ISSN: 2277-9655
Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

SECURE DATA STORAGE IN MULTI CLOUD ENVIRONMENT USING APACHE HADOOP

Lalitha Singh*, Prof. Jyoti Malhotra, Prof. Sayalee Narkhede

Department of Information technology, MIT College of Engineering, Pune, India

DOI: 10.5281/zenodo.51006

ABSTRACT

Cloud computing and Hadoop has become a new distributed storage model for most of the organizations, Industries etc. It provides a pay per use model in which customer has to only pay for the data he is storing on the cloud. However, relying on a single cloud storage provider has resulted in problems like vendor lock in. Therefore multi cloud environment is used to address the security and the data availability problems. In this paper, we proposed a system that uses Hadoop computing platform in multi cloud domain for storing customer's data reliably. Hadoop is used to conquer single point of failure problem which has been main issue in centralized environment as well as to deal with remote uploading.

KEYWORDS: cloud computing; hadoop; multi cloud; encryption; security

INTRODUCTION

Now-a-days most of the organizations and industries are using pay per use business model known as cloud for storage purposes. Cloud storage provides reliable, scalable and secure data storage. It provides the users with low maintenance cost and broad network access. In cloud computing users do not store the data on their own servers, the data is stored on the cloud storage provider's servers. User has to pay the cloud storage provider based on the amount of the data he needs to store for a particular period of time.

However, relying on a single cloud storage provider may lead to single point of failure and vendor lock-in. Vendor lock-in is relying on a single service provider with substantial switching costs. Also different cloud storage providers have different pricing policies and vary in the performance. For instance, Amazon s3 charges more for storage space, Google charges for bandwidth consumption. Every cloud storage provider has its own infrastructure which leads to variations in the pricing and the cloud performance. Therefore it is advisable to store the data on different cloud storage providers based on the storage requirements, data that needs to be accessed often must be stored on a cloud which offers less price for bandwidth access and data which is only for backup purpose must be stored on a cloud which charges less for storage.

ISSN: 2277-9655
Impact Factor: 3.785

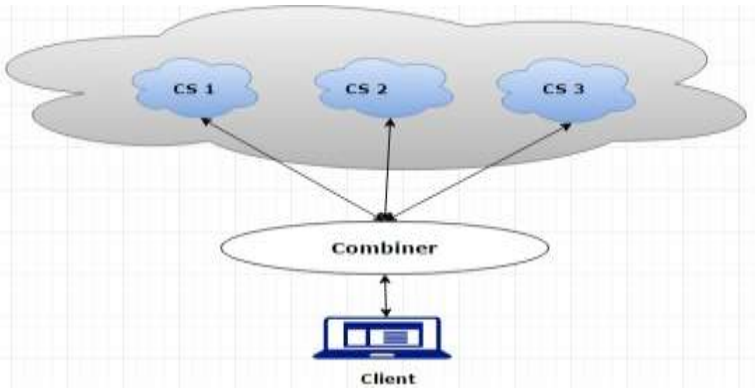

Fig. 1. Multi Cloud Environment

To provide users with data availability, less cost and secure storage it is advisable to store the data on multiple cloud storage providers. A multi cloud approach is one in which a user uses two or more clouds for storage purpose thereby minimizing the data loss and the risks of single point failures. Fig 1.Shows a multi cloud environment in which there is a combined platform for multiple cloud storage providers. As multi cloud is client centric model it gives a choice to the users to use the clouds that are better suited for their requirements. Here, two main issues that occur during data storage on multi cloud are handled. First issue is data Unavailability and Load balancing. To tackle with data availability Apache Hadoop is used. Hadoop is an emerging area in cloud storage as it handles large volume of unstructured data i.e. Big Data[11]. Hadoop is scalable distributed platform with low maintenance cost. Hadoop divides large data into smaller chunks and distribute them over multiple servers. Target of Load balancing technique is to provide optimized resource use, reduced cost. Second issue is to provide security to the data placed on multi cloud. Shared key encryption technique is used for providing privacy and confidentiality. Rest of the paper is divided in three sections. Related work is presented in section 2. Section 3 describes design issues and need of the system. Details of proposed system are given in section 4. Finally conclusion is drawn in section 5.

