www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 3 March 2015, Page No. 10715-10720

Data DeDuplication Using Optimized Fingerprint Lookup Method for Cloud Storage Using Android

Prof. Sulbha Ghadling ,Piyush Mehta, Piyush Oswal, Rohit Padwal, Amit Mohol

[1]Prof. sulbha Ghadling
sulbhaghadling@gmail.com

[2] plot no 6/25 flat no 1, lakshdeep society, shantai nagari phase 2,
,talegaon dabhade, pune 410507, India
piyushmehta2810@gmail.com

[3]724, shala chowk, talegaon dabhade,pune 410 506, India
Oswalpiyuush10@gmail.com

[4] adat/post-chakan , ambethan chowk, ganesh nagar,
pune 410501,India
padwalrohit522 @gmail.com

[5]Pune University, Computer Engineering,
At/post – pawananagar, Tal-maval, Dist-pune 410406, India
amitmohol003@gmail.com

Abstract: *Today we all keep hearing about how fast data is being generated and also how important this data is to different organizations and enterprises. Hence organizations all around the globe are looking forward to storing this data in an efficient manner to avoid data loss and also to store all important information in a compressed and secure manner. Data Deduplication is an effective emerging technology that provides a solution to store more information in less space. As the advantages of offsite storage, less maintenance of storage infrastructure and cost benefits are very useful, huge enterprises choose to back up their data on cloud services. Data DeDuplication strategy aims at exploiting the fact that redundancy between two different types of data sets is minimal. By treating each application differently and storing the related information separately the overall disk usage can be improved to a great extent. This is done by implementing adaptive chunking, hashing and storing of different application separately. Data Conscious deduplication is observed to thus give a greater deduplication factor. It not only improves the deduplication process but also gives good bandwidth saving and efficient resource utilization essential for cloud storage.*

Keywords: Cloud; Chunking; File Recipe Compression.

1. Introduction

Data deduplication or Single Instancing essentially refers to the elimination of redundant data. As the amount of digital information is increasing exponentially, there is a need to deploy storage systems that can handle and manage this information efficiently. Data deduplication is one of the emerging techniques that can be used to optimize the use of existing storage space to store a large amount of data. Basically, data deduplication is removal of redundant data. Thus, reducing the amount of data reduces a lot of costs storage requirements costs, infrastructure management costs and power consumption.

1.1 Motivation

There is a huge amount of duplicate or redundant data existing in current storage systems, especially backup systems. There are numerous techniques for reducing redundancy when data is stored or sent. Recently, data de-duplication, the new emerging technology motivated by this scenario of data expansion, has received a broad attention from both industries and academics. Data de-duplication, also called intelligent compression or single instance storage, eliminates redundancy caused by identical objects which can be detected efficiently by comparing hashes of the objects content. Storage space requirements can be reduced by a factor of 10 to 20 or even 90 when backup data is de-duplicated.

Client based deduplication processing can reduce the amount of data being transferred over the network to the storage system, but there are often requirements for extra CPU and disk I/O processing on the client side. Clients might be constrained to file level visibility of the data, which can reduce the granularity of the deduplicated component analysis. Also, clients might only have visibility to a limited pool of data, which can impact the duplicate frequency of the pool and reduce deduplication efficiency.

1.2 Background

Data deduplication is the new data compaction technology which eliminates redundant copies of data. The deduplication process consists of data chunking, creating smaller unique identifiers called fingerprints for these data chunks, comparing these fingerprints for duplicates and storing only unique chunks. Data deduplication partitions input data stream into smaller units, called chunks and represents these chunks by their fingerprints. After chunk fingerprint index lookup, it replaces the duplicate chunks with their fingerprints and only transfer or stores the unique chunks.

Efficiency of any data deduplication is measured by the

- Dedup ratio (size of actual data / size of data after deduplication) and
- Throughput (Megabytes of data deduplication per sec).

Data deduplication can be either client (source) based deduplication and server (target) based deduplication, depending on where the processing occurs. Source deduplication eliminates redundant data on the client side and target deduplication applies data deduplication at the server side. Client based deduplication provides huge savings in network bandwidth but it often requires extra CPU processing for intensive hash calculations and extra disk I/O processing for comparing the fingerprints. Server based deduplication examines the incoming data to find duplication. It allows you to deduplicate data at scheduled time but it requires extra CPU and disk I/O processing on the server side.

