ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Survey on Efficient Data Integrity Techniques for Distributed Cloud Storage

Mr.Divyesh K Joshi*, Prof. Miss Devangini Dave, Prof. Kishori Shekoka
Sigma Institute of Engineering, Bakrol,Vadodara, India

Abstract

Recent advances have given rise to the popularityand success of cloud computing. Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. It moves the application software and databases to the centralized large data centres, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges. To protect outsourced data in cloud storage against corruptions, enabling integrity protection, fault tolerance, and efficient recovery for cloud storage becomes critical. Therefore, we study the problem of remotely checking the integrity of regenerating-coded data against corruptions under a real-life cloud storage setting. In this paper surveys the various protocols to check cloud data integrity and compares them based on the integrity requirements. Finally we compared different remote checking integrity techniques: Replication, Erasure codes (Reed Solomon Code) and regenerating code for various file operations.

Keywords: cloud computing, remote data checking, security,integrity, regenerating code, FMSR-DIP.

Introduction

Cloud computing defined as “A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically, scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over theInternet.” Cloud storage offers an on-demand data outsourcing service model, and is gaining popularity due to its elasticity and low maintenance cost.

One major use of cloud storage is long-term archival, which represents a workload that is written once and rarely read. While the stored data is rarely read, it remains necessary to ensure its integrity for disaster recovery or compliance with legal requirements. The notion of integrity in cloud computing concerns are both data integrity and computation integrity. Data integrity implies that data should be honestly stored on cloud servers, and any violations (e.g., data is lost, altered, or compromised) are to be detected. Computation integrity implies the notion that programs areexecuted without being distorted by malware, cloud providers, or other malicious users, and that any incorrect computing will be detected. So it is desirable to enable cloud clients to verify the integrity of their outsourced data in the cloud, in case their data has been accidentally corrupted or maliciously compromised by insider/outsider Byzantine attacks.

To meet the requirements of the massive volume of storage, erasure codes have gained a significant amount of attention in cloud systems.

Background

Security in Cloud Computing

The popularity of Cloud Computing is mainly due to the fact that many enterprise applications and data are moving into cloud platforms; however, lack of security is the major barrier for cloud adoption [1]. According to a recent survey by International Data Corporation (IDC), 87.5% of the masses belonging to varied levels starting from IT executives to CEOs have said that security is the top most challenge to be dealt with in every cloud service. Many of the threats found in existing platforms. Out of them, the Security Threat is considered to be of High Risk.
The major security aspect is Confidentiality, Integrity, Authentication, Authorization, Non-repudiation and Availability which are further explained below:

Confidentiality is the process of making sure that the data remains private, confidential and restrictedfrom unauthorized users [2]. Data encryption is one of the most popular options of security before pushing the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

data into cloud.

Integrity is the guarantee by which the data is protected from accidental or deliberate (malicious)modification. Hashing techniques, digital signatures and message authentication codes are used to preserve data integrity [3]. Integrity problems are in big scale due to the multi-tenancy characteristic of cloud [4].

Authentication is the mechanism by which the systems may securely identify their users. Authorizationdetermines the level of access to system resources attributed to a particular authenticated user [5].

Non-repudiation is an extension to the identification and authentication service. It is used to ensure thatthe messages sent are properly received and acknowledgements are sent back to the sender. In other words, establishing a two way communication between a sender and a receiver.
Availability ensures that an organization has its full set of computing resources available and usable at all times for its real users [8]. In this paper we will discuss about the integrity of data in cloud storage.

Data integrity proving schemes

Juels and Kaliski [8]. Proposed a scheme called Proof of Retrievability (POR). Proof of retrievability means verify the data stored by user at remote storage in the cloud is not modified by the cloud. POR for huge size of files named as sentinels. The main role of sentinels is cloud needs to access only a small portion of the file (F) instead of accessing entire file. Sravan and Saxena [7].Proposed a Schematic view of a proof of retrievability based on inserting random sentinels in the data file.

