International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 4 696 – 699

Erasure Code Based Cloud Storage System

Makhan Singh
Computer Science and Engineering,
University Institute of Engineering & Technology,
Panjab University, Chandigarh, India.
singhmakhan@pu.ac.in

Abstract— Cloud Computing is the technology that provides on demand services and resources like storage space, networks, programming language execution environment on the top of Internet pay per use model. Cloud computing is globalized concept and there are no borders within the Cloud. Because of attractive features of Cloud computing, many organizations are using Cloud storage for storing their critical information. The data can be stored remotely in the Cloud by user and can be accessed using thin clients as and when required. One of the major issue in Cloud today is data security. Storage of data in the Cloud can be risky because storage is done on Cloud service providers' servers which mean less control over the stored data. One of the major concern in Cloud is how do we grab all the benefits of Cloud while maintaining security controls over the data. In this paper reliable storage system is proposed which can be robust in case of errors or erasures in data to be stored. Proposed system provides reliable storage while maintaining the integrity of the data. The files are split into parts to get an extra layer of security

Keywords-*Cloud storage; erasure codes; cloud service providers; reliabilty; data security.*

*****

I. INTRODUCTION

Cloud Computing is the technology that provides on demand services and resources like storage space, networks, programming language execution environment on the top of Internet pay per use model. One of these services provided by the cloud computing environment is the storage of data on the Cloud data centers. Digital data is very critical resource hence Cloud data centers provide sufficient data availability and security even in the case of failure and malicious attacks. This paper uses erasure code [1] which improves the security and availability of data on the Cloud data centers. The proposed scheme also completely reconstructs original data even it is partially damaged and allows secure storage of data. The cloud users can provide data that they need to store on the cloud data center as a means of creating backup for it. The backup of these files is traditionally maintained by replicating the files on multiple data centers such that any of these replicated parts can be used to retrieve the file on request from the user. Another method used is deduplication where the encryption of the data is overridden and only one copy of the file is stored and the encryption key for the file is deleted whenever the file needs to be removed.

These techniques, however, have a lot of drawbacks like lack of security of the data, high redundancy and transmission overheads.

In literature many efforts are made to use erasure code for securing cloud storage secure and reliable. All the efforts were made at server side of cloud service providers or work on securing cloud data [2][3][4][5]. Huang et al. used the erasure code in windows Azure storage [4] by introducing new set of codes termed as Local Reconstruction Codes. These codes helps in reducing the number of fragments required for reconstruction. They had divided the redundant data into two sets called local and global groups, and geographically separated storage local and global groups, and geographically separated storage servers are used to store the same. This reduces the total reconstruction cost as the local groups needs minimum input/output and network overhead for data reconstruction. Gomez et al. used erasure codes to influence the Iaas clouds to save data in reliable way without any additional overheads in Iaas. They have also proposed a scalable erasure coding techniques that provides high coding techniques that provides high degree of reliability for local storage with low computational overhead and cost less amount of communication [3]. Santos et al. proposed scheme which provides execution environment which acts as a closed box hence it guarantees confidential execution of guest virtual machines on virtual machines on cloud infrastructure. They proposed the design of a trusted cloud computing platform for Infrastructure as a Service [6].

All the above schemes for securing cloud data assumes that cloud service providers (CSPs) can be trusted and can prevent any physical attacks on servers at their site. But in reality this is not always true and there have been many incidents where cloud services were attacked by internal or external hackers hence users data was also compromised. Due to this relying on service provider's security techniques are not feasible solutions for storing critical user data. Due to these reasons there is a need for applying erasure coding methods at application level by the use of multiple CSPs. Here user's data has been encoded as well as redundancy is also introduced in it. After this the encoded redundant data have been stored across multiple cloud storage service provider's servers. This approach acts as fault tolerant when ever any cloud service providers server fails or attacked.

II. SECURE CLOUD STORAGE

The proposed model is designed in the various components, where each of the components independently handles the individual group of tasks as shown in figure 1. The proposed model can be primarily divided into the following components:

A. Data Storage Mechanism

The data storage mechanism is the primary model to store the data across the available number of data centers. The server list consists of the data centers/storage server along with the number of segments for each of the server on the list, which eventually depicts the upload scheme for the data storage mechanism. The probability of failures involves the risk of data loss, which is caused due to errors introduced during saving or transmission of the data. The data loss has been simulated in the simulation varying from server to server in the server list available to store the data. The probability of error in the retrieved data blocks is reduced by using the Reed-Solomon code, which is primarily designed for the error correction.

