International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 5, ISSN No. 2455-2143, Pages 106-109
Published Online March – April 2016 in IJEAST (http://www.ijeast.com)

DeDuCS: DE-DUPLICATION OF CLOUD STORAGE

Naresh Sharma
Asst. Professor,
Dept. of CSE
SRM University, Ghaziabad

Amit Mishra
Dept. of Computer Science & Engineering,
SRM University, Ghaziabad

Purvi Garg
Dept. of Computer Science & Engineering,
SRM University, Ghaziabad

Abstract: **In this paper we are representing DeDuCS which is a server side application or simply we would say a servlet which restricts the upload of duplicate files on the cloud storage, which in turn results in increase in efficiency of cloud storage. DeDuCS compares the file to be uploaded with the files already present on the cloud and checks if a copy of the same file is present on the cloud or not, in order to ensure that only a single copy of every file is stored. If the file to be uploaded is found to be duplicate i.e. a copy of the same file is already present on the cloud server, a link to the already present file is given to the user and the upload is restricted. If the file to be uploaded is found to be unique, it is uploaded to the server normally. DeDuCS is developed in java and is tested using a distributed file system of several windows XP virtual machines. The experimentation shows that DeDuCS can be efficiently used to avoid de-duplicate the cloud storage.**

Keywords: **Cloud Storage, Cloud Server, DeDuCS (De-Duplicating Cloud Storage), De-Duplication, Duplicate Data, Public Cloud, Private Cloud, Servlet.**

I. INTRODUCTION

As the data is at its uncontrollable growth, duplicacy in data is also increasing and with this comes the need for an efficient technique to manage such huge amount of data. Managing data means making data secure, providing remote access to data and most important is removing redundancy from data and organizing the storage structure so as to obtain maximum efficiency. The University of Southern California calculated that the amount of stored data grew by 23% per year between 1986 and 2007 [1] which would definitely have increased even more in the last eight years. This duplicate data occupies space and requires operations which are useless as the original copy has already been processed and stored and hence, the same processing and storage of the replicas are not required.

A. Cloud Storage

Cloud Storage is a service where data is remotely maintained, managed, and backed up. The service accessed by the users over the internet. It allows storage of files online which can be accessed from any location using any device connected to the internet. The file is kept online in an external server by a cloud service provider companies. This in turn provides the users with easy and convenient access of the files which can be potentially costly at times. While using cloud storage, it should always be kept in mind that a local backup of the data is still required as recovering data from the cloud is a slower process. Cloud Storage is mainly of two types:

1. Public Cloud: Shared by many users.
2. Private Cloud: Owned by a single user.

B. *Duplicate Data*

Duplicate data refers to the files or records having exactly same content. For example, two or more of the team members in any organization may have different copies of the same file at different locations.

C. Data De-Duplication

Data de-duplication is a technique by which we store exactly one copy of every data element and replaces the replicas by pointers which point to the original data i.e. the storage is free from any redundancy. Data de-duplication is mainly of two types:

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 5, ISSN No. 2455-2143, Pages 106-109
Published Online March – April 2016 in IJEAST (http://www.ijeast.com)

1. File level de-duplication: This is the de-duplication mechanism in which exactly one copy of any file is stored and the pointer to the file is distributed to all its users. This type of de-duplication is also known as Single Instance Storage (SIS) as only a single instance of a file is kept in the memory.

2. Block Level De-duplication: This type of de-duplication eliminates duplicate blocks of data from non-identical files.

II. PROBLEM STATEMENT

Development of technology is leading to an exponential growth of data present on the internet. People want to have an easy access of their data which is giving popularity to the cloud storage service. A user can just upload his documents once and can access them from anywhere using any portable device like laptop, smart phones or tablets via the internet. As the amount of data is growing, the storage devices are not. Common cloud service providers charge heavy amount of money for the storage services provided, and hence to efficiently use those services is a matter of concern. Current cloud storage models allow the storage of multiple copies of the same file which reduces the efficiency of the cloud storage as a lot of storage space is wasted in storing the duplicate content. The prime goal of DeDuCS is to restrict the upload of duplicate files on the cloud storage and make sure that only a single copy of any file is stored to the server.

III. RELATED WORK

In the last decade, much of the research has been done in the field of data de-duplication. Several de-duplication schemes have been proposed by the research community [4, 5, 6, 7] showing how de-duplication allows very appealing reductions in the usage of storage resources.

Many distributed file systems for large scale information systems have been proposed which are distributed over the internet include non-clustered and mutable peers. Frequent configuration changes occur in all these systems. Sparse indexing [8] and Bloom filters [5] with caching are the two famous approaches that have been proposed to handle large scale de-duplication.