Provable Data Possession (PDP)

Definition: A PDP scheme checks that a file, which consists of a collection of n blocks, is retained by a remote cloud server. The data owner processes the information file to generate some metadata to store it locally. The file is then sent to the server, and the owner deletes the native copy of the file. The owner verifies the possession of file in using challenge response protocol. This technique is used by clients to check the integrity of the data and to periodically check their data that is stored on the cloud server. So this technique ensures server security to the client.

Naive Method:

The main idea behind this algorithm is to compare the data. In this method client will compute the hash value for the file F and having key K (i.e. (K, F)) and subsequently it will send the file F to the server. Clients are having different collection of keys and hash values so it can check multiple check on the file F. Whenever client wants to check the file it release key K and sends it to the server, which is then asked to recomputed the hash value, based on F and K. Now server will reply back to the client with hash value for comparison.

Limitation

This method gives the strong proof that server is having the original file F.But this method has high overhead as every time hashing process is run over the entire file. It is having very high computation cost.

Proof of Retrivability (POR):

Juels and Kaliski [8]. Proposed a scheme called Proof of Retrievability. Proof of retrievability means Verify the data stored by user at remote storage in the cloud is not modified by the cloud. POR for huge size of files named as sentinels. The main role of sentinels is cloud needs to access only a small portion of the file (F) instead of.

In this scheme data are divided into number of block as shown in figure 2. This technique uses theauditing protocol when solving the problem of integrity.

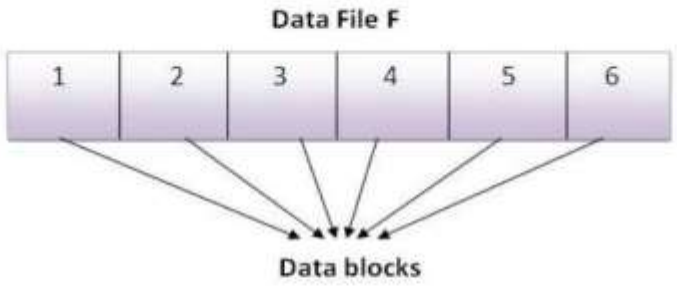


Fig 1-A data file with 6 blocks

Related work

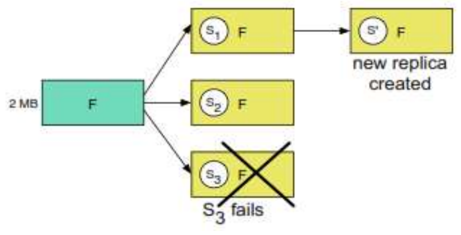
There are various study performed to check the integrity of data, which are typical in long-term archival storage systems. This problem is first considered by Juels et al. [8]. And Ateniese et al. [9]., giving rise to the similar notions proof of retrievability [8]. (POR) and proof of data possession (PDP) [9]., respectively, which are proposed to verify the integrity of a large file by spot-checking only a fraction of the file via various cryptographic primitives. The basic POR scheme [8]. Embeds a set

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

of pseudorandom blocks into an encrypted file stored on the server, and the client can check if the server keeps the pseudorandom blocks later on. Error correcting codes are also included in the stored file to allow recovery of a small amount of errors within a file. However, the number of checks that the client can issue is limited by the number of the embedded random blocks. On the other hand, PDP [9]. Allows the client to keep a small amount of metadata. The client can then challenge the server against a set of random file blocks to see if the server returns the proofs that match the metadata on the client side. These both schemes are single server storage scheme. So in these methods whole the data are stored on a single server in which single-point-failure [11]. And vendor-lock-ins [10]. Problems are arises. To overcome these problems one possible solution is to stripe data across multiple servers. Thus, to repair a failed server, we can (i) read data from other surviving servers, (ii) reconstruct the corrupted data of the failed server, and (iii) write the reconstructed data to a new server. MR-PDP [13]. And HAIL [12]. Extend integrity checks to a multi-server setting using replication and erasure coding, respectively. In erasure coding based system (e.g. Reed Solomon Code) requires less storage overhead compare to Replication based system [14]. For the same fault-tolerance level.