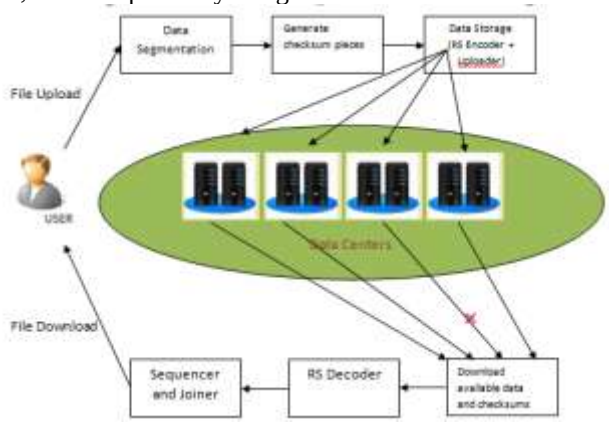

Figure 1. Architecture for secure Cloud storage system

B. Reed Solomon Coding

Reed-Solomon codes are specifically designed error-correcting mechanism, developed by Gustave Solomon and Irving S. Reed. The Reed-Solomon codes are utilized in the various applications such as DVD, CD, Blu-Ray Disk, data transmission channels, Barcodes or QR-codes, etc for the error correction for the accounted data [7]. The Reed-Solomon mechanism has been primarily designed for the correction of the multiple bit errors in a data block, and it is capable of covering the symbols by adding up the value t for the estimation of the errors in the given data. The Reed-Solomon encoding (RS-encoding) is capable of correcting the erroneous symbols up to [t/2], which defines a deeper range and corrects more than 90% of the symbols lost during the storage under our proposed procedure. In the proposed application, the RS encoding method is utilized to correct the erasures ranging between [t/2] and t.

C. Calculating the number of Checksums

Checksums are the parity blocks that are to be added to the original data which helps to recreate the original data in case of lost data parts. In order to get high performance from the system, minimum number of Checksums should provide the ability to handle more data center failures. Let S be the total number of data centers where each data center is denoted as $dc_1, dc_2 \ldots\ldots dc_S$, N be the number of parts data is divided into and $n_1, n_1, \ldots\ldots, n_N$ be the number of data pieces stored at the corresponding data center. Let M be the total number of data centers allowed to fail or be unavailable at a certain time. Let $m_1, m_2, \ldots\ldots, m_S$ be the number of checksums to be stored on corresponding data center and $m = m_1 + m_2 + \cdots\ldots + m_S$ is the total number of checksums needed for the file. The value of m is to be minimized to get better efficiency but with a condition that the number of total checksums in the remaining working data centers is greater than the number of data parts lost due to failure of one or more data center. An example of DC=3 and M=1 can be taken which can be solved as follows:

$$\text{minimize} \quad m_1 + m_2 + m_3$$

subject to

$$m_1 + m_2 \geq n_3 \qquad [\text{if data center } dc_3 \text{ fails}] \qquad (3.1)$$

$$m_2 + m_3 \geq n_1 \qquad [\text{if data center } dc_1 \text{ fails}] \qquad (3.2)$$

$$m_1 + m_3 \geq n_2 \qquad [\text{if data center } dc_2 \text{ fails}] \qquad (3.3)$$

From the above equations it was found out that an optimal number of checksums for a data file is N/2, so for example 5 data parts 3 checksums are optimal.

D. Calculating the Checksums and Data Reconstruction

In order to calculate the checksums firstly a $(m + n) \times n$ Vandermonde matrix A is created. Vandermonde matrix has elements of a geometric progression in each of its row. A $m \times n$ Vandermonde matrix looks as below:

$$V = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 \cdots & \alpha_1^{n-1} \\ \vdots & & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 \cdots & \alpha_m^{n-1} \end{pmatrix}$$

$$\text{or} \quad V_{i,j} = \alpha_i^{j-1} \qquad (3.4)$$

Vandermonde matrix has another property that if m rows are deleted from the matrix the new matrix formed is invertible. Now get a new matrix known as Information Dispersal matrix B by applying some finite sequence of elementary row operations on the matrix A which has an Identity matrix on top of a $m \times n$ matrix. It is to be noted that applying elementary row operations on the matrix B does not change the rank of matrix so its property that states that if m rows are deleted then the resultant matrix is invertible. Multiply the matrix B with the data matrix D where each row denotes one part of the data to get a new matrix whose first n rows are the data pieces and last m rows are the required checksums.