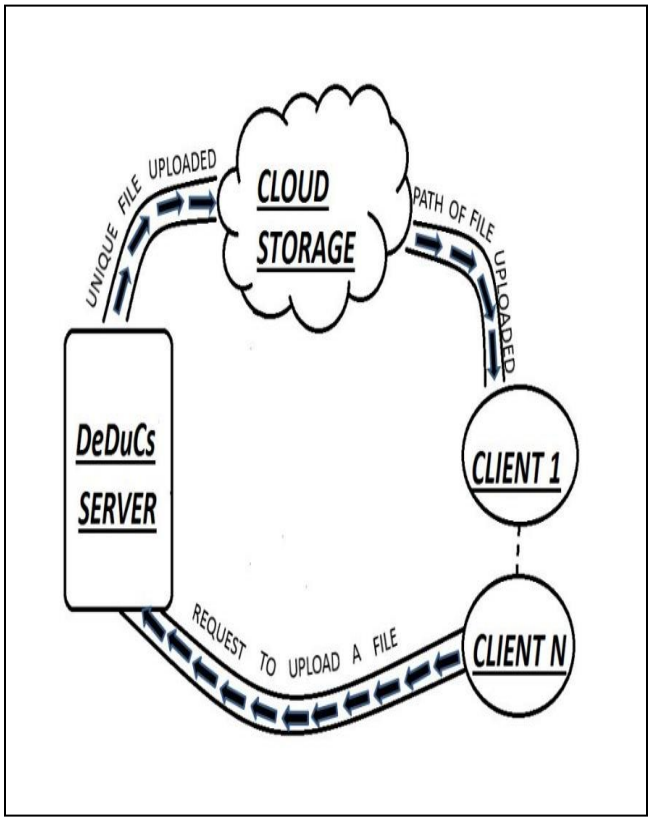
IV. PROPOSED METHOD

The 'once write multi read' storage concept became popular in the early 1990's, which promoted the use of optical disk. But this concept had to face many challenges like the obstacles encountered when sharing data over the internet and the wastage of storage space in keeping the backups. In this novel cloud computing era, we propose DeDuCS, a servlet to check for duplicates before uploading the file to the cloud. In the novel cloud storage which is being used these days, whenever a user uploads a file to the cloud, he login to the cloud server using a web application, uploads the file and the system provides a URL to the user in order to access the file later. This is how DeDuCS also works, but with a slight modification.

A. DeDuCS Work Flow

In the case of DeDuCS, the servlet first checks for any duplicates present and uploads only unique documents every time. Whenever a duplicate file is being uploaded, the system returns the URL of the original file to the user and thus in this way only a single copy of each file is present on the cloud.

In the DeDuCS work flow diagram shown below, client 1 to client n shows that DeDuCS is primarily designed of public or shared cloud, but it works equally well with the private cloud. Whenever the user or client tries to upload a file, the file is transferred to the DeDuCS server which is basically the cloud server on which the DeDuCS servlet is running. It performs content matching between the files present on the cloud and the file being uploaded. If no duplicate file is found, the file is simply uploaded to the cloud storage and a URL is given to the user to access the file as in the case of normal cloud storage.



International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 5, ISSN No. 2455-2143, Pages 106-109
Published Online March – April 2016 in IJEAST (http://www.ijeast.com)

Fig. 1: DeDuCS Work Flow Diagram.

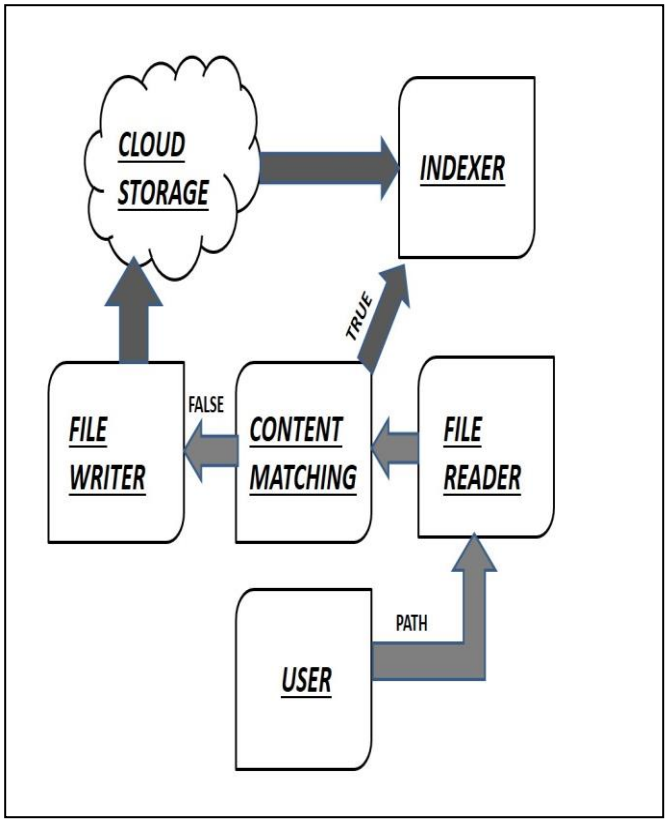


Fig 2: DeDuCS System Architecture

In the case of duplicate files, the DeDuCS server does not upload the file but gives the URL of the original file to the user. To understand the operation of DeDuCS, let us take an example of an enterprise cloud where the same storage is accessed by multiple users. Suppose user 1 uploads a document and gets a URL to access the document online.