RELATED WORK

In the past few years, despite of having a strong infrastructure the cloud storage providers have experienced service outage and privacy breach [2], [3]. Amazon services had experienced service outage which had taken down many companies [1]. There can also be chances of experiencing vendor lock-in in which it might be difficult to switch from one storage provider to another [4]. To avoid such disadvantages many works [4] [5] [6] [7] have used multi cloud storage for the storage purposes. These works split the data block into pieces and place them on different cloud storage providers based on the user's requirements.

RACS [4] used erasure coding in multi cloud to store the data on multiple clouds to avoid vendor lock-in. RACS (Redundant Array of Cloud Storage) divides the data and places the data among different storage providers. Instead of using full replication it employs erasure coding techniques to avoid the overhead of full replication. RACS proxies are designed to act as an interface between the client application and a set of n cloud locations. Apache zookeeper is used to provide the synchronization between these proxies. RACS attributes to performance by running the cloud repositories concurrently. It focuses on the economic failures and does not provide any solution for service outage. Moreover, it does not provide any security to the data and does not protect the data from corruption and loss. µLibCloud [7] tried to reduce the access latency and provide uniform access.

HAIL [5] uses replication techniques to provide data availability and uses cryptographic and encoding techniques to ensure security of the data. It protects the data on the servers from mobile adversary. It ensures availability of the data by protecting it from malicious entities using error correction algorithms. However, it guarantees security of the static data only. It does not deal with dynamic data which has multiple versions.

SCALIA [8] places the data based upon the access pattern statistics. It is placed at the client's side thereby reducing any network latency. It provides an adaptive data placement scheme which places the data on different cloud storage providers based on different access patterns and provides data placement solution for different cloud situations. But, Scalia does not take into consideration the access latency overhead while calculating the optimal cost.

DEPSKY [9] improves the availability and security of the data in the cloud using encryption, encoding and replication of data. It provides different protocols as Depsky–A, Depsky – CA to provide availability and the

confidentiality of the data. The read access latency is also reduced. However, it does not focus on reducing the cost of the data placement and also service outage.

TRIONES [10] use erasure coding techniques to store the data on the multi cloud. It reduces the access latency and improves the fault tolerance. It uses nonlinear programming for the data placement optimization.

MOTIVATION

In our paper, two types of design issues are considered: Unavailability of the data and security of data on cloud. First design issue is unavailability of the data, if there is a single point failure in the cloud storage provider then the data stored on that cloud will be lost. Therefore, the customers should not rely on as single cloud storage provider. Let us consider an example given in Fig 2.

