Data Deduplication for Efficient Cloud Storage and Retrieval

Rishikesh Misal and Boominathan Perumal
School of Computer Engineering, Vellore Institute of Technology University, India

Abstract: *Cloud services provide flawless service to the client by increasing the geographic availability of the data. Increasing availability of data induces high amount of redundancy and large amount of space required to store that data. Data compression techniques can reduce the amount of space required for that data to be store at various sites. Data compression will ensure that there is no loss of availability and consistency at any site. As there is huge demand for cloud services and storage due to this the amount of investment also increases. By using data compression we can reduce the amount of investment required and this will also decrease the amount of physical space and data centers required to store data. Various security protocols can be incorporated to secure these compressed files at various sites. We provide a reliable technique to store deduplicates and its management in a secure manner to accomplish high consistency as well as availability.*

Keywords: *Data deduplication, cloud computing, storage, file system, distributed system.*

Received February 23, 2017; accepted June 13, 2017

1. Introduction

Data compression deals with reducing the size of data with minimal interference with the actual data. Data compression is a field of information theory which has made its mark because of its need for high storage space [19, 24]. Data compression will ensure less usage of storage spaces but will also enhance performance [9]. Data compression can be applied to various types of files and sizes. Section 2 gives a survey on the literature used to comply this method including methods of data de-duplication. Section 3 explains the proposed methodology giving all the information on the process of data upload to data compression to data retrieval. Section 4 is about the results and performance achieved by the method with a pictorial view of the result and its detailed explanation. Section 5 is on the conclusion we could draw from our method and a future work discussion.

Cloud Computing has become more popular due its ability to provide services on the go (portability) and its high scalability to the user requirement. To be robust in its working, cloud services always have a backup [17] of all the data that is present (user/system data) [15]. But this also creates a problem of running out of storage space. Data files vary in sizes and many multimedia files are always large (few Megabytes to several Gigabytes). This becomes a trouble for the cloud service provider [15]. Data Deduplication can reduce the overhead of data storage logistics caused by these large to very large files. Cloud computing always focuses on utilizing the most of all the resources, but this always comes at a cost which is growing very fast day by day. It is not possible to reduce user demand and/or reduce the price required to buy more resources. Data compression will allow files to be uploaded on the large storage [19] at a reduced size this will ensure fewer amounts of resources to be bought and maintained at a given period [16].

Deduplication [17, 21] is removing the redundancy in the data across files/users/blocks of data. We can remove all the redundancy and store that chunk of data only once rather than ‘n’ number of times inside the data. The original file can anytime be reconstructed by the adding the redundant data in its original place. This will ensure that less space is required to store data and high bandwidth optimization. This will also increase the availability and consistency of data across various geographical sites [12].

Files being divided into chunks [4] can lead to many security issues such as master file owner determination, wrongful access of chunk of a file etc. The article by Puzio *et al*. [20] viz., PerfectDedup: Secure Data Deduplication has showed a way to perform data deduplication using a secure method [10, 17, 25, 27]

Data Deduplication can be implement in a distributed environment [3]. Distributed environment can also make way for user level data deduplication with some restrictions [21]. The restrictions may include the file permissions, file types and critical application/users for which the files are used [17]. The article by Luo *et al*. [14] viz, Boafft: Distributed Deduplication for Big Data Storage in the Cloud gives a better way to implement data deduplication for distributed environment [13, 21, 26].

In deduplication the primary goal is to find redundant data inside many files as a chunk of fixed/variable size. This becomes the key element of the implementation of the idea. The searching of this redundant chunk is trivial but time consuming. In order to save less amount of space and decrease the Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) we tend to tradeoff some computational time. The amount of computational time desired to perform this task has to be finite as the data present in the world is too high to even estimate its size. We need to optimize the way these deduplicate chunks are searched and stored. This will help us determine the correct size that is needed to store the file and it becomes easier to retrieve it. It is also important to reduce the number of I/O operations required [6] to rebuild the file once it was reduced to chunks during retrieval. We have used limited I/O calls to retrieve back the original file with a tradeoff for a mapping table just like a cache memory in hardware architecture.