A. Replication Based system

Ensuring reliability requires the introduction of redundancy. The simplest form of redundancy is replication, which is adopted in many practical storage systems. In which k identical copies of each data object are kept at each instant by system members. Figure 1 shows an example of replication based system.



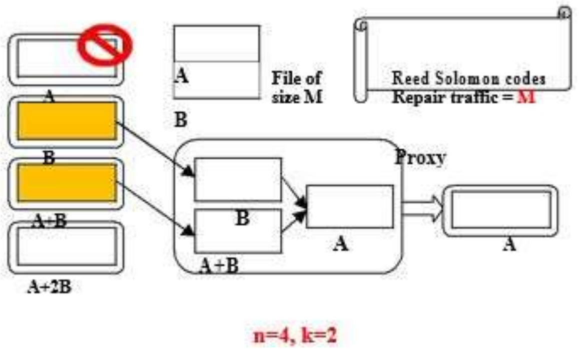
S3 fails

Fig. 1 Replication based distributed system

Simple replication offers one avenue to higher-assurance data archiving. Only single copy of file is required to repair any node. For example if any node fails then simply copy the replica of that file from healthy node and store it on new node. But it requires often unnecessarily and unsustainably high expense. The storage cost for replication based system is very high.

B. Erasure Coding Based (Reed Solomon Code) system

As a generalization of replication, erasure coding offers better storage efficiency. For instance, we can divide a file of size M into k pieces (to be called fragments), each of size M/k, encode them into n encoded fragments (of the same size) using an (n, k) maximum distance separable (MDS) code, and store them at n nodes. Then, the original file can be recovered from any set of k coded fragments. This performance is optimal in terms of the redundancy–reliability trade-off because k pieces, each of size M/k, provide the minimum data for recovering the file, which is of size M. Example of (4, 2) Erasure coding based system is shown in figure 2. In which repair traffic of the system if M which is same as our file size.



(n, k) MDS property: any k out of n servers can rebuild original file

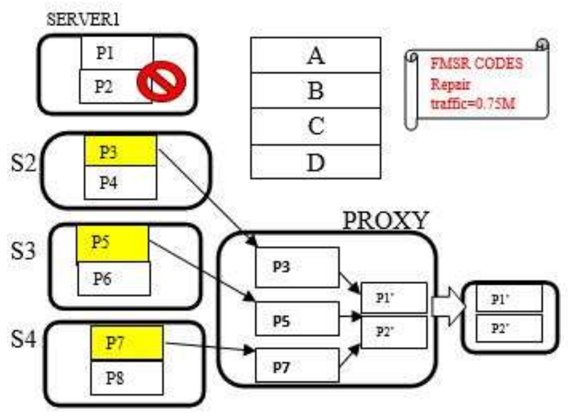
Fig. 2 Reed Solomon Erasure Coding

C. Regenerating Coding Based system

For an erasure coded system, a common practice to repair from a single node failure is for a new node to reconstruct the whole encoded data object to generate just one encoded block. This is clearly an inefficient way of regeneration, since the network bandwidth is often a critical resource. This has motivated the development of family of codes, referred as regenerating codes, designed to carry out the regeneration efficiently. Regenerating codes [15]. Have been proposed to minimize this repair traffic (i.e., the amount of data being read from surviving servers). In essence, they achieve this by not reading and reconstructing the whole file during repair as in traditional erasure codes, but instead reading a set of chunks smaller than the original file from other surviving servers and reconstructing only the lost (or corrupted) data chunks. Regenerating codes are