E. Uploading Data Blocks to Data Centers/Servers

This module has been designed for the server uploads. The segments of data are uploaded with the accordance to the Server List created in Server Selection after encoding each of the blocks using Reed-Solomon encoding. The Upload algorithm involving the overall design of the algorithmic workflow has been described in the following algorithm:

Algorithm 1: Server Upload Algorithm

1. Get data blocks (n) from Segmentation Algorithm.
2. Encode the data using Reed-Solomon Encoding (Algo. 2) to get (n + m) pieces containing checksums.
3. Load the Server List created from the Server Selection Algorithm.
4. Run the iteration considering the Server List for all the servers
   a. Do while Counter, C1 of server 1 is not equal to 0
      i. Upload current block on the server 1
      ii. Decrement C1 by 1
   b. Do while Counter, C2 of server 2 is not equal to 0
      i. Upload the current on the server 2
      ii. Decrement C2 by 2
   c. Continue till server n
   d. Do while Counter, Cn of server n is not equal to 0

 ISSN: 2321-8169
696 – 699

i. Upload the current on the server n
ii. Decrement Cn by 1

5. Return the final upload status

Each block or segment is encoded as using Reed-Solomon before uploading to the appropriate data center as it makes the system error resistant and can correct errors at the time of data download. The Reed-Solomon encoding process is explained using an algorithm as following:

Algorithm 3: Reed-Solomon Encoding

1. Convert each segment of data into data vector D of n-bytes.
2. Compute m number of n-bytes checksums by taking a $(n + m) * n$ Vandermonde matrix A and convert it into Information Dispersal matrix B.
3. Multiply the Information Dispersal matrix B to the data vector D to get the encoded data matrix, $B \times D = E$.
4. Add checksums to the original data to get the $(n + m)$ pieces of n- bytes each.

F. Reliable Download and Reconstruction of data

This module deals with downloading the different parts of the data, decoding them using Reed-Solomon decoder and then joining them to give the original data without any errors. It can be explained in detail in the following algorithm:

Algorithm 4: Reliable Download and Reconstruction of Data

1. Check for all the required parts to recover the original file.
2. If all the data parts are available
a. Download all the parts of the file and append them together
3. Else If some parts k ($k \leq m$) are lost
a. Use RS decoding (Algorithm 7) to recover the lost parts.
4. Put the data pieces into right order to get the file.

Here, n is the number of symbols in the codeword, k is the number of symbols in original message and m is the number of bits per symbol. After downloading the segments they are decoded using RS Decoding so as to correct errors or/and erasures which might have crept in at the time of storage before finally joining them to get the original data.

Algorithm 5: Reed-Solomon Decoding

1. Get the available (n-k) data parts and checksums.
2. Use all the data parts downloaded and k checksums to create a $n \times n$ matrix.
3. Remove the rows corresponding to the lost data in Information dispersal matrix B to get B´.
4. Multiply B´ with data vector, D.
5. Multiply with its inverse B´-1 to get the original data pieces.

III. Performance Analysis and Experimental Results

This section presents the overall results obtained from the proposed model in order to estimate the accuracy based assessment of the proposed model for uploading and downloading of data over the error-prone transmission as well as error-prone network servers (known as the connected peers). The various sizes of data files have been utilized for the estimation of the performance of proposed system using the performance parameters discussed above. Simulation results were taken carefully and plotted into charts to get a better perspective of the results achieved. The results are separated into two parts as Uploading and Encoding part and Download and Decoding part. The charts given below provide a lot of information about the system.

To analyze the performance of our approach, the file size is taken 156 MB. Figure 2 shows the encoding and uploading time vs. the number of data pieces set by the user. From the figure, we can see that when we increase the number of data pieces from 2 to 8, the uploading time drops

