A Framework for Cloud Storage System Based on Information Dispersal Algorithm

Makhan Singh, Sarbjeet Singh

Abstract*: Cloud computing is the technological model that deliver services on demand services using Internet service as pay and use model. Cloud computing is global abstract idea and there are no boundaries within the Cloud. Cloud computing is used by many organizations for backup their valuable data. The data can be backup over the Cloud servers by user and can be retrieved as and when needed. One of the main disadvantages related to Cloud today is data security. Storage on Cloud is termed as risky because storage is done on third party Cloud service providers. Due to this Cloud user have less control over the stored data. In order to explore full advantages of Cloud computing data security related issues have to be resolved. This paper proposed reliable storage system using information dispersal algorithm popularly known as erasures coding to secure data by splitting the files into parts to get implicit security.

Index Terms*: Cloud storage system, Cloud service provider (CSP), Download and upload time, Erasure coding, Information dispersal, Security.

I. INTRODUCTION

Cloud Computing is the technological model that deliver services and infrastructures like storage, networks, programming language execution environment on the top of Internet as per pay and use model. Storage is the one of most important services provided by the Cloud computing environment. Digital data stored on third party Cloud platforms is very critical resource has to be provided sufficient data availability and security even if some failure occurs and of course due to malicious attacks. This paper uses erasure code [1] as an information dispersal technique helps in improving the security and availability of data stored on data centers. The presented scheme also completely retrieves the original data even it is partially hacked or damaged and allowing security of stored data. Traditionally availability and security is maintained by replication, encryption and de-duplication.

These techniques, comes with many disadvantages like having less security for the data, high redundancy and transmission cost.

In literature all the efforts were made securing the Cloud service providers or work on securing Cloud data [2]-[5]. Huang et al. used the erasure code as information dispersal algorithm in windows Azure storage [4] by introducing new set of codes termed as LR codes (Local Reconstruction Codes). These codes helps in minimizing the count of data parts required for retrieval of data. Their proposed approach created two groups as local and global set for redundant data and stored on the geographically separated storage servers. This minimizes the total reconstruction cost due to less input or output and network overhead required for reconstruction of data. Gomez et al. used erasure codes to exploit the IAAS Clouds platform to protect data in reliable way without any additional expenses in IAAS. They have also proposed a scalable erasure coding techniques that provides robust coding techniques that provides higher degree of reliability for local storage with low computational and network communication overhead [3]. Santos et al. proposed scheme provides execution framework behaves like a closed box hence it maintains confidentiality of execution of guest virtual machines over virtual machines on Cloud infrastructure. Their framework is considered as design of a trusted Cloud computing platform for Infrastructure as a Service [6].

As presented in above literature the problem of securing user data can be solved by using information dispersal algorithm based on Reed-Solomon which is (k, n) threshold scheme which divides the data into n parts and k parts are required to reconstruct the original data and can be implemented as a application for securing data by storing it on different CSPs. This approach adds redundant information as well as encodes the data so that it can be stored on multi-Cloud storage architecture after dividing the data in different parts. By operating these operations on data the data can be implicitly secured from any type of maliciously attacks.

II. SECURE CLOUD STORAGE

The paper presented a framework having various modules that executes the individual group of processes as shown in figure1.The tasks performed by different modules of the proposed framework are as follows:

Data Segmentation: Splits the files into parts/pieces according to user requirement.

Budget Approval and Server Selection: Decides the cost of storage on the basis of reliability and budget provided by the user. Prompts the user to alter the cost in case it is lower than the minimum required for that reliability level or vice versa. Server selection is performed on the basis of the minimum reliability and cost approved by the cost model.

Revised Manuscript Received on December 22, 2018.

Makhan Singh, Computer Science and Engineering, UIET, Panjab University, Chandigarh, India.His Department Name, University/ College/ Organization Name, City Name, Country Name.

Sarbjeet Singh, Computer Science and Engineering, UIET, Panjab University, Chandigarh, India.

Data Storage: The data storage mechanism deals with encoding the divided parts using RS encoding and storing them on the selected data centers.

