

REGENERATING FAILED CLOUD WITH MINIMUM STORAGE

¹S. Sangeetha, ²B.Rasina Begum

¹Research Scholar, Department of Computer Science and Engineering, Mohamed Sathak Engineering College,
Ramanathapuram, India

²Assistant professor, Department of Computer Science and Engineering, Mohamed Sathak Engineering College,
Ramanathapuram, India

¹sangee510it@gmail.com, ²rasinayousuf@gmail.com

Abstract: When a cloud fails permanently, it is important to activate storage repair to maintain the level of data redundancy and fault tolerance. A repair operation reads data from existing surviving clouds and reconstructs the lost data in a new cloud. Functional minimum-storage regenerating (FMSR) code is used to regenerate the failed cloud with the help of surviving clouds. It maintains double-fault tolerance and use less repair traffic when recovering a single-cloud failure. FMSR maintains multiple copies of storage so the performance of cloud will be reduced. This code does not recover all the failed data and not provide efficient result. To overcome this problem, Replace recovery algorithm is proposed for the recovering process. With the help of this algorithm, cloud backup can easily retrieved when any one cloud moves to the failure due to some reasons such as collision, intrusion and fraudulency.

Index Terms: FMSR code, Surviving clouds, double-fault tolerance, Replace recovery algorithm, Cloud backup

1. INTRODUCTION

Cloud storage does have the potential for security and compliance concerns. It provides an on-demand remote backup solution. Cloud storage means the storage of data online in the cloud wherein a company's data is stored in and accessible from multiple distributed and connected resources that comprise a cloud. On the surface, cloud storage has several advantages over traditional data storage. For example, if storing the data on a cloud storage, need to able to get the data from any location that has internet access Single cloud storage vendor raises things such as having a single point of failure and vendor lock-ins. As suggested in, a plausible solution is to stripe data across different cloud vendors. Conventional erasure codes performs well while striping data and when some clouds experience short term failures or foreseeable permanent failures. Regenerating codes for distributed storage is built on the concept of network coding [1], [4]. It is demonstrated that regenerating codes reduce the data repair traffic over traditional erasure codes subject to the same fault-tolerance level. They aim to intelligently mix data blocks that are stored in existing storage nodes, and then regenerate data at a new storage node. This work focuses on unforeseen cloud failures.

Two types of failures are considered such as transient failure and permanent failure. At the time of these failures, it is important to activate storage repair to maintain the level of data redundancy. A transient failure is expected to be short-term, such that the

“failed” cloud will return to Normal after some time and no outsourced data are lost, where the durations of such failures range from several minutes to several days.

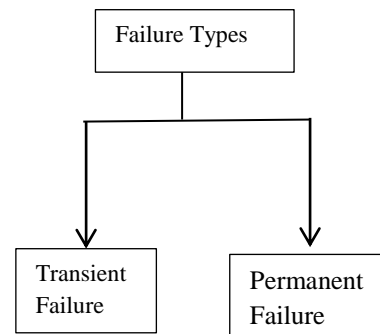


Figure 1: Types of failures in cloud

A permanent failure is long-term [13], in the sense that the outsourced data on a failed cloud will become permanently unavailable. Clearly, a permanent failure is more disastrous than a transient one. Cloud computing environment unexpected failures are occurred rarely. In that situation, repair operation had done for retrieving the data. FMSR regenerating code is used for retrieving the information from the failure cloud. Proxy based storage system for produce fault tolerance over multiple cloud service providers.



Figure 2: Traditional repair operation in cloud

The figure 2 shows that the proxy regenerates data for the new cloud. A repair operation reads data from existing surviving clouds and reconstructs the lost data in a new cloud. It is adorable to reduce the repair traffic and monetary cost due to data migration [10].