In distributed system environment it is quite apparent that the data gets replicated in order to have higher availability and this is applicable only if the data is allowed to be shared around different users at the same time. But the extent of replication is a tradeoff between high availability and consistency [1] as explained by Kleppmann in [12]. Our method is specific to a user but it can be extended to different users in a data sharable scenario. Our proposal for data deduplication is proven to work on personal desktop environments instead of cloud storage [9]. In Douceur *et al*. [5] have proven that on an average a regular desktop storage has nearly half of the data duplicated without a purpose. Here in this case a data deduplication model will not only help reduce the data redundancy but also will be able to track files and increase efficiency.

2. Literature Survey

2.1. Data Deduplication

Data deduplication is the elimination of redundant data within an environment. Data deduplication or single-instance storage technique which reduces storage needs by eliminating redundant data [18, 29]. Only one unique instance of a file is stored in the data center for multiple users having the same file [18, 23, 24, 29].

This file is then linked to as many users which had the file. For example, user A, B, C have the same .mp3 file of 100MB so instead of storing it at three different sites and occupying 300MB data essentially it is preferred to have a single copy since all the three files are the same. Data deduplication checks for such redundant files and removes all the copies and only keeps the master copy and provides a link to all the users' sites. No user knows whether the .mp3 has been shared by many users or not. Deduplication is kept transparent from the users and every user thinks it has received its own copy [7, 9, 10, 11].

What can be achieved by data deduplication?

- Elimination of redundant data within an existing environment.
- Bandwidth optimization.
- Disk space storage savings (CAPEX).
- 50:1 or 20:1 ratio for storage saving.

Where does redundant data reside?

- Within files.
- Across files.
- Across Applications.
- Across clients.
- Over time (time residing in the data center after many updates).

2.2. File Level Deduplication

Consider there are three users Bob, Rob and Matt they all have three identical .mp3 files with them and they want to store those files on the cloud. Bob, Rob and Mat name the file 12.mp3, 13.mp3 and 14.mp3 respectively. In file level deduplication since all the three files are identical it only stores one file named 1X.mp3 and creates link to all the users. Figures 1 and 2 Gives a pictorial representation of the description.

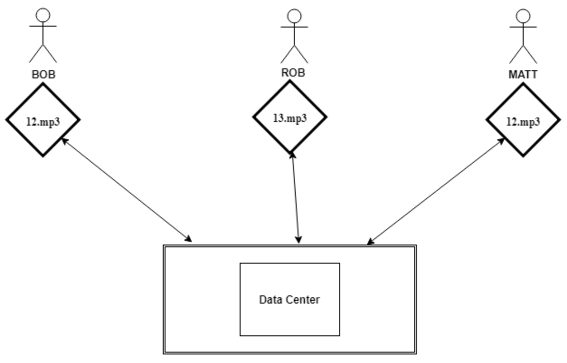

Figure 1. File uploading phase for file level Dedup.

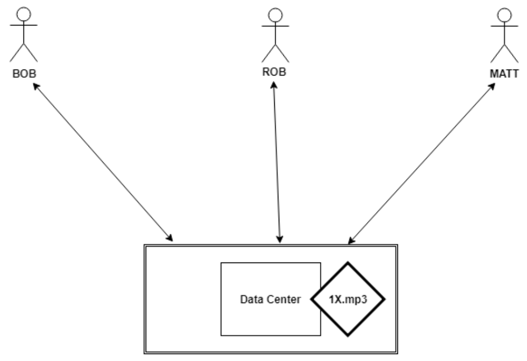

Figure 2. After file level deduplication.

2.3. Block Level Deduplication

In this type of data deduplication file is divided into blocks of data (number of blocks are predetermined).

These files are logically divided into blocks of data, as many users may have the same file but after a certain point of time one might change or modify some part of the file but not as a whole. Most part of the file remains the same as the original one but changes ever so slightly that it becomes a completely different file in case of file deduplication. In block level deduplication it keeps a track of which of those blocks are being updated/modified and only those modified blocks are stored in the data center including the blocks that are common to those files.

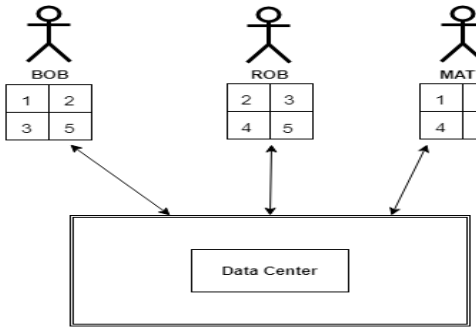

Figure 3. File uploading phase for file level Dedup.

