

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320-088X

IJCSMC, Vol. 4, Issue. 1, January 2015, pg.470 – 477

SURVEY ARTICLE

A Survey on Cloud Computing Storage and Security

Priyanka Hokrane¹, Asst. Prof. Judith SherinTilsha²

Student, M. Tech (Software Engineering), NHCE, Bangalore, India¹

Assistant Professor, Information Science & Engineering Department, NHCE, Bangalore, India²

Priyanka.cs045@gmail.com ; Tilsha.jack@gmail.com



Abstract- Cloud storage provides the solution for on demand remote backup to users. To provide the fault tolerance and security for cloud storage many recent techniques have been proposed. When the cloud fails permanently it should be effectively recovered. But in the cloud environment maintenance of stored data in a secure manner is not an easy task that is the stored data is not completely trustworthy. This survey shows that different techniques of cloud storage in cost effective manner to overcome from failure along with the security. The advantages and drawbacks of existing models have been discussed.

Keywords: cloud computing, data storage, encoding, recover, security

I. INTRODUCTION

Cloud storage provides the solution for on demand remote backup. Using a single cloud for storage may have many problems such as single point failure and also other problems like vendor lock-ins. One solution to this problem is to stripe the data across different clouds. The conventional erasure codes used for storing the data across the different clouds but this method performs well only for short- term transient failure and foreseeable permanent failure. When the cloud fails permanently it is necessary to recover from the permanent failures by activating repair operations which retrieves data from surviving clouds and reconstructs the lost data in the new cloud.

During the repair operations enormous amount of data needs to be moved which may introduce high significant costs. In order to overcome from the problem of increasing cost due the amount of data being transferred over the network during the repair, regenerating code have been proposed for storing data in the distributed systems. Regenerating codes are built on the top of network coding concept in sense that nodes perform encoding operation and send the encoded data into the new cloud to recover from the failed cloud.

Network coding

Network coding is based upon the conventional routing method that is store-and-forward. In the conventional method each intermediate node simply stores and forwards the information received. Where as in the network coding each intermediate node is allowed to generate output data by encoding previously received input data. That is it allows the mixing of information. The main advantages of this method are resource efficiency, computational efficiency, and robustness.

Functional Minimal Storage Regenerating (FMSR)

Regenerating codes have been proposed to reduce the repair traffic. FMSR (Functional Minimal Storage Regenerating) codes keep the same storage size as in RAID-6 codes, while having the storage nodes send encoded chunks to the proxy so as to reduce the repair traffic. FMSR specifies three operations for a particular file object:

- 1) File upload
- 2) File download and
- 3) Repair

One property of FMSR codes is that it does not require lost chunks to be exactly reconstructed, but instead in each repair, code chunks that are not necessarily identical to those originally stored in the failed node.

II. EXISTING METHODS

The survey includes different methods which are implemented earlier. It also contains the advantages and disadvantages of each method.

A. High-Availability and Integrity Layer (HAIL)

HAIL (High-Availability and Integrity Layer) was proposed by Kevin, in the year 2008. HAIL checks the file integrity and availability across a collection of servers or independent storage services. To restore security assurance researchers have proposed two basic approaches to client for verification of file availability and integrity. The cryptographic community has proposed tools called proofs of retrievability (POR) and proofs of data possession (PDPs). HAIL manages file integrity and availability across a collection of servers which makes use of PORs as building blocks by which storage resources can be tested and reallocated when failures are detected.

A POR uses file redundancy within a server for verification. In a second, complementary approach, researchers have proposed distributed protocols that rely on queries across servers to check file availability. In a distributed file system, a file F is typically spread across servers with redundancy—often via an erasure code. Such redundancy supports file recovery in the face of server failures. A POR enables a prover i.e. cloud-storage provider to demonstrate to a client that a file F is retrievable without any loss or corruption. PORs are mainly useful in environments where F is distributed across multiple systems, such as independent storage services. In such an environment file F is stored in redundant form across multiple servers. A user can test the availability of file F on individual servers via a POR. If it detects corruption within a given server, it can appeal to the other servers for file recovery.