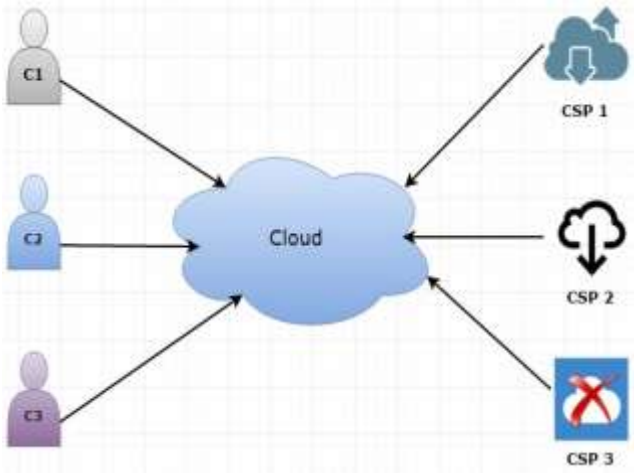

Fig 2. Cloud Storage Failure

There are three cloud storage providers and there are three customers who store their data on different clouds. There occurs an internal failure in CSP 3 then the data placed on that cloud would be lost. One Solution to this problem is that the customer will store his data on multiple cloud storage providers to ensure better availability of the data. Second design issue is that malicious insiders can intrude the cloud and can corrupt the data. Solution to this problem is to place the data of the customer by encrypting and dividing the data and placing the parts of the data on different cloud storage providers. Therefore, even if the intruder gets a part of data he will not be able to make any meaning out of the data. Fig 3 shows that a malicious insider in cloud storage provider CSP3 has control over a part of data, for security purpose that data is encrypted and rest of parts of the data is placed into different clouds so even if one part of the data is compromised it will not give any meaningful information. But again there is one drawback in this solution. Scenario is if we divide data into three splits and store these splits on three different CSPs. If any one CSP fails then there is a loss of part of the data stored on that cloud. Solution to this problem is to replicate the data on multiple servers in cloud. This can be done by using Hadoop which has the ability to replicate chunks over distributed servers; hence, providing better availability of data.

ISSN: 2277-9655
Impact Factor: 3.785

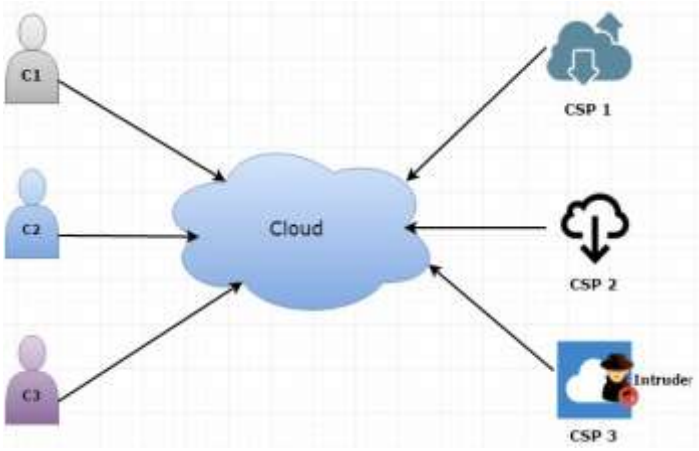

Fig 3. Malicious Insider

SECURE DATA STORAGE IN MULTI CLOUD

A. Architecture of The System

Fig 4 shows the architecture of the system for securely storing data on multi cloud servers. There are three entities are involved in the process for securing the data – client, trusted third party, multi cloud servers. Client uploads encrypted file on cloud servers so that no other entity can get the information out of it. Trusted third party generates shared key and uses shared key encryption for encrypting the file which is to be stored on cloud. In a way Trusted third party acts as a cryptographic server and is responsible for shared key generation, encryption.

