SECURE STATIC DATA DE-DUPLICATION

[1]Rohit Pawar ,[2]Payal Zanwar, [3]Shruti Bora, [4]Shweta Kullkarni
Dept. of computer Engineering,Sinhgad College of Engineering, Savitribai Phule Pune University Pune
Email:[1]Payal.zanwar15@gmail.com, [2]rohitpawar00748@gmail.com, [3]shrutibora2010@gmail.com
[4]svkulkarni1993@gmail.com

***Abstract* - Data de-duplication is a technique used to improve storage efficiency. In static data de-duplication system, Hashing is carried out at client side. Firstly hashing is done at file level. The de-duplicator identifies duplication by comparing existing hash values in metadata server. If match is found, then logical pointers are created for storing redundant data. If match doesn't exist, then same process is carried out at chunk level. Duplicated data chunks are identified and only one replica of the data is stored in storage. Logical pointers are created for other copies, instead of storing redundant data. If it is a new hash value, it will be recorded in metadata server and the file or corresponding chunk will be stored in file server and its logical path in terms of logical pointers is also stored in metadata server. Basically static de-duplicator is implemented with three components: interface, de-duplicator and storage. Interface carries out hashing of uploaded file and interfaces client with de-duplicator. After receiving hash value, de-duplicator carries out its function as mentioned above. The last component storage consists of file server and metadata server. Thus, de-duplication can reduce both storage space and network bandwidth.**

Keywords – SHA-Secure Hash Algorithm, TTTD-Two Threshold Two Divisor

I.INTRODUCTION

As information is important in many organizations, techniques involving processing and management of data have gained popularity. Most of the data in large databases is redundant. Storage of redundant data leads to wastage of storage space as well as network bandwidth

Data de-duplication technique can be used to reduce cost associated with storing and managing data as decreasing the amount of data equals reducing a lot of costs for storage, power, and bandwidth. The concept of data de-duplication technology is to identify redundant data, and store only once and return pointer to the copy so that users can access the data when needed. All the details of processing are hidden from users and applications as the information can be retrieved as it is. There are three types of data de-duplication

A. *File-level De-duplication*

In File-level de-duplication it searches for same file and check whether that file is already exist in cloud storage or not? It searches file by its content not by name due to this even though the filename is different we can identify the duplication of file. We store only one copy of same file and if we found the file is already exists then we provide ‘pointer’ to existing file. In this way we avoid redundancy in cloud storage. Advantage of using this method is it is very easy to implement we don’t need very complex

computation and algorithm. Major disadvantage of using this method is even a single minor change to the file will result in an additional copy being stored which leads our de-duplication system less efficient.

B. Byte-level De-duplication

In byte-level de-duplication we divide our file data into bytes of data and we check that byte is already stored on cloud storage or not? This scheme provides greater de-duplication because we can reduce lot of storage space but problem with this method is we need to maintain too much meta data for single byte data .Moreover in this scheme we need lot of time to divide file into bytes and lot computation time to check byte already exist or not? To avoid this type of problem we go for block level de-duplication.

C. Block-level De-duplication

In Block-level de-duplication we first divide the entire file into different blocks. After that we check whether that block is already stored on cloud storage or not? If block is already exist then we will not store that block again onto cloud storage hence we avoid redundancy and increase storage efficiency. In this case we return memory pointer of already existing block so that it can be refer by the another user. In case block is already not exist on cloud storage then we store that block on cloud storage. Process of dividing file data into different block is called as 'chunking'.

Two types of Chunking:

1. Fixed block level chunking

In fixed block level chunking method chunk boundaries are predetermined due to this it divides the file into fixed size block. Let's look this concept into more detail using figure 1.

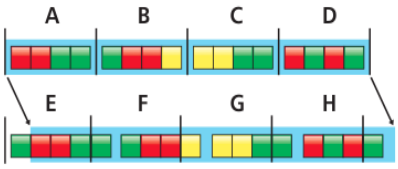

Figure1: Fixed block level chunking

In figure1 we divide our file into fixed length block A, B, C, D. Assume we add new byte before block A due to this every block shifted one byte so that we will get new block E, F, G, H. This is major drawback of this approach so we use variable block level chunking.