1) HAIL Overview

The first idea behind this method is to replicate the file F along different n number of servers. Integrity check is performed by using redundancy clients. Simply choose a random file block F_j of file F from each server. If inconsistencies found it will constructs from the remaining replicas of file F .

2) HAIL Design Strategy: Test and Redistribute (TAR)

HAIL uses protocol called TAR (test and redistribute) along with TAR client uses POR (proofs of retrievability) to detect file corruption and then trigger reallocation whenever files are required. When the fault occurs in one server then it communicates with other surviving servers in order to recover the failed file.

3) HAIL Mainly Uses Three Codes for its Construction

➤ Dispersal code

In the HAIL method we use the dispersal coding for robustly spreading file blocks across different servers. It is the cryptographic method which makes use of the IP-ECC code (Integrity Protected error-correlating code).

➤ **Server code**

All the file blocks in each of the server are additionally encoded using an error-correlating code called server code. This code helps in the small or low level kind of failures when integrity fails upon checking.

➤ **Aggregation code**

Aggregation code is used in the HAIL in order to compress the response from the different servers when it needed by the client.. HAIL method provides Strong file intactness, Low overhead, strong adversarial model, and supports direct client-server communication but in HAIL Client can only feasibly inspect small portion of F and the time to recover and storage is more. HAIL does not provide the security assurance well.

B. Online data storage using implicit security

This method is proposed by Parakh A and Kak S in 2009. In traditional or explicit method the data is stored on a single server for back-up and allows the access upon the use of passwords and this password needs to be changed frequently. In explicit method there is a tendency that the user keeps simple and memorable passwords leading to the possibility of brute force attacks. Furthermore data on the web is archived therefore explicit security architecture may not be useful for many applications.

The use of implicit security for online data storing in the cloud computing environment is advantageous, in which security is provided among many entities. In this approach the stored data is portioned into two or more pieces which are stored at randomly chosen servers that are known only to the owner of data. Access to these pieces requires the knowledge of passwords and also the knowledge of location of pieces. The partition is made in such a way that knowledge of all pieces is required to recreate the original data. None of the individual pieces does not reveal any of the useful information. Online data storage using implicit method provides security over explicit method which is weightless and more expeditious. In case if the user forgot where the data is stored it becomes difficult and also if one of the data piece is lost recreating the original data is not possible.

C. Redundant Array of Cloud Storage (RACS)

This system is proposed by Hussam Abu-libde in the year 2010. Redundant Array of Cloud Storage (RACS) transparently stripe data across multiple cloud storage. RACS reduces the cost of switching data from one storage provider to another storage provider. It also includes additional operations such as put, get, delete and list. RACS exposes its interface to its user i.e. the media by which users are communicating with storage provider.

RACS mainly focuses on the problem of vendor lock-ins which reduces the storage vendor lock-ins which can be explained as follows. Some cloud storage provider requires high price where as some require low price and also some storage provider offers discounts on storing high amount of data where as other may provide attractive features to clients. In all these cases clients wants to switch from one storage provider to another to take the advantage of other cloud. In some cases the cost required to switch may be more than cost of storing therefore clients may not switch in such cases. In order to overcome from the problem of switching RACS provides the striping of data across different clouds so that only fraction of data is needed to move. The information about features of all storage providers are known to clients by keeping track of features of different cloud.

1) Design of RACS

RACS uses the same data model as that of the amazon S3. In S3 data are stored in the buckets; each bucket has objects and key associated with it. The nesting of buckets is not allowed and objects can have arbitrary size limited up to 5 Gigabytes. In the buckets partial reads are allowed where as partial writes are not allowed.

RACS itself present the proxy called RACS proxy which is the interconnection between the client applications and set of n repositories. Upon receiving the put request from the client RACS divides these new objects into equal size m, where $m < n$ and it uses the erasure coding to create an additional $(n-m)$ redundant shares. Similarly when the get request is made by the client the m shared data will be fetched and reassembles all the data and metadata such as bucket names, key names, MIME times and modification times are replicated across servers.