Now user 2 tries to upload the same document, he is not allowed and is given the URL of the original document which was uploaded by user 1. Now when user 1 modifies the document, a copy of the original document is made which is at the original URL given to user 1 and user 2. The modified document is saved as a new document whose URL is given to user 1.

V. SYSTEM ARCHITECTURE

The architecture diagram shown below explains the different components of a DeDuCS enabled cloud storage.

User: User is the client who is using the cloud storage to back up his data online.

Path: The path of the file on the client machine which is given to the cloud storage interface in order to upload the file.

File Reader: File reader reads the file from the client machine and sends for content matching.

Content Matching: File level content matching is done between the file being uploaded and the files present on the cloud.

File Writer: If the result of the content matching is false i.e. the file being uploaded is unique, the file writer transfers the file to the cloud using TCP socket.

Indexer: The indexer contains the details of the files present on the cloud using which the URLs of different files are taken by the content matching program.

Cloud Storage: This is the cloud server on which the files are being saved.

VI. IMPLEMENTATION

DeDuCS has been implemented in Java SE 1.7 using the eclipse IDE on a windows 8.1 machine. It contains separate classes for reading the file i.e. File Reader, to check the duplicates i.e. Content Matching, to transfer the file to the cloud storage via TCP socket i.e. File Writer and an Indexer to get the URLs of the files already present on the cloud. The content matching has been done using the Apache Commons io-2.4 library which performs file level content matching and gives Boolean results. The DeDuCS program has been designed to embed it as a servlet in the front end web application of the cloud storage.

VII. EXPERIMENTATION & RESULT

To test the performance of DeDuCS, we modelled a distributed cloud storage system by creating five windows XP virtual machines which were connected via LAN. The host computer i.e. the windows 8.1 machine on which the VMs were running acted as a client. One of the VMs were treated as the cloud web server on which the DeDuCS program ran and we would use the term DeDuCS server to refer to it and the other VMs were treated as the distributed file system of the cloud and we would use the term storage server to refer to them.. The client that is the windows 8.1 machine login to the DeDuCS server using telnet and gave

the path of the file he wanted to upload. The DeDuCS server accessed the distributed file system of the storage servers and checked for the presence of any duplicates on them. The file transfer between the client and server was done using TCP socket programming. We checked the system with a combination of documents, music files, html files, etc. and got appreciable results. Only unique files were transferred to the storage servers and the URL to access the file was shared between users who wished to upload the same file. The DeDuCS work flow shown above was strictly followed by the system. The response time of the system in performing the content matching varied form 27 ms to 332 ms depending on the amount of files being present on the storage server.

VIII. CONCLUSION AND FUTURE SCOPE

By the results of the experimentations and observation of the performance, we conclude that DeDuCS can be efficiently used with any cloud storage system and increase the storage efficiency of the cloud as by storing only a single copy of every file, it would save a lot of storage space wasted in storing the duplicates. Primarily DeDuCS focusses on the public i.e. the shared cloud but it works equally well with the private cloud. As of now, commercial cloud storage systems like Google Drive and Sky Drive does not provide any de-duplication mechanism in the private clouds. With the use of DeDuCS, it can be easily provided and thus a lot of storage space can be saved on the global scale.

As we are doing direct content matching between the files, the approach will not be much scalable and hence an alternate approach to check the similarity of documents would be required. The second goal in the future would be implement security mechanisms and provide secure de-duplication as security in cloud as it is a matter of prime concern and cloud security is itself a wide area of research hence the DeDuCS project can be expanded to enhance the security aspects. These are the prime targets as of now which may expand as de-duplication of cloud is a vast area of research in this decade.

IX. REFERENCES

[1] Suzane Wu, “How much information is there in the world?” *USC News, February 2011*. http:/news.usc.edu/29360/how-much-information-is-there-in-the-world/

[2] Jiawei Yuan; Shucheng Yu, "Secure and constant cost public cloudstorage auditing with deduplication," in *Communications and Network Security (CNS), 2013 IEEE Conference on* , vol., no., pp.145-153, 14-16 Oct. 2013

[3] Jin Li; Yan Kit Li; Xiaofeng Chen; Lee, P.P.C.; Wenjing Lou, "A ybrid Cloud Approach for Secure Authorized Deduplication," in *Parallel and Distributed Systems, IEEE Transactions on* , vol.26, no.5, pp.1206-1216, May 1 2015

[4] Lior Aronovich, Ron Asher, Eitan Bachmat, Haim Bitner, Michel Hirsch, and Shmuel T. Klein. “The design of a similarity based deduplication system” *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference (SYSTOR '09). ACM, New York, NY, USA*, Article 6, 14 pages.

[5] Stanek, Jan, et al. "A secure data deduplication scheme for cloud storage." *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2014. 99-118.

[6] D. Cezary, G. Leszek, H. Lukasz, K. Michal, K. Wojciech, S. Przemslaw, S. Jerzy, U. Cristian and W. Michal, HYDRAstor: a Scalable Secondary Storage, *Proccedings of the 7th conference on File and Storage Technologies, San Francisco, California*, Vol. : No. : pp. 197-210, 2009.