2. Variable block level chunking

In variable block level chunking method chunk boundaries are not predetermined due to this it divides the file into variable size block. Chunk boundaries are determined based on the content of files because of this it evenly distribute block and it more resistant for insertion and deletion. Let's look this concept into more detail using figure2.

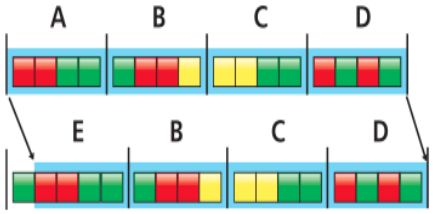

Figure2: Fixed block level chunking

In figure2 we divide our file into block A, B, C, D. Assume we add new byte before block A so we increase boundary of block A so that other blocks will not get affected. So now result of this only Block A content changes so it treat as new block E but all other block B, C, D are not affected.

From this we can conclude that variable block level chunking method is best suitable method for de-duplication scheme.

II. BACKGROUND WORK

Before our proposed system, attempts were made to develop techniques to reduce storage which include standard file-compression tools, such as LZ77 and LZ78. The purpose of these techniques was to find short repeated strings inside a file whereas the purpose of data de-duplication is to find large scale data in entire files or large sections of files – that are identical, in order to store only one copy of it.

III. PROPOSED SYSTEM

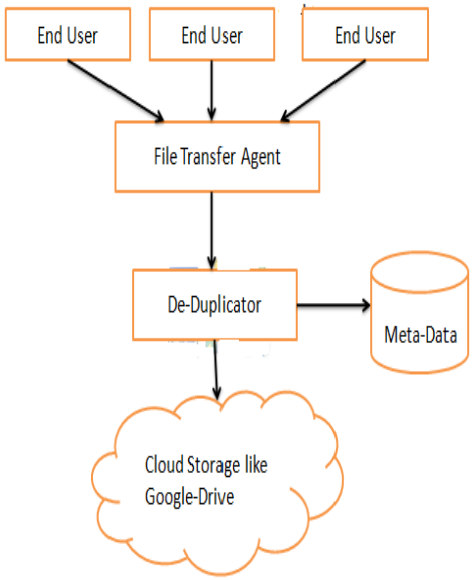

Figure3: Data De-duplication system architecture

The proposed data de-duplication system is divided into 3-tier architecture including presentation tier, business logic tier and database tier.

Presentation tier: Presentation tier contains a number of end users such as desktop machines, laptops, mobiles etc and file transfer agent carries out tasks of uploading, downloading and deleting files.

Business logic tier: Business logic tier consists of Data de-duplicator. Detailed working of data de-duplicator is explained in next section.

Database tier: Database tier consists of cloud storage such as Google-Drive, Drop box etc which is used to store uploaded files. Metadata is used to store hash value and all other data required for reconstitution of file.

IV. DETAILED WORKING OF DATA DE-DUPLICATOR ENGINE

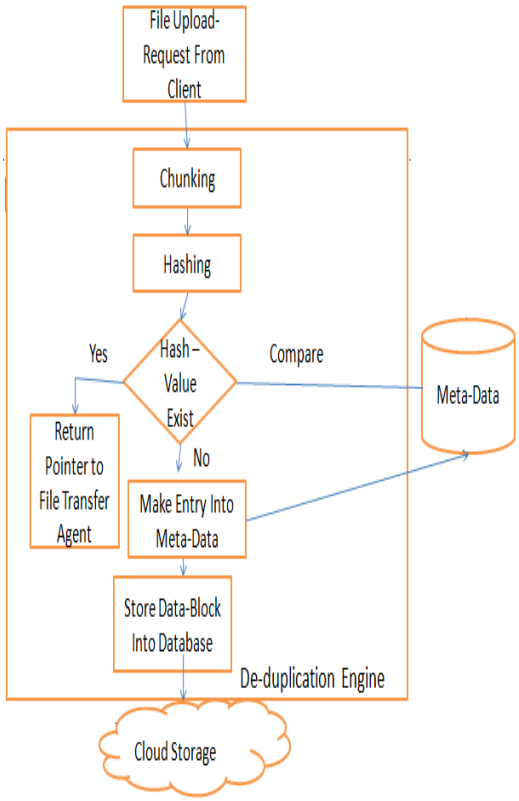

Figure4: Detailed working of data de-duplicator

Whenever client send file upload request then we will check for file level de-duplication using file attributes such as date when file created, date when file modified and file size. This file attribute we compare with file attribute stored in meta-data if the file attribute of current file is matched with file attributes in meta-data server then we conclude that entire file is duplicated so we will not do further process.