Consider that three users Bob, Rob and Mat have the same file but they modify a bit in one of the blocks of data as shown in Figure 3, the common block of data is block number 5 all the other blocks (1, 2, 3, and 4) are either unique to one user or common to any two of them but not all. Block level deduplication only stores these blocks i.e., (1, 2, 3, 4, and 5) in the data center.

Whenever a user wants to retrieve the file it checks which of these blocks comprises a file for that user and it constructs the file accordingly and displays it to the user (Figure 4).

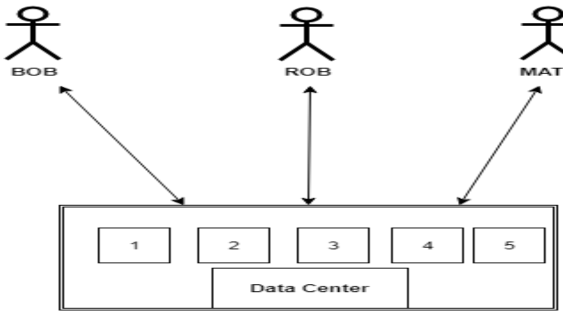

Figure 4. After block level deduplication.

2.4. Byte Level Deduplication

Data in terms of byte stream has a string of bits that repeat for various slightly similar or dissimilar data files. This repeating data can be stored as a pattern inside the data center with a track of the files that contain this pattern and to store their respective locations inside the files.

- Byte Stream 1-0110 1110.
- Byte Stream 2-1101 0110.
- Byte Stream 3-10001101.
- Byte Stream 4-01101100.

As shown in the above examples, all of them contain one pattern 0110 which repeats in every byte stream (this may be for different/same files). The patter 0110 can be stored once so that it reduces the amount of redundancy in the data stored at bit level. By making sure that repeating binary data is never more than once stored we can achieve the savings as mentioned above.

Deduplication usage conditions:
Before implementing a data deduplication exercise it is important to understand:

- Pain points of the company.
- SLAs (Service Level Agreement).
- Business objectives.

Environments data deduplication can be applied

- Files systems.
- Low change rate Databases.
- Virtualization.
- SAN/LAN (Storage Area Network/Local Area Network).
- ROBO (Remote Office / Branch Office).

3. Proposed Methodology

3.1. File Uploading Phase

As discussed in the literature survey File Level Data deduplication will provide highest amount of data savings. Although same file determination will take a longer while and success rate will also be lower. The second method fixed block size data deduplication will provide a better success rate and will take lesser execution time than File level but won't achieve high level of data storage savings [18, 22, 29].
The third method of variable block size chunking will have the highest amount of data storage savings but would take a longer time than to execute than fixed size chunking [19]. We've used Rabin-Karp algorithm [11] for variable size chunking. We use file level data deduplication only for small size files such as audio file, photos and small size documents which are no greater than 10-15MB in size. File larger than that are usually movies, documents or executable files which are rarely duplicated. Figure 5 represents a detailed diagram of the proposed architecture.

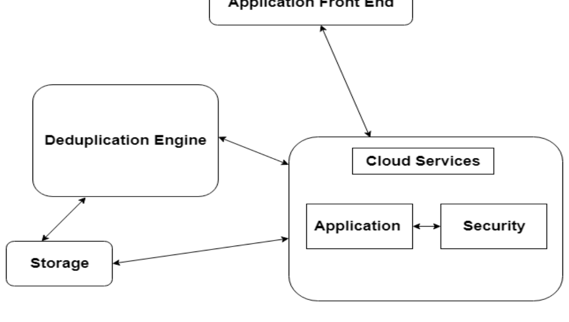

Figure 5. Architectural diagram.

1. Check for file deduplication.
2. Check for fixed size block deduplication (within file).
3. Check duplicated chunks using Rabin-Karp algorithm (across files).
4. Choose best of the above three algorithms to separate out duplicated elements.
5. Filter out the unique instances of all duplicated elements separate.
6. Store these unique instances of duplicated files/blocks.