Deduplication can be categorized into inline deduplication and post process deduplication depending on when the redundant data is eliminated. Inline deduplication processes data before it is written to disk. An advantage of inline deduplication is that duplicate chunks are never written to the disk. Post process deduplication refers to handling processing after the data has been written to the disk, generally with scheduled or manual runs. Deduplication can be either file-level or block-level deduplication. In file based deduplication, each chunk is a single file and hence it compares files for duplicity. Since the files can be large it can adversely affect both the deduplication ratio and the throughput. In block based deduplication, the data object is chunked into blocks of either fixed size or variable size. Fixed size chunking breaks the entire file into fixed size blocks from the beginning of the file. It is easier to implement and it requires less compute power and provides higher deduplication throughput.

1.3 Need

- When properly implemented, data deduplication lowers the amount of storage space required, which results in less disk expenditures. More efficient use of disk space also allows for longer disk retention periods, which offers better recovery time objective (RTO) for a longer time and reduces the need for tape backups.
- Data deduplication also reduces the data that must be sent across a WAN for remote backups and replication saving sufficient amount of bandwidth which can be use to sent some other data.
- Different applications and data types naturally have different levels of data redundancy. Backup applications generally benefit the most from de-duplication due to the nature of repeated full backups of an existing file system.
- Different applications and data types naturally have different levels of data redundancy. Backup applications generally benefit the most from de-duplication due to the nature of repeated full backups of an existing file system.

2.1 Related Work

There have been many solutions to the data compression scenario in the form of different deduplication software. Each method had its own benefits and drawbacks. Following are the deduplication strategies that we have studied and come up with new and improved enhancements to them.

2.2 Demystifying Data Deduplication

The research deals with studying and examining various factors that affect data deduplication effectiveness. The task being data intensive and at a cost of higher resource overhead it states that proper deduplication technology implementation can give 30 percent more storage savings and a 15 times better CPU utilization.

1. Deduplication solutions differ along three key dimensions, namely: Placement of the deduplication functionality client side or server side.

2. Timing of deduplication with respect to the foreground IO operations- synchronous or asynchronous.

Algorithm used to find and reduce redundancies in the data- inter file or intra file granularity.

2.2.1 AA-Dedup :An Application-Aware Source Deduplication Approach for Cloud Backup Services

AA-Dedup design is inspired by the following four observations of deduplication for cloud backup services in the personal computing environment: 1. The majority of storage space is occupied by a small number of compressed files with low sub-file redundancy. 2. Static chunking method can outperform content defined chunking in deduplication effectiveness for static application data and virtual machine images. 3. The computational overhead for deduplication is dominated by data capacity 4. The amount of data shared among different types of applications is negligible. These observations reveal a significant difference among different types of applications in terms of data redundancy, sensitivity to different chunking methods and independence in deduplication process. Thus, the basic idea of AA-Dedup is to effectively exploit this application difference and awareness by treating different types of applications differently and adaptively during

the deduplication process to significantly improve the deduplication efficiency and reduce the overhead.

2.2.2 Two Threshold Two Divisor (TTTD)

Chunking algorithms play an important role in data de-duplication systems. The Basic Sliding Window (BSW) algorithm is the first prototype of the content-based chunking algorithm which can handle most types of data. The Two Thresholds Two Divisors (TTTD) algorithm was proposed to improve the BSW algorithm in terms of controlling the variations of the chunk-size. The maximum and minimum thresholds are used to eliminate very large-sized and very small-sized chunks in order to control the variations of chunk-size. The main divisor plays the same role as the BSW algorithm and can be used to make the chunk-size close to our expected chunk-size. In usual, the value of the second divisor is half of the main divisor. Due to its higher probability, second divisor assists algorithm to determine a backup breakpoint for chunks in case the algorithm cannot find any breakpoint by main divisor.

2.2.3 File Recipe Compression in Data Deduplication Systems

The most common approach in data deduplication is to divide the data into chunks and identify the redundancy via fingerprints of those chunks. The file content can be rebuilt by combining the chunk fingerprints which are stored sequentially in a file recipe. The correspondingly recipe data can occupy a significant fraction of the total disk space, especially if the deduplication ratio is very high. The paper proposes a combination of efficient and scalable compression schemes to shrink the file recipes size. A trace-based simulation shows that these methods can compress file recipes by up to 93 %.