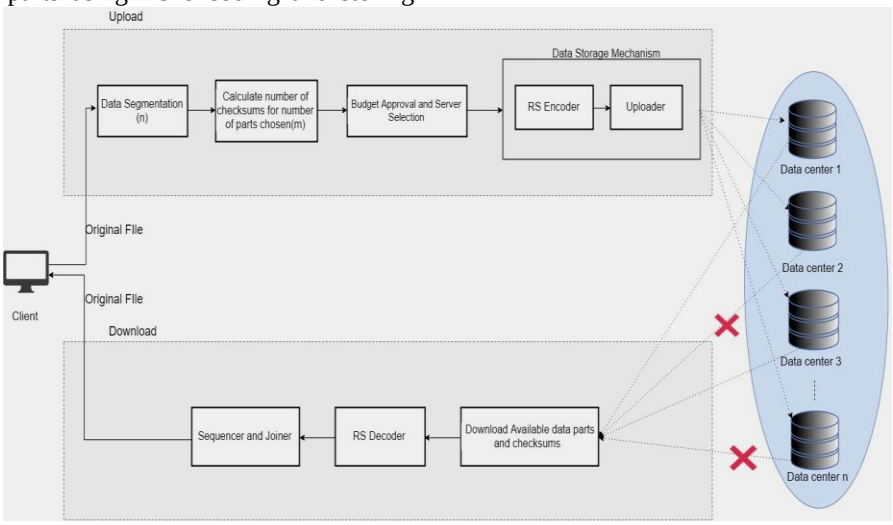

Figure 1: Framework for secure Cloud storage system

Data Retrieval: This module is concerned with downloading parts of a file, decoding them using RS decoding and joining them together to get the original data.

The Cloud-based online storage systems are rising across the internet world, as they are kept on offering the multi-faceted online storage systems. The performance of the Cloud-based online storage model relies upon the reliability and availability of the data storage over the data centers in the Cloud platforms. The points of failure in the Cloud platform, which can primarily occur in the target data centers or some of their clusters, raise the major threat to the online storage models. The fault tolerance and accuracy becomes the primary indicator for such systems in order to offer the vital resource for the online storage models. The data centers are consisted of the high-performance hardware, such as servers, disk storage, etc, along with the higher intra-network capacity and the online network link (dedicated links, leased lines, etc) to offer the high capacity storage systems. In this thesis, the robust, flexible and secure online storage model has been proposed, which is capable of handling different sizes and types of the data over the online storage. Cost and Reliability optimization is also been done in the proposed system so as to give the highest possible reliability for a certain budget. Segmenting the file into pieces further improves data security as all the information is not present in one block. The data blocks segmented are then stored across different servers selected on the basis of minimum reliability and budget given by the user.

A. Data Segmentation

The data segmentation approach is intended to divide the data into the given number of blocks, which may be fixed or flexible varying by application's nature. In this algorithm, data is divided into a number of parts according to the user requirement. If the user needs more confidentiality, he/she can have more parts but with some overhead for encoding a large number of parts. The default value for the division is taken to be 16 because dividing the data into 16 parts gives us a decent level of confidentiality without adding much overhead for encoding and decoding of the parts. The reliability is very important for the data segmentation algorithm, it must be highly accurate to meet the requirements segmentation of the data performed during the upload and download methods respectively.

Algorithm 1 is used for segmenting the data. The level of segmentation to be applied is decided by the user as discussed above. The proposed data segmentation algorithm is also capable of padding the data matrix accordingly i.e. vertically, horizontally or in both directions according to the requirement of the user. The input data is first checked for the need of padding, if yes then the matrix is padded appropriately before dividing it into equal size parts. The algorithm used is given as following:

Algorithm 1: Data Segmentation Algorithm

1. Input data, dataMatrix
2. Input the number of segments/parts (n)
3. Obtain the size of data rows and columns, [R, C]
4. If data matrix gives condition, [R==1] || [C==1]
    a. Divide the vector with the total number of segments (n), gives Nc
    b. Find the floor value from Nc, given Ncf
    c. Compute total number of elements in the given data matrix, T = R*C
    d. Compute the required number of entities in the data matrix, Tnc = (Ncf * N)
    e. Compute the symbols to be padded for the successful division, Pn = Tnc – T
    f. If R==1
        i. Apply padding of [Pn] values over columns
    g. Else if C==1
        i. Apply padding of [Pn] values over rows
    h. Return the data matrix