Algorithm 1: File Level Data deduplication

```
if file.size() <= 15MB:
        for(Files f:  List_of_Files):
                if(file == f):
                        Map file.name to f
else:
        break
```

Algorithm 2: Fixed Sized Chunking

```
size = 1024*1024 (1MB)
List_of_Files – List of all the files in the directory
File – current file to be uploaded
MessageDigest md;
byte[] chunk = new byte[size];
for (File f : List_of_Files) {
        count = 0;
        if (f.isFile() && !f.isHidden()) {
        // Read file into a byte array and use SHA-1 hash the
chunk
        try {
        fis=new FileInputStream(f.getAbsolutePath());
                while (fis.read(chunk) != -1) {
        // perform the hash on the chunk
                        md.update(chunk);
                byte[] mdbytes = md.digest();
String hashvalue = byteToHex(mdbytes);
        // If not exist then save
if(!indexTable.containsKey(hashvalue)) {
indexTable.put(hashvalue, f.getName());
                }
        }// If
    }// while
  }//try
}// for
```

Rabin-Karp Algorithm for Variable size chunking Rabin-Karp string searching algorithm [11] calculates a numerical (shash) value for the pattern p, and for each m-character substring of text t. Then it compares the numerical values instead of comparing the actual symbols. If any match is found, it compares the pattern with the substring by naive approach. Otherwise it shifts to next substring of t to compare with p.

Algorithm 3 Rabin-Karp:

```
Compute $h_p$ (for pattern p)
Compute $h_t$ (for the first substring of t with m length)
For i = 1 to n - m
If $h_p = h_t$
        Match t[i…i+m] with p, if matched return 1
Else
        $h_t = (d\,(h_t - t([i+1].d^{m-1}) + t[m+i+1]) \bmod q$
```

3.2. File Retrieval Phase

The user typically will via the Application interface give a command to retrieve the file. This triggers the deduplication engine to locate the file and check if it was divided into chunks. Each user's data is divided into different directories with respect to their file type and extension. For example if the user wants to retrieve a file names "Test.mp3" with the user ID "U123", the lookup would be directory U123 inside which should contain a directory mp3 and inside which should contain the master file "Test.mp3" or should contain chunks of the files with the mapping information.

Each chunk has a unique hash value (SHA-1 hash) [7], each of these chunks for each file type are connected to each other by a linked list. Whenever a new chunk is formed it is added to the linked list. To retrieve the file first the dedup lookup table is checked if the file was deduplicated. This table contains information of the file name, file extension and control information (Yes/No) if it was deduplicated or not.

If it was deduplicated, then look into the mapping information which contains a list of nodes values in order of their appearance in the master file. These chunks are located and added in order as stated in the mapping information to form the master file which is then read to be downloaded.

4. Performance and Result

The below table represents the basic testing results of deduplication using Rabin-Karp algorithm [11] across all files for a single user. We've divided the analysis in terms of file type. The total size for all the files was 19.9GB and after deduplication the size became 17.89GB which saves about 2.01GB of space for my personal storage. In terms of percentage savings it sums up to ~11% savings. The compression factor would become 0.8.

If analyzed carefully, the duplication seems to be higher for Image, audio and text document files. For image files the savings is about ~830MB which is about 4 times less space than required about the same is seen for audio and text documents. The only file types which do not show large amounts of duplicated chunks are video files. Video files have shown the least amounts of duplicated content in our analysis which is about 1.7% only as compared to the 75.5% savings for image file types. The below Figure 6 gives a graphical representation of the tabular data provided in Table 1.

Table 1. Results for each files type.

File Type	Total Size (GB)	Deduplicated Size (GB)	% Savings
Image	1.1	0.269	75.5%
Text document	0.5	0.305	39%
Video	17.2	16.9	1.7%
Audio	1.1	0.42	61.8%
	19.9	17.89	10.1%

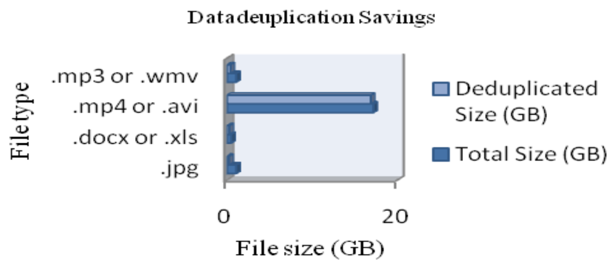

Figure 6. Results graph.