The idea of file recipe compression is to assign (small) code words to fingerprints. The code word is then stored instead of the fingerprint in the file recipe. Since file recipes can be responsible for a significant fraction of the physical disk usage of deduplication systems, these results enable significant overall savings.

2.2.4 Application-Aware Local-Global Source Deduplication for Cloud Backup Services of Personal Storage

ALG-Dedup, an Application-aware Local-Global source deduplication is a scheme that improves data deduplication efficiency by exploiting application awareness, and further combines local and global duplicate detection to strike a good balance between cloud storage capacity saving and deduplication time reduction. It combines local and global deduplication to significantly reduce the disk index lookup bottleneck by dividing central index into many independent small indices.

3 Literature Review

3.1 Secure Hash Algorithm(SHA1):

Sha1 algorithm is an iterative algorithm which produces one-way hash functions that can process a message to produce a condensed representation called a message digest (Hash function). This algorithm enables the determination of messages integrity: any change to the message wills, with a very high probability, result in a different message digests. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits). This algorithm can be described in two stages: preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest. The length of message digest produced by the sha1 algorithm is 160 bit and it gets change every time whenever there is even a small change in the document producing a whole new hashes value. Hence it can be used for checking the presence of any particular file or data block in the storage system whether it is present or not. SHA 1 maybe used to hash a message, M, having a length of L bits, where $1 < L < 264$.

The algorithm uses

1.a message schedule of eighty 32 bit words,
2.five working variables of 32 bits each, and
3.a hash value of five 32-bit words. The final result of SHA 1 is a 160-bit message digest

The words of the message schedule are labeled W0, W1, and W79. The five working variables are labeled a, b, c, d, and e.

3.2 Two Threshold Two Divisor-Switch (TTTDS)

TTTDS is a variable length content defined chunking algorithm. Chunking is the process of partitioning the entire file into small pieces of chunks. TTTDS algorithm is used to segment the input file into chunks. For any data deduplication system, chunking is the most time consuming process as it requires traversing the entire file without any exceptions. The process time of chunking totally depends on how the chunking algorithm breaks a file. Whatever is the metadata scheme used the cost of the metadata and the process of hashing depends on the number of chunks generated for a file. The smaller the chunk size better is the deduplication. However large number of chunks leads to higher metadata overhead as each chunk needs to be indexed, also processing cost increases. On the other hand large chunks can reduce ones chances of identifying duplicate data. TTTDS applies a minimum and a maximum threshold when setting the boundaries of every chunk. If the boundary condition using one of the divisors evaluates to be true, this boundary is held as a backup. This is done in case the next boundary condition, using the second divisor, does not evaluate to be true for a long time resulting in a large chunk. Therefore, it can avoid small chunking and large chunking.

3.3 Rabin Fingerprinting

Traditionally, a hashing function is used to map a character string of undetermined length to a storage address. The mapping is done by computing the residue of that string, viewed as a large integer, modulo p - a prime number. Since this is a mapping from an unbounded domain into a finite range of integers, it would be very difficult to devise a good hashing function that will produce a set of distinct mapped values with minimal chance of collisions. This is especially true when the set of character strings to be mapped is of a huge size. In 1981, Michael O. Rabin published a paper describing a finger-printing method using polynomials. It is claimed that the method provides efficient mapping that has little chance of collision even with huge dataset. Rabin fingerprinting is a very efficient technique for generating hashes. It is based upon the idea of representing variable length byte strings using fixed size fingerprints .i.e. hashes. We can apply this for generating fingerprints for variable length chunks formed during the storing process.

3.4 Data DeDuplication Algorithm

1. Accept file for chunking.

2. Perform file size filtering.

3. if(filesize< 8KB)

4 . {

5. perform wholefile chunking();

6. }

7. elseif(file extension==.txt)

8. {

9. perform variable chunking();

10. }

11. else

12. {

13. perform static chunking();

14. }

15. Calculate hash if(filesize< 8KB)

16. {

17. calculate Rabinhash();

18. }

19. elseif(file extension==.txt)

20. {

21. calculateMD5Hash();

22. }

23. elseif(file extension==.pdf)

24. {

25. calculate SHA 1Hash();

26. }

27. Update Database

28. if(Duplicate Status == 1)

29.{

30. Delete duplicate chunk();

31. }

32. Send unique chunks and related information.

33. Select file to download.

34. Download file.

35.Stop.

4 Proposed Work

4.1 Problem Statement

Data conscious deduplication using optimized finger print look up methods for
cloud storage services.