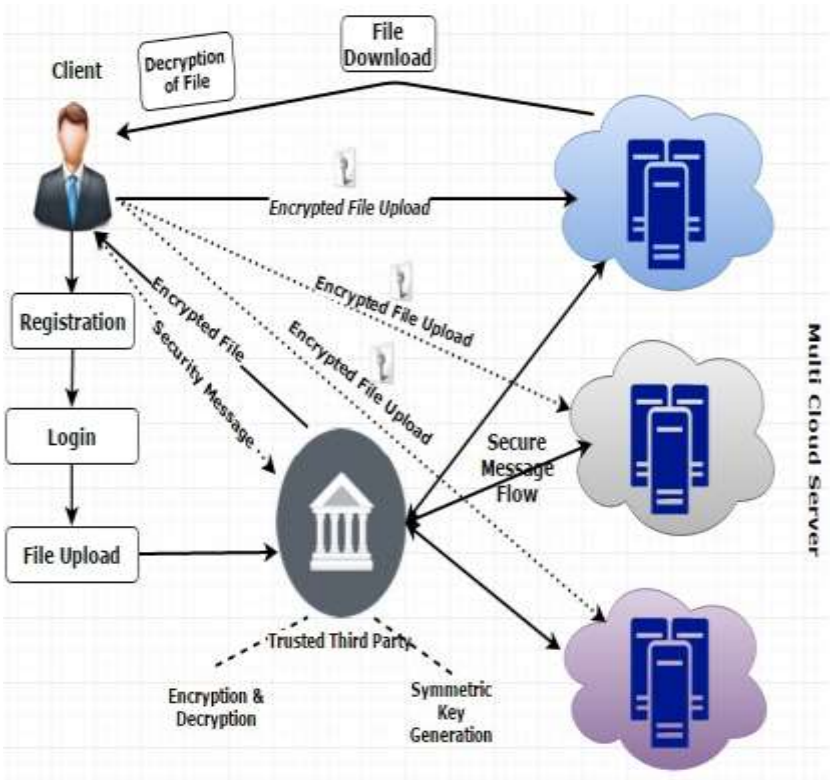

Fig. 4: Architecture Diagram for Secure Data Storage

Data Storage can be done by splitting the encrypted data in smaller chunks and distributing chunks over multi cloud serves. Here comes Hadoop in picture. Hadoop Distributed File System is the best storage platform in order to store

client's files reliably. HDFS divides file into smaller chunks called as block. Each block is replicated and stored on multiple nodes in a Hadoop cluster to avoid loss of data in case of node failure.
When the user wants to download the data block the split pieces are combined and then used for the successful retrieval of the block. The data block can be successfully retrieved only when it is decrypted using the secret key.

B. System Design

The data is encrypted using the Algorithm I given in Table 1. Once the encryption of files is done; files can be divided into n number of blocks and then can be placed on the cloud.
Each of the users is provided with a set of P cloud storage providers. Each cloud storage provider is associated with a cost C_i to store single data unit on the i^{th} cloud storage provider. Customer stores the a_i data units on the i^{th} storage provider.
The total cost for storing a_i data units on the i^{th} storage provider is

$$C_{Total} = \sum_i^P a_i C_i \quad (1)$$

Since a_i is the number of data units stored on the CSPi, Therefore

$$\sum_{i=1}^P a_i = N \quad (2)$$

Secure data placement tries to minimize the total cost of the data placement, minimize C_{Total}.
A storage variable S is included, where

$$S_i = \begin{cases} 1, & if\ a\ data\ piece\ is\ alloted\ to\ CSPi \\ 0, & if\ there\ is\ no\ data\ alloted\ to\ CSPi \end{cases} \quad (3)$$

Where i = 1, 2, 3,….P.
Our objective is to minimize the cost of the data placement.

Minimize $\sum_{i=1}^P (s_i * C_i)$
Subject to $\sum_{i=1}^P S_i = N$
And $0 < a_i < q$ (4)

Where q is the number of data pieces required to retrieve the data from the cloud storage provider.

Table 1. Encryption Algorithm

Algorithm I : Encryption Algorithm Encrypt(File) **Input:** File to be encrypted **Output:** Encrypted output file
1. **Begin** 2. f_{in} = *inputfile* 3. f_{out} =*output file to be generated(null string)* 4. *key = k* 5. *k = generateDesSecret();*///generate the secret key 6. **if** 7. mode ==Encrypt 8. Encrypt the text using cipherInputStream 9. **else if**

10. *mode == Decrypt*
11. Decrypt the text using cipherOutputStream
12. the output file is stored

Data placement is modeled into three modules- (a) selection module, (b) security module and (d) storage module.
(a)The selection module will first get the underlying cloud storage providers. Matrix M is initiated with the available number of cloud storage providers. As M is global, synchronization is needed so as to avoid incorrect optimization. Use a Boolean flag B_f which returns true or false to indicate whether updating and deleting actions are being done successfully or not. The selection module then gets the entire data placement configurations based on the rules and conditions of the user.
(b) Encrypt files using the encryption algorithm in the security module using a secret key.
(c) The storage module is responsible for interacting with the cloud storage provider and transfers the data blocks on to the cloud.
Algorithm II given in Table 2 provides the data placement configuration for placing the data securely on the cloud. The matrix M will get the cloud providers and store them (line 2). The requirements and the rules for placing the data by user are stored in rules.cfg file which is a configuration file. This configuration file rules.cfg is taken into consideration while placing the data. Based on the rules specified by the user a data placement scheme is used to place the data (lines 4~6).