5. Conclusions and Future Work

The results show that we can save at least ~11 % of the personal data present at the desktop storage. Deduplication does not perform any complicated math to compress unlike other compression algorithms. The method is fast and efficient over a large size of data. The efficiency of the method will only increase as the data grows larger in size.

Data deduplication is also secure as it does not interfere with the underlying security algorithm/ protocol [28]. If can fit right into the security protocol and adhere to it. It does not violate any permission protocols as only the personal user data is used to deduplicated only with itself [23]. Data deduplication process work overtime and not as the data is uploaded to the storage device. Since we've used the mechanism of single user and overtime data deduplication method. This will ensure that data retrieval phase is smooth and secure.

The future work for this proposal would be to enhance its capability to perform more faster and increase it horizons by adding over multiple user data deduplication with some privacy transparencies. It can also be extended to a distributed environment with requirements of high availability allowing the underlying environment to replicate the deduplicated chunks in an efficient [8, 10] way.

We intend to use this method in a cloud environment, as we feel that a desktop environment is a virtualization of a private storage without the portability option. As a desktop storage is private and not shared with anyone the experimental results show a smaller picture than what we can achieve in a cloud environment. Implementing it in a cloud storage environment we would be able to perform data deduplication across clients as well and not just for one single user.

The results that we got from desktop data is the least that we would achieve across any data storage environment with large storage space. Cloud storage will only increase our data savings. As the number of clients and data increases, data savings will increase with that because as the data increases so does the redundancy.

References

[1] Biggar H., "Experiencing Data De-Duplication: Improving Efficiency and Reducing Capacity Requirements," *The Enterprise Strategy Group*, pp. 902-906, 2012.

[2] Castiglione A., Pizzolante R., De Santis A., Carpentieri B., Castiglione A., and Palmieri F., "Cloud-Based Adaptive Compression and Secure Management Services for 3D Healthcare Data," *Future Generation Computer Systems*, vol. 43-44, pp. 120-134, 2014.

[3] Chu X., Ilyas I., and Koutris P., "Distributed Data Deduplication," *Proceedings of the VLDB Endowment*, vol. 9, no. 11, pp. 864-875, 2016.

[4] Dolan M., Kochan L., Ram T., Rohr S., Tu K., and Miller S., Patent No. US20160292048, Retrieved from https://www.google.com/patents/US20160292048, Data Deduplication Using Chunk Files, Google Patent, Last Visited, 2016.

[5] Douceur J., Adya A., Bolosky W., Simon D., and Theimer M., "Reclaiming Space from Duplicate _Les in A Serverless Distributed _Le System," *in Proceedings of 22nd International Conference on Distributed Computing Systems*, Vienna, pp. 617-624, 2002.

[6] Demystifying Data Reduplication: Choosing the Best Solution, FalconStor Software, White Paper Dynamic Solutions International, https://www.varinsights.com/doc/demystifying-data-deduplication-choosing-0002, Last Visited, 2017.

[7] Eastlake D. Jones P., White paper: Description of SHA-1, http://tools.ietf.org/html/rfc3174, Last Visited, 2017.

[8] Estes J., Patent No. US20140258245, Retrieved from https://www.google.ch/patents/US20140258245, Efficient Data Deduplication, Last Visited, 2014.

[9] Harnik D., Pinkas B., and Shulman-Peleg A., "Side Channels in Cloud Services, the Case of Deduplication in Cloud Storage," *IEEE Security and Privacy Magazine*, vol. 8, no. 6, pp. 40-47, 2010.

[10] Jiang T., Chen X., Wu Q., Ma J., Susilo W., and Lou W., "Secure and Efficient Cloud Data Deduplication with Randomized Tag," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 532-543, 2017.

[11] Karp R. and Rabin M., "Efficient Randomized Pattern-Matching Algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249-260, 1987.

[12] Kleppmann M., A Critique of the CAP Theorem, http://arxiv.org/abs/1509.05393v2, Last Visited, 2017.