2) Distributed RACS

In RACS all the data must pass through the RACS proxy. Using a single proxy may become bottleneck to avoid this problem RACS are provided with distributed systems which allows the concurrent communication of RACS proxy with each other. The RACS proxy provides states information such as location information, credentials and user authentication for each repository.

Whenever the state of information is changed this information will broadcasts to other all RACS proxies. The apache zookeeper is used as distributed systems along the different RACS proxies.

3) *Failure Recovery in RACS*

When the repositories fail in the RACS they must be recovered without any loss of data. The economic failures for example increase in the price of storage providers will be known in advance to clients. In such cases RACS moves data from failed repositories to new repositories. When repository fails RACS will not accept the get request from client to other repositories and put requests are redirected from the failed repositories to new repositories. RACS manages the concurrent failure and keeps the information of all storage providers. In this method it is not always known that when the cloud fails so always it doesn't support recovery.

D. *Dependable and Secure Storage in a Cloud-of-Clouds (DEPNISKY)*

DEPNISKY (Dependable and Secure Storage In a Cloud-of-Clouds) method proposed by Alysson Bessani Miguel Correia and Bruno in the year 2010. The combination of different clouds to build a cloud-of-cloud is called virtual storage. As the third party will be involved in the cloud many issues will present which needs to overcome.

1) *Design Models*

➤ *Registration of Clients*

The details of clients who want to store their data in the cloud will be collected in the server cloud before entering into the multicloud. Once the authentication is proved clients will be allowed to stores their files in the multiclouds.

➤ *Security Provider Cloud*

In this cloud the registration of user who wants service from the clouds will be stored. The registration will be done in this cloud by logging into it. Where the user name and password will be provided to users. Users are then verified. Only the verified users are allowed to access. During registration if the non-authentication user found they are blocked by this cloud.

➤ *Service Provider Cloud*

This cloud is responsible for providing only service to users which is handled by security provider cloud. Once the user login approval is done by the security provides cloud it provides services to requested users.

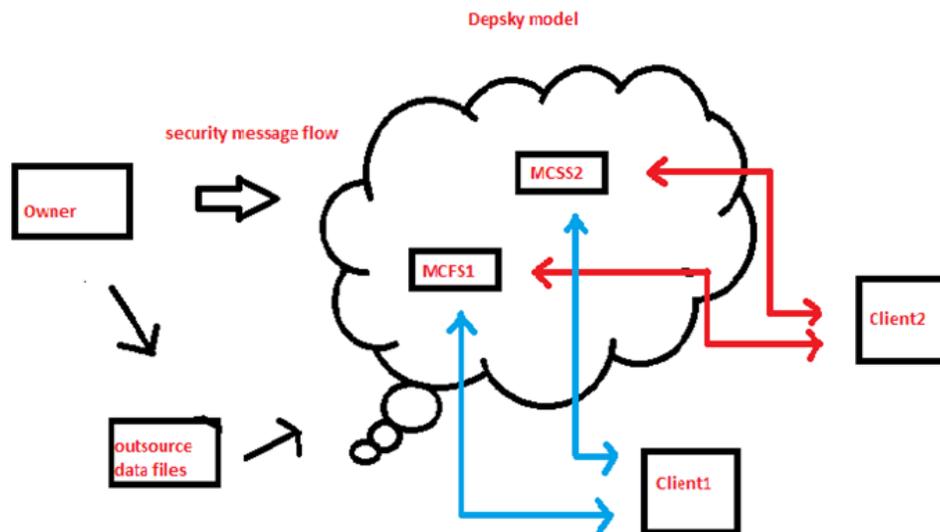


Figure. 1 Design modules

2) **DEPNSKY Architecture**

The DEPNSKY architecture consists of storage clouds and two other components called readers and writers which are the task of clients. The reader can go wrong randomly where as the writer can go only crashing.