4.2 System Features

- The various features included in the system are -
- File Filtering
- Intelligent chunking.
- Adaptive Hashing(Finger print Generation)
- Storing Finger print and other chunk details into database
- Duplicate chunk identification.
- Storage and maintenance of chunked data
- Retrieval of chunks and generation of original file.

4.3 What is to be developed

4.3.1 Client(Source) Deduplication

Deduplication in the client follows a client-server approach wherein a deduplication client communicates with its server counterpart installed on either a deduplication appliance or the Storage Array itself. This type of deduplication is often referred to as transmission deduplication where in, the

deduplication client processes the data, determines if duplication exists and the deduplication client would transmit only the unique data over fiber channel or the IP network.

Deduplication at the Client provides huge savings in network bandwidth. However, these savings come at a cost. Firstly, client side CPU and IO resources are used for detection of duplicates. Secondly, it has a wide range of security implications, since any client can query the Storage Array with deduplication metadata, one could attempt to reverse engineer the system to determine the contents stored by other backup clients on the same storage array. Lastly, client-side deduplication may affect the choice of deduplication timing at the server.

4.3.2 Server

The server stores unique chunks sent by the client and reconstructs the file when the client demands based on the information stored in database and makes it ready for download. Later it stores the file into the directory client demands.

4.4 Goals

- To have Reduced computational overhead(Adaptive processes based on nature of data stream)
- Increase the overall efficiency(Disk utilization is improved with sorted database)
- Reduce backup window time.
- Less storage space
- Save Cloud cost.
- Achieve Energy efficiency.

4.5 Constraint

- Our project concentrates on de-duplication of just text and pdf files.
- We shall not be considering the de-duplication of data present in music, video or other files.
- The effectiveness of data transfer taking place between all the entities depends upon MTU (Maximum Transmission Unit) which is not considered in our project. It mainly depends upon the medium of connection and it's totally a user'schoice.

Compressed code word generation is a future enhancement.

5 Research Methodology

5.1 Research Methodology

Research Methodology is a collective term for the structured process of con- ducting research.

5.2 Input For The System

Clients selects input files for storage in the system and the type of these file should be either .txt or .pdf as we are implementing our project only for the .txt and .pdf files but in the future the scope of the project can be increased to the other document format as well as audio and video file format.

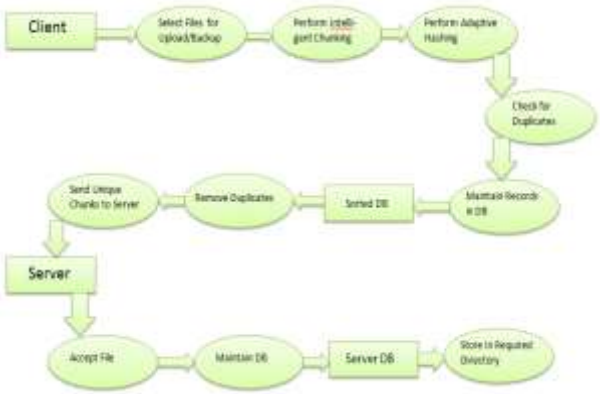

Figure 1: Work Flow Diagram (Part A)

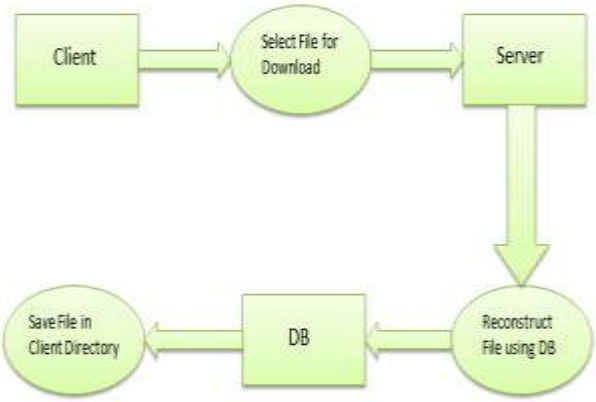

Figure 2: Work Flow Diagram (Part B)