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

constructed systematically such that the source symbols are stored in k nodes called as data nodes, the remaining n-k nodes are called as parity nodes which contain symbols obtained through suitable encoding operation. Such systematic codes, where the data nodes are regenerated exactly but only functionally equivalent form of parity nodes are regenerated. Example of (4, 2) Regenerating code is shown in figure 3. In which repair traffic is 0.75M which is less than the Reed Solomon Erasure coding based system.



n=4, k=2

Fig. 3 Regenerating Code based system

Preliminaries

A. Functional Minimum Storage Regenerating Code

FMSR [16].Belongs to Maximum Distance Separable (MDS) codes. An MDS code is defined by the parameters (n, k), where k < n. It encodes a file F of size |F| into n pieces of size |F|/k each. An (n, k)-MDS code states that the original file can be reconstructed from any k out of n pieces (i.e., the total size of data required is |F|). An extra feature of FMSR is that a specific piece can be reconstructed from data of size less than |F|. FMSR is built on regenerating codes, which minimize the repair bandwidth while preserving the MDS property based on the concept of network coding.

FMSR codes have three design properties, which we elaborate below.

1. *Preserve the fault tolerance and storage efficiency of MDS Codes*

MDS codes are defined by two parameters n and k (k < n). An (n, k)-MDS code divides a file of size M into k pieces of size M/k each, and encodes them into n pieces such that any k out of n encoded pieces suffice to recover the original file. By storing the n encoded pieces over n nodes, a storage system can tolerate at most n − k node failures. An example of MDS codes is Reed-Solomon codes [14].

[1] FMSR codes minimize the repair bandwidth

If a node fails, we must reconstruct the lost data of the failed node to preserve fault tolerance

Reed-Solomon codes reads k pieces from any k surviving nodes to restore the original file (by the design of MDS codes). Clearly, the amount of data read is the file size M. FMSR codes seek to read less than M units of data to reconstruct the lost data. FMSR codes are designed to match the minimum storage point of regenerating codes when repairing a node failure, while having each node store M/k units of data as in Reed-Solomon codes. To repair a failed node in FMSR codes, each surviving node transfers data of size $\frac{M}{k(n-k)}$ Units or equivalently, a size of one parity chunk.

In a special case of n = 4 and k = 2, the repair bandwidth is 0.75M, i.e., 25% less than that of conventional repair of Reed-Solomon codes. In general, the repair bandwidth of FMSR codes for k = n − 2 is $\frac{M(n-1)}{2(n-2)}$, and its saving compared to RAID-6 codes [17]. (Which are also double-fault tolerant) is up to 50% if n is large.

Property 3: FMSR codes use uncoded repair

During repair, each surviving node under FMSR codes transfers one parity chunk, without any encoding operations. This also minimizes the amount of data read from disk.

B. NCCloud

NCCloud (formerly known as CloudNCFS) [18].is a proof-of-concept prototype of a network-coding-based file system that aims at providing fault tolerance and reducing data repair cost when storing files using multiple-cloud storage (or any other kinds of raw storage devices). NCCloud is a proxy-based file system that interconnects multiple (cloud) storage nodes. It can be mounted as a directory on Linux, and file uploading/downloading are done by copying files to/from the mounted directory. NCCloud is built on FUSE, an open-source, programmable user-space file system that provides application programmable interfaces (APIs) for file system operations. From the point of view of user applications, NCCloud presents a file system layer that transparently stripes data across storage nodes. Network codes for storage

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

repairrequire that storage nodes encode the stored data during the repair process. However, this may not be feasible for some storage systems where nodes only provide the basic I/O functionalities but do not have the encoding capability. Our work is to adapt the benefits of network codes in the storage repair of a practical storage setting, by relaxing the encoding requirement of storage nodes. NCCloud supports a variety of coding schemes, in particular the Functional Minimum Storage Regenerating (F-MSR) codes. Compared to traditional optimal erasure codes (e.g., Reed-Solomon), FMSR codes maintains the same storage overhead under the same data redundancy level, but uses less repair traffic during the recovery of a single failed storage node. NCCloud realizes regenerating codes in a practical cloud storage system that does not require any encoding/decoding intelligence on the cloud storage nodes.