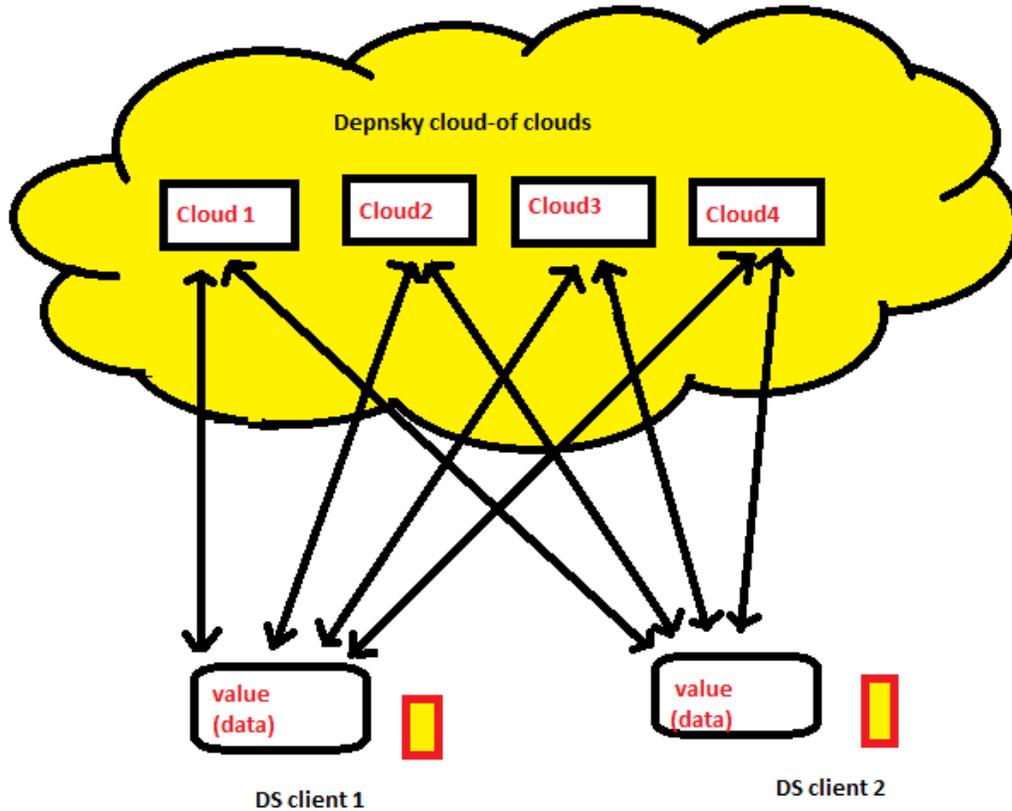


Figure . 2 The DEPNSKY architecture

3) **Data Integrity**

For checking integrity hash functions are used, the user will divide the file into numerous blocks. At any instance of time this will be stored into the multicloud. Multicloud keeps the track of number of time the file is corrupted for user and also the number of times the files are corrupted in the clouds. If the number of times the file corrupted for user it concludes that the authentication of the user has been leaked similarly the failure of file in the cloud storage. If the reputation of service is found than it specifies to administrator to use firewall and advance security method. The DEPNSKY maintains availability and confidentiality where as it is difficult to distribute the data when it is large.

E. Distributed access control in clouds (DACC)

Distributed access control in cloud (DACC) is proposed by sushmitaruj, amianayak and ivanstojmenvic in 2011. Earlier work assumed that the owners encrypt data with the attributes they have and send it to the cloud to store them securely. The owners give not only the access rights to the user but also the attributes and corresponding secret keys. Therefore key distribution centres (KDCs) are not required. The main problem involved here is the user requires different secret keys from

different owners for same attribute this increases the total no of secret keys given to the user, which in turn increases the storage and communication overhead.