6 Project Design

6.1 System Architecture

The System Architecture is divided in four main components: -

1. Clients
2. De-Dup Server
3. Meta-Data Server
4. Cloud Storage

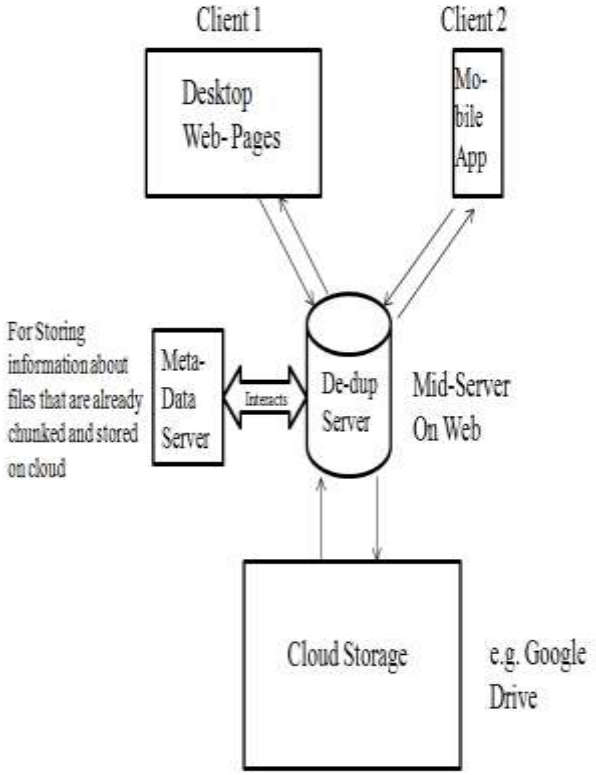

Figure 3: System Architecture

1. Clients : -

Multiple Clients are able to upload files to the cloud storage. Clients can login through web servers and select the files the they want to upload in cloud storage. Multiple Clients can login in common server so that they can access files and update them accordingly.

2. De-Dup Server : -

Once the client has uploaded the file, it first goes to De-Dup Server where all the chunking, hashing, assembling algorithms are stored on the Server. Once the files are sent to server chunking algorithm comes into effect and chunks the file as per the code. These data is stored in Meta-Data Server for future references. Then the chunks are uploaded to Cloud Storage as individual Chunk Files. When the Client asks for the file stored on cloud, De-Dup Server interacts with Meta-Data Server to collect the information of the chunks made of the file and then sends request to cloud storage for downloading the file. Once the chunk-files are downloaded the server assembles the chunk-files into one file and then sends it to the client.

3. Meta-Data Server : -

Meta-Data Server interacts with De-Dup Server and stores all the information about the file that is chunked and stored in file. The number of chunks that were made of the file, Name of the chunks and the names of the chunk files that are stored on cloud. When the file is to be downloaded from Cloud De-Dup Server collects the chunk information from Meta-Data Server so that proper file can be downloaded.

4. Cloud Storage : -

Cloud Storage (e.g. Google Drive) are used so that the files can be stored and accessed all over the Web.

6. Conclusion

In Data Duplication the file is first scanned then the file is upload .The file is check already present or not in server. The file present then the file is directly stored. If not present then the file is chunked using TTTDs Algorithm. i.e. File, Block, Variable, Fix Chunking. Each and every chunk has unique hash code using hash Algorithm. The hash code is in binary code format. Each and every hash code is compare with stored chunk in database. The hash code is present then the this chunk will be discarded. If not present then the hash code will be stored. If file is give for client then file chunk will be downloaded in proper sequence and reassemble it. The security is important . Any one can access it they will give just binary code.

7. Acknowledgements

We would like to express our sincere thanks to our guide, Prof. Sulbha Ghadling, Assistant Professor at the Department of Computer, Nutan Maharashtra Institute of Engineering and Technology and our beloved Sir Prof. Shyamsunder B.Ingle HOD of Computer Dept, for the support and motivation while carrying out the work presented here and also in writing this paper. We thanks our principal Dr. R.D.Kanphade for providing the necessary environment and facilities to conduct our project work.

References

1. Mark W. Storer Kevin Greenan Darrell D. E. Long Ethan L. Secure Data Deduplication Miller, Storage Systems Research Center University of California, Santa Cruz 2013

2. IBM System Storage N series Data Compression and Deduplication, 2012

3. Jan Stanek, Alessandro Sorniottiy, Elli Androulakiy, and Lukas Kencl A Secure Data Deduplication Scheme for Cloud Storage, Czech Technical University in Prague, 2011

4. Peter Christen and Karl Goiser, Quality and Complexity Measures for Data Linkage and Deduplication, 2013

5. TCS Hi-tech white paper: Effective Data Deduplication Implementation, 2012