Table 2. Data Placement Algorithm

Algorithm II : Data Placement Algorithm **Input:** Files to be uploaded Number of cloud storage providers **Output:** Optimized data placement on Hadoop
1. **Begin** 2. *M ← Number of cloud storage providers available* 3. *F ← rules.cfg // get configuration rules* 4. dataPlacement ← getDataFrom(F) 5. X ← getDataPlacement(dataPlacement) //all the configurations of data placement 6. Insert file F_p into the cloud provider based on X 8. encyptedData ← Encrypt(F_p) 9. data ← splitAndMerge(encryptedData) 10. *upload.OnCloud(data)*
//file is split and updated on the cloud storage providers as per the rules of Data Storage Configuration rules discussed in next section.

C. Data Storage Configuration Rules

In our system, Hadoop cluster contains Small File Pool (SP) and Large File Pool (LP). LP contains number of containers C_n. Threshold value (T_L) is considered to decide whether to divide file into blocks. If file size (F_{size}) is smaller than T_L then store that file into SP else split that file into smaller blocks B_i. Each block B_i is of equal size. Store these B_i in containers C_n of LP. To access small files quickly SP is used. Blocks are stored in containers till each container C_n reaches its threshold level T_c. Threshold level is set for containers in order to balance load among all containers. HDFS rebalancer works fine here. To move data from one node to another or to replicate the data it checks T_c of containers. If free space falls below T_c balancer will move block of data to that particular container.

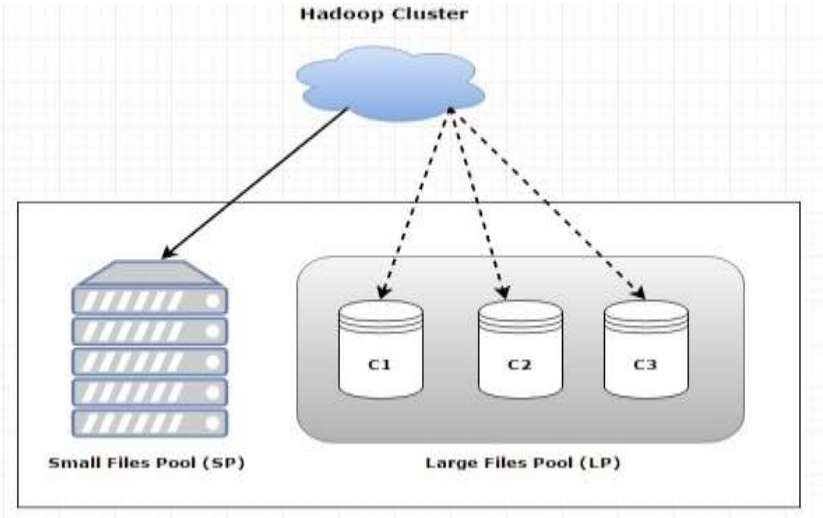

Fig. 5. Distribution of Blocks over Hadoop Cluster

Following are the rules to place data block on Hadoop cluster.

F_{en} – *Encrypted File*
F_{size} - *Size of encrypted file*
If $F_{size} < T_L$
UploadTo SP(F_{en})
Else
Divide F_{en} *into* $B_1 \ldots\ldots B_n$
UploadToLP($B_1 \ldots\ldots B_n$)

In order to accumulate the storage details such as (users, Files uploaded, Files deleted, Files Distribution over the pools and containers and authentication/authorization), we compared two scalable table stores- Apache accumulo[15] and hbase[16]. Both are open-source, apache implementations of scalable distributed storage and perform well. Storage formats and caching codes are very similar. Both have good Map-reduce integration. Fig 6 shows the search trends of Hbase and Accumulo used by the customers.