Problems in existing system

As it is noted in the different techniques of checking integrity of cloud storage data in analysis part there are some drawbacks in the existing system, which are like not secure against byzantine mobile adversary. Mobile Byzantine means that the adversary compromises a subset of servers in different time epochs (i.e., mobile) and exhibits arbitrary behaviours on the data stored in the compromised servers (i.e., Byzantine). To ensure file availability, we assume that the adversary can compromise and corrupt data in at most n−k out of the n servers in any epoch, subject to the (n, k)-MDS fault tolerance requirement. At the end of each epoch, the client can ask for

Randomly chosen parts of remotely stored data and run a probabilistic checking protocol to verify the data integrity. Servers under the control of the adversary may or may not correctly return data requested by the client. If corruption is detected, then the client may trigger the repair phase to repair corrupted data. Instead of performing whole-file checking,which incurs a substantial transfer overhead, it is only feasible for the client to randomly sample data for integrity checking.

The adversary may corrupt a small portion of data within the Error-correcting capability in each epoch, but the level of Corruption can render the errors unrecoverable after several Epochs if they are not spotted early. This leads to creeping Corruption [12]. Thus, it is necessary that the client can Quickly spot the corrupted data without accessing the whole
File.

Comparative study of PDP techniques

To analyse all the provable data techniques, builds the table below, in which cover all that parameters on which the different schemes can be compared possession.

Table1.0 Comparative study of PDP techniques

	[8].	[19].	[20].	[9].	[21].	[22].	[23].
DATA POSSESSION	no	No	yes	yes	yes	yes	yes
SUPPORT SAMPLING	no	No	no	yes	yes	yes	yes
TYPE OF GUARANTEE	**Deterministic** **Probabilistic**						
SERVER BLOCK ACCESS	O(n)	O(logn)	O(n)	O(1)	O(logn)	O(logn)	O(logn
SERVER COMPUTATION OVERHEAD	O(n)	O(1)	O(1)	O(1)	O(logn)	O(logn)	O(logn
CLIENT COMPUTATION OVERHEAD	O(1)	O(1)	O(1)	O(1)	O(logn)	O(logn)	O(logn
COMMUNICATION OVERHEAD	O(n)	O(logn)	O(1)	O(1)	O(logn)	O(logn)	O(logn
STORAGE COST	O(1)	O(1)	O(n)	O(1)	O(n)	O(1)	O(1)
SUPPORT DYNAMIC INTEGRITY	No	No	No	No	yes	No	Yes
SUPPORTING PUBLIC AUDITABILITY	No	No	No	yes	No	yes	Yes
DATA RECOVERY	No	No	No	No	No	No	No

Conclusions

Though Cloud computing offers great potential to improve productivity and reduces costs. It also imposes many new security risks which are related to cloud storage. As cloud is mainly used for the storage of the data, data integrity is the main issue of the client side because after uploading data to the server, client will lost the control of the data. There are so many techniques available in the literature, out of which we have analyze Provable Data Possession (PDP) and Proof of retrievability (POR), This paper facilitate the client in getting a proof of integrity of the data which He/She wishes to store in the cloud storage servers with bare minimum costs and efforts. The scheme used in this paper reduce the computational and storage overhead ofthe client as well as to minimize the computational overhead of the cloud storage server. This also minimized the size of the proof of data Integrity so as to reduce the network bandwidth consumption. Seeing thepopularity of outsourcing archival storage to the cloud, it is desirable to enable clients to verify the integrity of their data in the cloud. We study design of data integrity protection (DIP) scheme for functional minimum storage regenerating (FMSR) codes under a multi-server setting. This DIP scheme preserves the fault tolerance and repair traffic saving properties of FMSR. And also it allows clients to remotely verify the integrity of random subsets of long term.