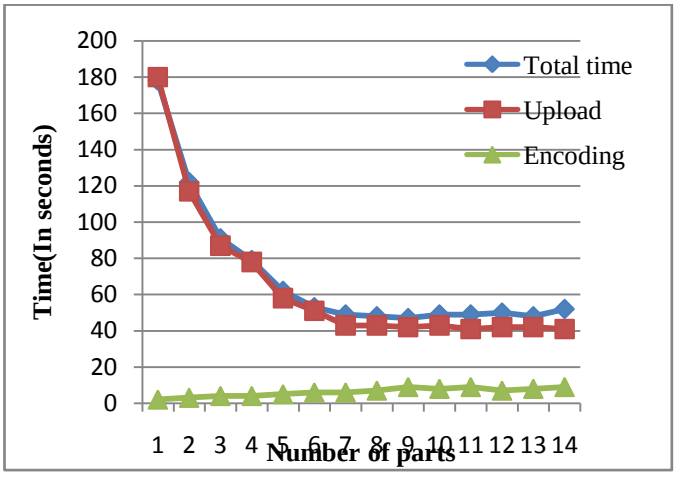

Figure 2. Encoding & Uploading Time vs. Number of Data Pieces

down significantly; while the encoding time has slightly increased. The significant performance improvement for uploading is due to the parallel processing of the systems, while the increased number of data pieces along with more checksum pieces results in more overhead for encoding. However, when the number of data pieces n is increased further, the uploading time and the total processing time very with very less variation.

The proposed model has also been analyzed for its performance during the downloading of the data from the centralized cloud service. The data download involves the two major steps, retrieval of the file parts followed by the decoding of data, which are further combined from all file parts to combine the output file. Figure 3 shows the downloading and decoding time vs. the number of data pieces. From the figure, when we increase the number of data pieces from 2 to 8, the

ISSN: 2321-8169
696 – 699

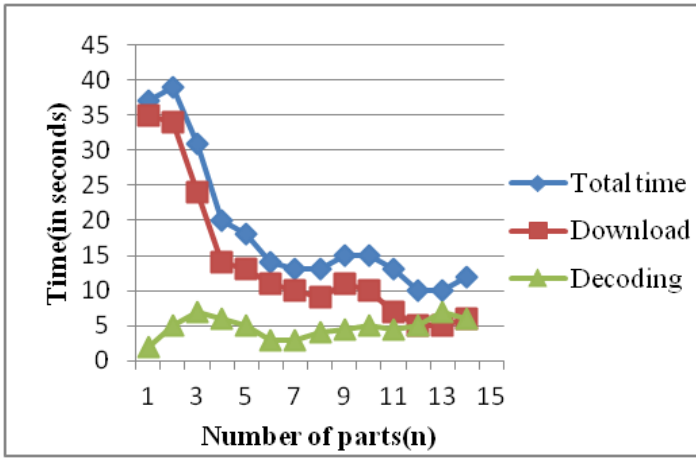

Figure 3. Downloading and Decoding Time vs. Number of Data Pieces

downloading time drops down significantly; while the decoding time has slightly increased. Similar to the case of uploading, the significant performance improvement for downloading is also due to the use of multithreading technique, and the increased number of data pieces along with more checksum pieces results in more overhead for decoding. When the number of data pieces n is increased further, the total processing time only slightly goes up due to the overhead of decoding files. However, when $n \geq 8$, the total processing time is constantly below 15 seconds.

IV. CONCLUSION

The proposed model has been designed for the online storage model based on the Reed Solomon encoding for the storage of data. The proposed model consists of the data centers designed to store the data over the target storage system. Data is split into parts as required by the user and encoded with the checksums calculated. Experimental results show that besides the advantages of being secure and fault tolerant, this approach provides very good performance in both file uploading and downloading, with the cost of minor overhead for encoding and decoding data.

REFERENCES

[1] J. S. Plank, "Erasure Codes for Storage Systems: A Brief Primer," Login: The USENIX Magzine, www.usenix.org, December 2013, Vol. 38, No. 6, pp. 44-50.

[2] C. Huang, H. Simitci, Y. Xu et al., "Erasure Coding in Windows Azure Storage," Proceedings of the 2012 USENIX Annual Technical Conference, Boston, MA, USA, pp. 15-26 , June 13-15, 2012.

[3] L. B. Gomez, B. Nicolae, N. Maruyama, F. Cappello and S. Matsuoka, "Scalable Reed-Solomon-based Reliable Local Storage for HPC Applications on IaaS Clouds," Proceedings of the 18th International Euro-Par Conference on Parallel Processing (Euro-Par'12)Greece, pp. 313-324, August 2012.

[4] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads," Proceedings of the 10th USENIX Conference on file and Storage Technologies (FAST-2012), San Jose, CA, USA, pp. 20-33, February 2012.

[5] K. Hwang and D. Li, "Trusted Cloud Computing with Secure Resources and Data Coloring," IEEE Internet Computing, Vol. 14, No. 5, pp. 14-22, 2010.

[6] N. Santos, K. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," Proceedings of the Workshop on Hot Topics in Cloud Computing (HotCloud09), Article No. 3, San Diego, CA, June 15, 2009.

[7] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes, North-Holland Mathematical Library, Amsterdam, London, New York, Tokyo, 1977.