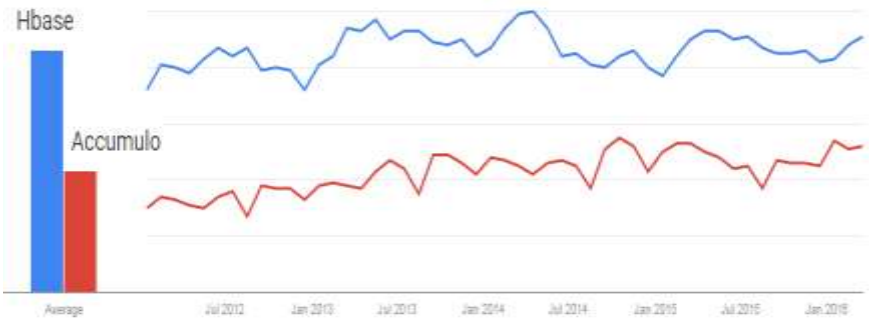

Fig 6: Users Trends of using Hbase and Accumulo

From Fig. 6, it is clear that Hbase is more trending search as compared to Accumulo; but latter provides improved authentication and cell-level security that enables fine-grained data access control and strong security mechanism.

CONCLUSION

In this paper an efficient method of securely storing data on multi cloud framework is proposed. Proposed Method allows users to make storage decision in the cloud with low cost, also it provides high availability of data and efficient resource utilization with the use of Apache Hadoop. Distributing data over multiple servers in cloud overleap single point of failure of centralized system. Shared key encryption algorithm used for security of user's data achieves two security principles like confidentiality and privacy. It provides a solution to the vendor lock-in in a cost effective way, which is good for critical applications. Access Latency and storage constraints are not being discussed here.

REFERENCES

[1] Amazon.com, "Amazon s3 availablity event: July 20, 2008", Online at http://status.aws.amazon.com/s3-20080720.html, 2008
[2] M. Arrington, "Gmail Disaster: Reports of mass email deletions", Online at http://www.techcrunch.com/2006/12/28/gmail-disasterreports-ofmass-email-deletions/, December 2006
[3] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors", Online at http://www.techcrunch.com/2008/-7/10/mediamaxthelinkup-closes-itsdorrs/, July 2008
[4] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: a case for cloud storage diversity," in Proceedings of the 1st ACM symposium on Cloud computing (SoCC'10). ACM, 2010, pp. 229–240
[5] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in Proceedings of the 16th ACM conference on Computer and communications security (CCS'09). ACM, 2009, pp. 187–198
[6] A. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," ACM Transactions on Storage (TOS), vol. 9, no. 4, p. 12, 2013
[7] S. Mu, K. Chen, P. Gao, F. Ye, Y. Wu, and W. Zheng, "_libcloud: Providing high available and uniform accessing to multiple cloud storages," in Proceedings of ACM/IEEE 13th International Conference on Grid Computing (GRID). IEEE, 2012, pp. 201–208
[8] T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: an adaptive scheme for efficient multi-cloud storage," in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12). IEEE Computer Society Press, 2012, p. 20
[9] A. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," ACM Transactions on Storage (TOS), vol. 9, no. 4, p. 12, 2013
[10] Maomeng Su, Lei Zhang, Yongwei Wu, Member, IEEE, Kang Chen, and Keqin Li, Fellow, IEEE, "Systematic Data Placement Optimization in Multi-Cloud Storage for Complex Requirements," IEEE Transactions on Computer, pp. 1-1, 2015
[11] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Sunnyvale, California USA, vol. 10, pp. 1-10, 2010
[12] T. White, "Hadoop: The Definitive Guide", O"Reilly, Scbastopol, California, 2009
[13] M. Ali, R. Dhamotharan, E. khan, S. Khan,A. Vasilakos, K. Li, A. Zomaya, "SeDaSC: Secure Data Sharing in Clouds", IEEE Systems Journal pp. 1-10 , 2015
[14] Shigeru Imai; Stacy Patterson; Carlos A. Varela, "Cost-Efficient High-Performance Internet-Scale Data Analytics over Multi-cloud Environments", 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 793-796, 2015
[15] "Apache Accumulo", https://accumulo.apache.org/
[16] "Apache Hbase" https://hbase.apache.org/