We will evaluate the running time under various parameter choices. We will compare the three different techniques for checking integrity and fault tolerance of system and also evaluate the overhead of DIP scheme. Our scheme will preserves the fault tolerance and also less repair traffic.

In addition, we may exploit certain inherent properties of such combination to further reduce the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

computational overhead, and we pose this issue as our extended work. On the other hand, our modular approach allows us to flexibly enable DIP on demand in real deployment.

References

1. Bansidhar Joshi, A. Santhana Vijayan, Bineet KumarJoshi,"Securing Cloud computing Environment against DDoS
2. Ramgovind S, Eloff MM and Smith E, "The Management of Security in Cloud Computing", IEEE, 2010.
3. Wentao Liu, "Research on Cloud Computing Security Problem and Strategy", IEEE, 2012, pp.1216-1219.
4. Nitin Singh Chauhan and Ashutosh Saxena. "Energy Analysis of Security for Cloud Application.
5. Rohit Bhadauria, Rituparna Chaki, Nabendu Chaki and Sugata Sanyal, "A Survey on Security Issues in Cloud Computing", IEEE 2010.
6. Xiaojun Yu and Qiaoyan Wen "A view about cloud data security from data life cycle", IEEE 2010.
7. Saxena, Sravan Kumar and Ashutosh,"Data Integrity Proofs in Cloud Storages", IEEE 2011.
8. A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In Proc. ACM CCS, pages 584–597, 2007.
9. G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. Remote Data Checking Using Provable Data Possession. ACM Trans. on Information and System Security, May 2011.
10. H.Abu-Libdeh, L. rincehouse, and H. Weatherspoon. RACS: A Case for Cloud Storage Diversity. In Proc. of ACM SoCC, 2010.
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. Communications of the ACM, 53(4):50– 58, 2010.
12. K. Bowers, A. Juels, and A. Oprea. HAIL: A High-Availability and Integrity Layer for Cloud Storage. In Proc. of ACM CCS, 2009.
13. R. Li, J. Lin, and P. Lee. CORE: Augmenting Regenerating-Coding-Based Recovery for Single and Concurrent Failures in Distributed Storage Systems. arXiv, preprint arXiv:1302.3344, 2013.
14. I. Reed and G. Solomon. Polynomial Codes over Certain Finite Fields. Journal of the Society for Industrial and Applied Mathematics, 8(2):300–304, 1960.
15. J. Breckling, Ed., the Analysis of Directional Time Series: Applications to Wind Speed and Direction, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61. Y. Hu, P. P. C. Lee, and K. W. Shum. Analysis and Construction of Functional Regenerating Codes with Uncoded Repair for Distributed Storage Systems. In Proc. of IEEE INFOCOM, Apr 2013.
16. N Cao, S Yu, Z Yang, W Lou, YT Hou. LT codes-based secure and reliable cloud storage service- INFOCOM, 2012 Proceeding IEEE, 2012.
17. Y. Hu, H. Chen, P. Lee,and Y. Tang. NCCloud: Applying Network Coding for the Storage Repair in a Cloud-of-Clouds. In Proc. of USENIX FAST, 2012.
18. Y. Deswarte and J. Quisquater, "How to Trust Files Stored on Untrusted Servers,"2002.
19. F. Sebe, A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferrer, and J.-J. Quisquater. "Time- bounded remote file integrity checking." Technical Report 04429, LAAS, July 2004.
20. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," Proceedings of the 16thACM conference on Computer and communications security - CCS '09, p. 213, 2009.
21. Q. Wang, S. Member, C. Wang, and K. Ren, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," pp. 1–13.-2009.
22. S. Ni-Na and Z. Hai-Yan, "On Providing Integrity for Dynamic Data Based on the Third-party Verifier in Cloud Computing," 2011 First International Conference on Instrumentation Measurement,Computer Communication and Control, pp. 521–524, Oct. 2011.