In case entire file is not duplicated then we will go for block level de-duplication in that we perform chucking means we are dividing our content of file into different block using Two Threshold Two Divisor (TTTD) algorithm.

Steps in TTTD algorithm are:

1. Start from minimum threshold to calculate hash value.

2. If match found then that is the breakpoint for that text file and thus that is the chunk.

After chunking we will perform hashing in that we calculate hash value for our chunked block using Secure Hash Algorithm (SHA-2).

Steps in calculating hash value using SHA-2 are:

1.Append padding bits to message with 1 followed by no. of 0s to make its length 896 bits and length of original message is appended as 128 bits.
2. Initialize hash buffer (512bits).
3. Process message in 1024-bit block.

Then we send hash value to de-duplication engine. It compares that hash value with hash table in meta-data. If the hash value is matched then we conclude that block is already stored into cloud storage so we will not store that block again. If hash value is not matched then we will store that block and make entry into meta-data. We repeat this process for every chunk.

V.SECURITY MODEL

Now a day's security becomes very important aspect while storing data on cloud storage so it is necessary to provide security to our data. In our de-duplication security model we use AES (Advanced Encryption Standard) for encryption and decryption of that data block.

Advantages of AES Encryption Algorithm

- AES improves security as well as performance in various smartcards and hardware device implementation.
- Till this time no non-brute force attacks against AES is found as its federal information processing standard.
- AES is stronger and powerful algorithm because of this US security agency use AES for secret information exchange.

Figure5 shows our proposed upload security model.

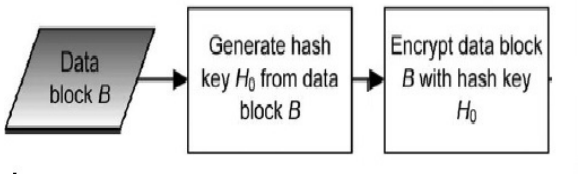

Figure5: De-duplication Upload security model

In upload security model we generate hash for data block then we encrypt that data block using hash value of that data block.

Figure6 shows our proposed download security model.

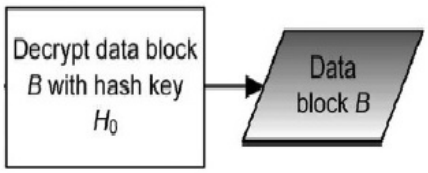

Figure6: De-duplication Download security model

In download security model we search into metadata for hash value of data block. Once we get hash value of our data block we decrypt our data block using hash value.

VI. FUNCTIONALITY

Upload File:

When client sends an upload file request then file is transferred to middleware i.e. to de-duplicator then de-duplicator engine divide the entire file into different blocks of data and generate hash value using SHA-512 for each block and check that hash value is already exist or not into metadata. If hash value is already exist into metadata it means that block already exist so no need to store that block again on cloud storage hence we avoid redundancy and we increase reference count of that block by 1. In case block already not exist we store that block on cloud storage and make reference count as 1.While uploading a block on cloud storage we encrypt our data block using hash value of our data block so that we maintain security of data block. In order to improve efficiency and speed we use Queue mechanism so that multithreading mechanism is possible. For faster comparison of hash value we use Indexing mechanism so time required for comparison is less. With Queue and Indexing mechanism we improve efficiency and speed of uploading file.

Download File:

When client sends "download file" request then download file request transfers to middleware i.e. to de-duplicator then it download all block from cloud storage then it looks for hash value of that block into metadata and retrieve that hash value and decrypt that block using hash value send that block to client. After successfully

download of data block we check integrity of downloaded file with uploaded file so that we come know whether we get original file or altered file. We us fc (file compare) utility for windows user or diff (difference) utility for Linux user to check whether file contents are altered or not reconstitute properly. Due to this mechanism we can provide better quality of service (Qos) mechanism to end user.

Delete File:

When client sends "delete file" request then check number of references to blocks of that file if there exists more than one reference then decrement reference count by one, keep common chunk as it is and delete other chunks if reference count is one. Means only one user is associated with that file so delete all chunks associated with that file.