DACC (Distributed access control in clouds) model provides the data storage and access in clouds. DACC scheme avoids the storing of multiple encrypted copies of the same data. To provide the secure data storage the cloud stores encrypted data. The main novelty of DACC is addition of key distribution centres (KDCs), where one or more KDCs distribute keys to data owner and users. KDCs provide access to particular fields in the record. The users and owner are assigned to certain set of attributes. The owner encrypts data with attributes assigned and stores the data in clouds. The user with matching attributes access the data from the cloud. DACC reduces the overhead involved in communication, computation and storage but the identity of owner and user are not hidden which leads to security issues.

F. Redundant Array of Independent Disks (RAID-6)

This method proposed by reed Solomon in the year 2012. As the demands on the cloud storage such as capacity, speed and reliability are increasing more techniques have been proposed to achieve these requirements.

One of the method used is RAID-6 which has many advantages over RAID-5. The RAID-5 is capable of recovering when one disk fails where as RAID-6 is capable of recovering when two disks fails concurrently.

1) Overview

RAID-6 consists of 4 data drives and 2 parity drives. It calculates the maximum number of codewords for a given symbol. For example consider a symbol of size s , the maximum length of codeword is given by $n=2^s-1$. Consider the symbol of 8 bits then the maximum no of code words are $2^8-1=255$. The implementation of P+Q RAID-6 forms a codeword by taking single bytes from each of K data disks in the array along with two parity bits of it. For example a symbol of size 8 forms 255 codeword which includes 253 data drives and 2 parity drives.

2) P and Q Check Value Generation in RAID-6

When the data is being striped on to the disks P and Q check values are generated. Whenever the data is written or updated new check values must also be generated.

If P and Q elements are always mapped to the same two disks then every write operation done by the array need to update the same two disks in order to avoid this problem of bottleneck, the P and Q values are rotated through the disks. Consider one possible layout of rotating checks. For striping of data 0 the 1 to 4 drives are used where as the drive 5 and 6 are used for parity respectively, similarly for striping of data 1 disk 0 to 3 and 6 are used where as the drives 4 and 5 are used for storing parity bits P and Q. This will be continued until P and Q values reach their original position.

stripe	0	1	2	3	4	5	6
0	D(0,0)	D(0,1)	D(0,2)	D(0,3)	D(0,4)	P(0)	Q(0)
1	D(1,0)	D(1,1)	D(1,2)	D(1,3)	P(1)	Q(1)	D(1,4)
2	D(2,0)	D(2,1)	D(2,2)	P(2)	Q(2)	D(2,3)	D(2,4)
3	D(3,0)	D(3,1)	P(3)	Q(3)	D(3,2)	D(3,3)	D(3,4)
4	D(4,0)	P(4)	Q(4)	D(4,1)	D(4,2)	D(4,3)	D(4,4)
5	P(5)	Q(5)	D(5,0)	D(5,1)	D(5,2)	D(5,3)	D(5,4)
6	Q(6)	D(6,4)	D(6,0)	D(6,1)	D(6,2)	D(6,3)	P(6)

Figure.3 RAID 6 Data Blocks on Drives

3) *Recovery of RAID-6*

Recovery in RAID-6 is done when two drives fails concurrently which may be 2 parity drives, P parity drive and one data drive, Q parity drive and one data drive and two data drives.

When two parity drives fails the recovery of these two parity drives is the easiest because all data drives are available to recover. In case of P parity drive and one data drives it is recovered from Q parity and remaining data drives. Similarly when Q parity drive and one data drive fails it is recovered from the P parity drive and the remaining data drives .The recovery of two parity drives is the most difficult among all in the RAID-6. The RAID-6 has various advantages over RAID-5 which tolerates concurrent failure and has high space efficiency. But RAID-6 has high complexity overhead.

G. *EMSR (Exact Minimum-Storage Regenerating)*

One class of the regenerating code called Exact Minimum-Storage Regenerating (EMSR) codes keep the same storage size as in RAID-6 codes, But the storage nodes send encoded chunks to the proxy so as to reduce the repair traffic. The EMSR code based distributed storage systems have fixed number of redundancy nodes. An (n, k) EMSR based distributed storage system is a system having n nodes and can tolerate any $n-k$ nodes failure. Such a system is able to provide the same reliability with replication schemes using much less redundancy storage, while the bandwidth cost of failed nodes repairing is much less than erasure code systems. Exact-MSR which means that in the repairing process of a failed node, the replacement node is constrained to store exactly the same data as the corresponding failed node. EMSR has many advantages over other methods such as reduces the repair traffic. The cost of repairing the failed node is less but the replacement node needs to store exactly same data as the failed node which is difficult.