5. Else
    a. Divide n in two prime numbers, [NC, NR]
    b. Divide C with the total number of segments (NC), gives Nc
    c. Divide R with the total number of segments (NR), gives Nr
    d. Find the floor value from Nc, given Ncf
    e. Find the floor value from Nr, given Nrf
    f. Compute the required number of entities on columns in the data matrix, Tc = (Ncf * NC)
    g. Compute the required number of entities on rows in the data matrix, Tr = (Nrf * NR)
    h. Compute the symbols to be padded to the columns for the successful division, Pc = Tc – C
    i. Compute the symbols to be padded to the rows for the successful division, Pr = Tr – R
    j. Apply padding of [Pn] values over columns
    k. Apply padding of [Pn] values over rows
    l. Return the data matrix
6. Divide the number of blocks from the data matrix
    a. If data matrix gives condition, [R==1] || [C==1]
        i. Set the seed value of seed to 1
        ii. Compute the block size of 1-D data vector, blockSize = (Rs * Cs) / n
        iii. Extract the block from the target data matrix, block = dataMatrix[Rs : (Rs + blockSize – 1)]
        iv. Increment the seed value, seed = seed + blockSize
    b. Else
        i. Set the seed value of seed, [Rs, Cs] to [1, 1]
        ii. Compute the block size of N-D data vector, blockSizeRows = Rs / Nr
        iii. Compute the block size of N-D data vector, blockSizeCols = Cs / Nc
        iv. Compute the number of planes in the given data matrix, denotes depth
        v. Iterate for each plane defined with depth, number of rows and number of columns
            1. Extract the block from the target data matrix, block = dataMatrix [Rs : (Rs + blockSizeRows – 1)] [Cs : (Cs + blockSizeCols – 1][depth]
            2. Increment the seed value, Rs = Rs + blockSizeRows
            3. Increment the seed value, Cs = Cs + blockSizeCols

c. Return the divided blocks.

B. Server Selection

After the approval of cost, the system also selects the servers on which the divided parts are to be stored. Server Selection algorithm 2 selects the servers that can be used to store the segments of data while also provide the highest level of reliability for the cost.

The Server Selection mechanism is based upon the selection of the servers among the given list of the available data centers. The Server Selection Algorithm has been defined in the following section:

Algorithm 2: Server Selection Algorithm

1. Get available cost for uploading
2. Load the server list from the disk in order of increasing reliability levels
3. Find the maximum possible cost for the upload of given segments/pieces
4. Find the minimum possible cost for the upload of given segments/pieces
5. If input cost (step 1) is lesser than the minimum cost (step 8)
    a. Return the error
6. If input reliability is lesser than the reliability of top server on the list
    a. sL=[1 2 3 ..n]
7. Else if input reliability is higher than top server and lesser than the second server on the list
    a. sL=[1 2 3....n-1]
8. Run the iteration till we get a final Server List sL[ ]
    a. if counter>=countCap
        i. counter=countCap;
        ii. inc=0;
    b. if j==1
        i. servOut(countCap,2)=servOut(countCap,2)+1;
        ii. j=j-1;
    c. elseif (((totRotation-j)+1)*serverList(counter+inc,1)) <= availCost
        i. if(servOut(1,2) + servOut(2,2) + servOut(3,2) + servOut(4,2)) < totRotation
            1. availCost=availCost-(serverList(counter,1));
            2. servOut(counter,2)=servOut(counter,2)+1;
        ii. else if (servOut(1,2) + servOut(2,2) + servOut(3,2) + servOut(4,2)) < totRotation
            1. availCost=availCost-(serverList(sL(end),1));
            2. servOut(sL(end),2)=servOut(sL(end),2)+1;
9. Return the Server List.

Here, servOut is a table which contains the list of all the data centers or servers and also the number if segments/parts are to be stored in that particular server. countCap is the total number of data servers available for uploading data. availCost is the cost of all the servers selected for uploading the data.

