
return {F_id, F_popularity}
```

Algorithm 2 is a default function for generating 13-bits crypto hash through any hash function if file size is less then threshold value and file type is not PDF. (We used SHA-1 as hash calculating function)

IV. RESULTS AND DISCUSSION

As an open-source, object-oriented, and multi-paradigm language, Python3.0 was our first choice for use in the project's implementation. Python is the best option for any dataset project because of its extensive library, natural readability, and simplicity. PyPDF2, SciPy, NumPy, and Scilkit-Learn are just a few of the specialised Python libraries we've utilised for this project; together, they make it simple to analyse scientific data and give all the necessary modules for preparing the data. In addition, VMware Workstation Pro was set up as a virtual machine inside Windows 10 Pro for the sake of deployment.

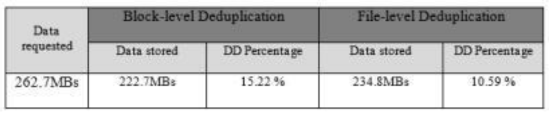
HP PO1 has an Intel(R) Core(TM) i7-7700 CPU running at 3.60GHz, 3600Mhz, 4 core(s), 16GB of RAM, and a 1000GB solid-state drive (SSD). Microsoft Windows 10 is loaded on this machine. In addition, Lubuntu (Lite Ubuntu) 18.04 LTS amd64 (desktop) has been set up on the embedded virtual machine's 64-bit architecture.

The media dataset and the corporate dataset were used to test the file-level deduplication model. However, we have built our simulation utilising data from a private local network that collects information on MS and PhD students and their academic pursuits. The aforementioned system may be found on the supercomputer at NUST Islamabad's Research Centre for Modelling and Simulations.

We have used the RCMS Super computer's 267 MBs dataset for our practical evaluation of the File-level and Block-level deduplication strategies. We used 1200 files spread over many directories and subdirectories to simulate File-level deduplication; 937 files were kept and the rest were discarded as duplicates.

Our simulator took in a total of 262.7 MBs of data during File-level deduplication, but the actual amount of data kept on the server was only 234.8 MBs. In a nutshell, we were able to keep our percentage of space savings at 10.5%, and our computed time per file processing was 3.2ms. In contrast, we had to implement some supplementary functions when conducting Block-level deduplication in order to separate files that were larger than the threshold value of 100 KB. We got 5700 chunked-files for storage, however only 4600 were really kept since the remainder were found to be duplicates. Our simulator received 262.7 MB of data for storage during Block-level deduplication, but the actual amount written to disc was 222.7 MB. In a nutshell, we were able to reduce our storage needs by 15.21% and process files in an average of 14.2ms (an increase in processing time was the result of additional calculation performed during file chunking).

By breaking down the massive files into manageable chunks of constant size, we were able to increase our deduplication efficiency from 10.5% to 15.21%. To further aid in displaying and analysing the performance of our suggested strategy, we have included visualising our findings in the next portion of our research.



Data requested	Block-level Deduplication		File-level Deduplication	
	Data stored	DD Percentage	Data stored	DD Percentage
262.7MBs	222.7MBs	15.22 %	234.8MBs	10.59 %

Figure 2: File-level Vs Block-level storage reduction in Local Network dataset

V. CONCLUSION

The primary motivation for the recent uptick in study on deduplication has been the need to save space, but the practise also raises a number of interesting questions about how best to optimise storage reduction, bandwidth utilisation, authentication, privacy, security, and availability of shared files. In this work, we provide a safe deduplication technique that saves as much space as feasible while maintaining the client's faith in the system. When a file exceeds a certain threshold in size, we use a crypto-hash algorithm to divide it into smaller pieces of the same size, which greatly increases our deduplication rate at the block level.

We found that our system outperformed the previously discussed framework by 5% based on our experimental results, and that our security model offers the same privacy and security measures to the user's data whether the data is being stored at the file level or the block level.

REFERENCES

[1] Gantz, John, and David Reinsel. "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east." IDC iView: IDC Analyse the future 2007.2012 (2012): 1-16.

[2] Dropbox hacked." http://www.businessinsider.com/dropbox-hacked 2014-10. Accessed on November 2014.

[3] Aldossary, Sultan, and William Allen. "Data security, privacy, availability and integrity in cloud computing: issues and current solutions." International Journal of Advanced Computer Science and Applications 7.4 (2016): 485-498.

[4] Sen, Jay dip. "Security and privacy issues in cloud computing." Cloud Technology: Concepts, Methodologies, Tools, and Applications. IGI Global, 2015. 1585-1630.

[5] Fan, Chun-I., Shi-Yuan Huang, and Wen-Che Hsu. "Hybrid data deduplication in cloud environment." Information Security and Intelligence Control (ISIC), 2012 International conference on. IEEE, 2012.

[6] Lillibridge, Mark, Kave Eshghi, and Deepavali Bhagwat. "Improving restore speed for backup systems that use inline chunk-based deduplication." FAST. 2013.

[7] Puzio, Pasquale, et al. "Clou Dedup: secure deduplication with encrypted data for cloud storage." Cloud Computing Technology and Science (Cloud Com),

2013 IEEE 5th International Conference on. Vol. 1. IEEE, 2013.

[8] Wen, Mi, et al. "BDO-SD: An efficient scheme for big data outsourcing with secure deduplication." Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on. IEEE, 2015.

[9] Harnik, Danny, Benny Pinkas, and Alexandra Shulman-Peleg. "Side channels in cloud services: Deduplication in cloud storage." IEEE Security & Privacy 6 (2010): 40-47.

[10] Prasadu Peddi (2023), Using a Wide Range of Residuals Densely, a Deep Learning Approach to the Detection of Abnormal Driving Behaviour in Videos, ADVANCED INFORMATION TECHNOLOGY JOURNAL, ISSN 1879-8136, volume XV, issue II, pp 11-18.

[11] Heen, Olivier, et al. "Improving the resistance to side-channel attacks on cloud storage services." New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. IEEE, 2012.

[12] Afreen Bari, Dr. Prasadu Peddi. (2021). Review and Analysis Load Balancing Machine Learning Approach for Cloud Computing Environment.Annals of the Romanian Society for Cell Biology,25(2),1189–1195.

[13] Liu, Jian, N. Asokan, and Benny Pinkas. "Secure deduplication of encrypted data without additional independent servers." Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015.

[14] Rohit Kiran, Rafael Fourq "SECURE DEDUPLICATION OF ENCRYPTED BLOCKS OF FILES WITHOUT INDEPENDENT SERVERS" International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835, Volume-3, Issue-10, Oct.-2016

[15] Prasadu Peddi (2021), "Deeper Image Segmentation using Lloyd's Algorithm", ZKGINTERNATIONAL, vol 5, issue 2, pp: 1-7.