VII. BENEFITS

Reduction in storage, bandwidth and cost: Storage-based data de-duplication reduces the amount of storage needed.

Reduction in bandwidth: Network data de-duplication is used to reduce the number of bytes that must be transferred between endpoints, which can reduce the amount of bandwidth required.

VIII. CONCERN

Data loss: In data de-duplication, data is not stored as it is but is stored after some transformations which may cause data loss. So users are concerned about maintaining integrity of their data.

Data corruption: As data de-duplication uses cryptographic hash functions, concern arises about data corruption. But as we are using powerful hash algorithms, there is very negligible probability of data corruption.

IX.EXPERIMENTAL RESULT

Our project tested on only single de-duplicator engine and following Execution Environment

1. Windows 7(64 bit)
2. Intel Dual core processor(2.2GHz)
3. 3 GB RAM
4. JDK 7.0

So following result may vary depending upon Execution Environment.

Table I shows operation performed and time elapsed without Multithreading and Queue

File Type	***Size***	***Upload***	***Download***	***Delete***
Text	922B	1 s	1 s	1 s
Text	26.4KB	2 s	1 s	1 s
PDF	172KB	21s	1 s	10s
PDF	1.62MB	2m:47s	3 s	1m:35s
MP3	3.92MB	7m:15s	5 s	3m:39s

Table I: Operation and time required without multithreading and queue

Table II shows operation performed and time elapsed with Multithreading and Queue

File Type	***Size***	***Upload***	***Download***	***Delete***
Text	922B	1 s	1 s	1 s
Text	26.4KB	2 s	1 s	1 s
PDF	172KB	9s	1 s	6s
PDF	1.62MB	1m:2s	2 s	45s
MP3	3.92MB	3m:14s	3 s	1m:39s

Table II: Operation and time required with multithreading and queue

By comparing table I and table II we conclude that with multithreading and queue we are able to reduce half of our execution time .Due to queue mechanism we are able to get acknowledgment of operation which helps us to improve Qos(Quality of Service) mechanism.

X. FUTURE WORK

In order to increase the density of data chunks, we need centralize lengths of data chunks. This can be done using TDSW –which can make distribution of chunk sizes more stable and achieve a smaller unbiased variance. It is hard to obtain parameters used in TDSW theoretically; work on this problem may help in obtaining better chunks, thus a better system.

Also, there are still many related problems and issues needed to be studied for data de-duplication system. For example, the hashing-based data de-duplication systems have to face the natural limitation-the hash collision problem. These related topics are also worth working on in the future.

XI. CONCLUSION

Our proposed system provides an efficient storage scheme for file storage. With the help of data de-duplication, storage cost and network bandwidth usage can be considerably reduced. In brief, the TTTD algorithm, not only successfully achieves the significant improvements in running time and average chunk-size, but also obtains the better controls on the variations of chunk-size by reducing the large-sized chunks. Our project focuses on the chunking algorithm. Moreover, we have also provided a way to index the metadata to speed up the lookup table searching. Finally, the security issue of the hashing-based data de-duplication is also handled. We have successfully implemented AES (Advance Data Encryption) standard for encryption and decryption of data to maintain security of data block.

XII. ACKNOWLEGMENT

This work is sponsored by Persistent System Private Limited (PSPL), Pune, Maharashtra,India. Guidance provided by Mr.Vinod Hire, IBM Module Leader,PSPL and Mrs.A.R.Joshi ,Assistant Professor ,Sinhgad College of Engineering, Pune,Maharashtra ,India.

XIII. REFERNCES

[1] “A Running Time Improvement for Two Thresholds Two Divisors Algorithm” by BingChun Chang,*San Jose State University,2009.*

[2] Waraporn Leesakul, Paul Townend and Jie Xu “Dynamic Data Deduplication in Cloud Storage”, 2014 IEEE 8th International Symposium on Service Oriented System Engineering.

[3] Dependent chunking for differential compression, the local maximum approach nikolaj bjørner, andreas blass, and yuri gurevich

[4] A Framework for Analyzing and Improving Content-Based Chunking Algorithms Kave Eshghi (kave.eshghi@hp.com) Hsiu Khuern Tang (hsiu-khuern.tang@hp.com) Hewlett-Packard Laboratories Palo Alto, CA February 25, 2005

[5] “Cryptography and Network Security”, 5th edition by William Stalling.