III. PERFORMANCE ANALYSIS AND RESULTS

The proposed model has been tested for the performance while uploading and downloading the files of various sizes on the simulation platform. The upload and download parameters have been obtained in the form of the upload time, download time, encoding time, decoding time, total upload time and total download time. The comparison of the proposed system is also done with the existing system and below is the visual representation of the comparison. As it is visible from the figure 2 and 3that the proposed system is as good as if not better than the existing system while also

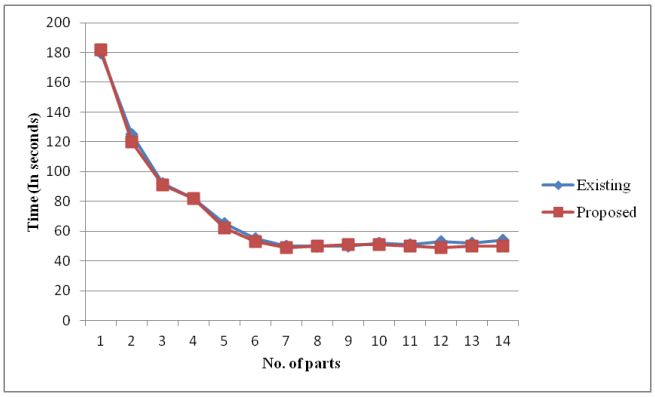

Figure 2: Comparison of Total Upload Time

adding the ability for the user to input the budget and reliability requirements.

The upload time of 67 seconds has been recorded on an average for the existing model, which is slightly higher than the time of 64.86 seconds for the proposed model. The encoding time of 6.64 seconds (existing) has been outperformed by the 6.21 seconds in the proposed model. The proposed model has been proved to be efficient in terms of the encoding time and upload time, eventually, the comparative analysis based upon the total time clearly shows the higher performance of the proposed model (Average 71.07 seconds) against the existing model (Average 73.64 seconds).

In the presented framework analysis of performance is done for reconstruction of the file from different Cloud service providers. The different steps performed during reconstruction the file are retrieving, decoding and combining the different parts to get original file or data.

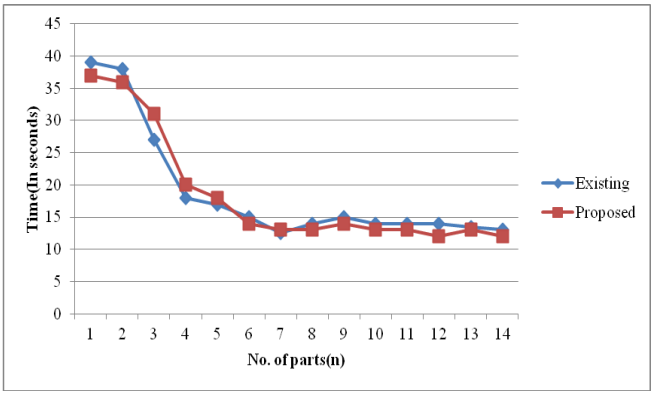

Figure 3: Comparison of Total Download Time

The total download time for the proposed model (Average 17.46 seconds) has been found improved than the existing model (Average 18.64 seconds), which improves the performance of this papers framework. The proposed model consumes nearly 13.43 seconds for the downloading of the data in comparison with the 14.07 seconds for existing model. The time of 4.57 seconds has been taken by the existing data storage model, which is outperformed by the proposed model (Average 4.04 seconds) on the basis of decoding time.

IV. CONCLUSION

This paper uses Information dispersal algorithm popularly known as the Reed Solomon encoding scheme for securing data stored on Cloud servers. The online data storage models, specifically Cloud-based storage models are becoming popular every year and adding the higher number of users with gaining popularity. There are several reasons, which cause the error bits in the online data storage systems, such as signal noise during the data transmissions, computational errors, data formation and conversion errors, etc. The error-correction becomes the mandatory operation for the online data storage systems, where the error-correction codes are used to increase the reliability of the proposed online storage based model. The reliability and cost become the primary factors for the online storage models, where the reliability depicts the accuracy of data retrieval mechanism and cost becomes the critical factor, which decides the opinion of the user to use or refuse the online storage models. The models with high cost or error-print cost models with false cost computations can decrease the popularity of the online storage model. In this paper, the framework has been designed for the higher level of reliability with optimized cost factor that takes less upload/download time. It also improves the network performance and provides the error-free service under optimal cost for the online data storage systems.

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.