III. CONCLUSION AND FUTURE WORK

Cloud computing is a new emerging computing paradigm which provides the services to the user over the internet and allows sharing the resources and information from the pool of distributed computing system. In order to overcome from the various problems of cloud we present a regenerating method based on ncloud. A proxy based multicloud storage that addresses the problem of reliability of today's cloud storage and taking security as other important aspect of future enhancements.

REFERENCES

- [1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A Case for Cloud Storage Diversity," Proc. ACM First ACM Symp. Cloud Computing (SoCC '10), 2010.
- [2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, *Provable data possession at untrusted stores*, In ACM CCS, pages 598–609, 2007.
- [3] A. Juels and B. Kaliski, *PORs: Proofs of retrievability for large files*, In ACM CCS, pages 584–597, 2007.
- [4] A. and Kak S (2009). Online data storage using implicit security, Information Sciences, vol 179(19), 3323–3331, storage Diversity, Proc. ACM First ACM Symp Cloud Computing (SoCC '10), 2010.
- [5] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung, *Network Information Flow*, IEEE Trans. Information Theory, vol. 46, no. 4, pp. 1204-1216, July 2000.
- [6] Amazon Web Services, *AWS Case Study: Backupify*, Available: <http://aws.amazon.com/solutions/case-studies/backupify/>, 2013.
- [7] Amazon Web Services, *Case Studies*, <https://aws.amazon.com/solutions/case-studies/#backup>, 2013.
- [8] A. Bessani, M. Correia, B. Quaresma, F. Andre', and P. Sousa, *DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds*, Proc. ACM European Conf. Computer Systems (EuroSys '11), 2011.
- [9] Amazon Web Services, *Amazon Glacier*, Available: <http://aws.amazon.com/glacier/>, 2013.
- [10] Amazon Web Services, *Amazon S3*, <http://aws.amazon.com/s3>, 2013.
- [11] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, *A View of Cloud Computing*, Comm. the ACM, vol. 53, no. 4, pp. 50-58, 2010.
- [12] Asigra, *Case Studies*. Available: <http://www.asigra.com/product/casestudies/2013>.
- [13] Amazon Web Services, *Amazon S3 Availability Event: July 20, 2008*, Available: <http://status.aws.amazon.com/s3-20080720.html>, July 2008.

- [14] A. Bessani, M. Correia, B. Quaresma, F. Andre', and P. Sousa, *DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds*, Proc. ACM European Conf. Computer Systems (EuroSys '11), 2011.
- [15] K.D. Bowers, A. Juels, and A. Oprea, *HAIL: A High-Availability and Integrity Layer for Cloud Storage*, Proc. 16th ACM Conf. Computer and Comm. Security (CCS) 2009.
- [16] H. Blodget, *Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data*. Available: <http://www.businessinsider.com/amazon-lost-data-2011-4/>, Apr. 2011.
- [17] Amazon.com. Amazon simple storage service (Amazon S3), 2008. Referenced 2008 at aws.amazon.com/s3.
- [18] [Ama 2010] Amazon S3 FAQ: What data consistency model does amazon S3 employ? <http://aws.amazon.com/s3/faqs/>, 2010.
[tcl 2010] Project TLOUDS – trustworthy clouds - privacy and resilience for Internet-scale critical infrastructure. Available: <http://www.tclouds-project.eu/>, 2010.
- [19] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. *Racs, A case for cloud storage diversity*, SoCC'10, June 2010.
- [20] C. Suh and K. Ramchandran, *Exact-Repair MDS Code Construction Using Interference Alignment*, IEEE Trans. Information Theory, vol. 57, no. 3, pp. 1425-1442, Mar. 2011.