[13] Leesakul W., Townend P., and Xu J., "Dynamic Data Deduplication in Cloud Storage," Service Oriented System Engineering (SOSE), *in Proceedings of IEEE 8th International Symposium on Service Oriented System Engineering*, Oxford, 2014.

[14] Luo S., Zhang G., Wu C., Khan S., and Li K., "Boafft: Distributed Deduplication for Big Data Storage in the Cloud," *IEEE Transactions on Cloud Computing*, pp. 1-1, 2015.

[15] Meyer D. and Bolosky W., "A Study of Practical Deduplication," *ACM Transactions on Storage*, vol. 7, no. 4, pp. 14, 2012.

[16] Nelson M. and Gailly J., *the Data Compression Book*, M&T Books, 1991.

[17] Ngo D. and Muller M., Patent No. US8930306B1, Retrieved from https://www.google.com/patents/US8930306**,** Synchronized Data Deduplication, Google Patent, Last Visited, 2015.

[18] Park D., Fan Z., Nam Y., and Du D., "A Lookahead Read Cache: Improving Read Performance for Deduplication Backup Storage," *Journal of Computer Science and Technology*, vol. 32, no. 1, pp. 26-40, 2017.

[19] Patterson R., Reddy S., Prabhakaran V., Smith G., Bairavasundaram L., and Venkitachalam G., "System and Methods for Storage Data Deduplication," U.S. Patent No. 20,170,031,994, 2017.

[20] Puzio P., Molva R., Önen M., and Loureiro S., "PerfectDedup: Secure Data Deduplication," *in Proceedings of 10th International Workshop on Data Privacy Management, and Security Assurance*, Vienna, pp. 150-166, 2015.

[21] Qinlu H., Zhanhuai L., and Xiao Z., "Data Deduplication Techniques," *in Proceedings of International Conference on Future Information Technology and Management Engineering*, Changzhou, 2010.

[22] Ram T., Patent No.US20140095439, Retrieved from https://www.google.com/patents/US20140095439 Optimizing Data Block Size for Deduplication, Google Patent, Last Visited, 2014.

[23] Rehman A. and Saba T., "An Intelligent Model for Visual Scene Analysis and Compression," *The International Arab Journal of Information Technology*, vol. 10, no. 13, pp. 126-136, 2013.

[24] Sayood K., *Introduction to Data Compression*, Morgan Kaufmann, 2006.

[25] Shin Y., Koo D., and Hur J., "A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems," *ACM Computing Surveys*, vol. 49, no. 4, pp. 74, 2017.

[26] Slater A. and Pelly S., Patent No.US20110184908, Retrieved fromhttps://www.google.si/patents/US20110184908, Selective Data Deduplication, Google Patent, Last Visited, 2011.

[27] Stanek J., Sorniotti A., Androulaki E., and Lukas K., "A Secure Data Deduplication Scheme for Cloud Storage," *in Proceedings of International Conference on Financial Cryptography and Data Security*, Christ Church, pp. 99-118, 2014.

[28] Storer M., Greenan K., Long D., and Miller E., "Secure Data Deduplication," *in Proceedings of the 4th ACM international Workshop on Storage Security and Survivability*, Alexandria, pp. 1-10, 2008.

[29] Xia W., Jiang H., Feng D., Hua Y., "Similarity and Locality Based Indexing for High Performance Data Deduplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp.1162-1176, 2015.

Rishikesh Misal graduated from University of Mumbai with a bachelor's degree in Computer Engineer in 2015. He completed his Master's in Computer Science and Engineering from VIT University, Vellore. He has been working at General Electric for the past 1 year as a Software Engineering Specialist. His professional works are based on building Cloud applications for IoT based scenarios. His research work interests include Distributed Systems, Cloud Computing, System Programming and Compiler Construction.

Boominathan Perumal is an Associate Professor working in VIT University, Vellore, India. He received his B.E in Computer science and Engineering from Barathidasan University, Tirchy, India, M.E in omputer Science and Engineering from Anna University, India and he received his Ph.D. from VIT University, Vellore, India.He has 12 years of teaching experience. He has good number of publications in reputed conference proceedings and journals. His research interests include cloud computing, Network Security, and Evolutionary optimization, etc.