As in traditional regenerating code (FMSR), maintaining the proxy system and their data is complex one. Client and cloud system communicate through the proxy system, so the overall processing time is long, so replace recovery concept is introduced to reducing the overall processing time. It has an index file of each storing data, that index can be used to easily finding the location. The design key feature of replace recovery concept is to reduce the overall processing time and also it does not occupying the large amount of space so the performance of cloud will not be degraded. When compared to FMSR, Index file is not occupying the large amount of space. Remote Recovery mechanism is portable for all disks.

2 RELATED WORK

The repair traffic involved in different coding schemes that via following example of repair operation. Suppose store a file of size M on four clouds, each viewed as a logical storage node. Proxy-based design is proposed that interconnects multiple cloud repositories. The proxy serves as an interface between client applications and the clouds. If a cloud observes the permanent failure, the proxy activates the repair operation, that is, the proxy reads the essential data pieces from other remaining clouds, reconstructs new data pieces, and writes these new pieces to a new cloud.

FMSR codes [1] have higher computational overhead in decoding and recent studies improve the degraded read performance for erasure-coded data.

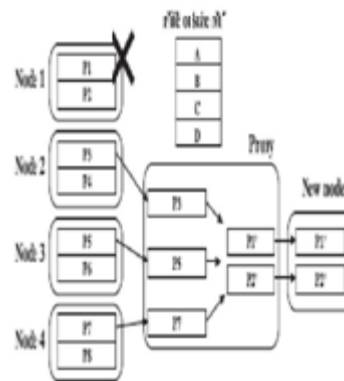


Figure 3: Example of repair operation in FMSR code

Now consider the double-fault-tolerant implementation of FMSR codes as shown in the above figure. Divide the file into four native blocks, and construct eight distinct code blocks P_1, \dots, P_8 formed by different linear combinations of the native blocks. Each code chunk has the same size $M/4$ as a native block. Any two nodes can be used to recover the original four native blocks. Suppose Node 1 is done, the proxy collects one code block from each surviving node, so it downloads three code blocks of size $M/4$ each. Then, the proxy regenerates two code blocks P_1' and P_2' formed by different linear combinations of the three code blocks. Note that P_1' and P_2' are still linear combinations of the native blocks. The proxy then writes P_1' and P_2' to the new node. A key property of FMSR codes is that nodes do not perform encoding during repair.

FMSR codes are nonsystematic, as they keep only code chunks but not native chunks. To access a single chunk of a file, need to download and decode the entire file for that particular chunk. This is opposed to systematic codes (as in traditional RAID storage), in which native blocks are kept. Nevertheless, FMSR codes are acceptable for long-term archival applications, where the read frequency is commonly low. Also, to restore backups, it is natural to retrieve the entire file rather than a particular chunk.

FMSR code is used to retrieve the failure cloud information and it maintains multiple copies of the storage. If anyone cloud failure due to some reasons, easily retrieve the backup from the remaining cloud. Figure 3 show that the four cloud storage is maintained and if any one cloud fails, the data need to retrieve from

remains three. The primary objective is to minimize the cost of storage repair (due to the migration of data over the clouds) for a permanent single-cloud failure. Here, repair traffic is defined as the amount of outbound data being downloaded from the other surviving clouds during the single-cloud failure recovery. Then seek to minimize the repair traffic for cost-effective repair. Here, inbound traffic is not considered as it is free of charge for many cloud providers.

For implementing FMSR codes in multiple-cloud storage, specify three operations for FMSR codes on a particular file object. They are following:

- File upload
- File download
- Repair

Each cloud repository is viewed as a logical storage node [5], [7]. The implementation assumes a thin cloud interface, such that the storage nodes (i.e., cloud repositories) only need to support basic read/write operations. Thus, expect that FMSR code implementation is compatible with today's cloud storage services. One property of FMSR codes is that do not require lost chunks to be exactly reconstructed, but instead in each repair, regenerate code chunks that are not necessarily identical to those originally stored in the failed node. FMSR maintains multiple copies so the performance of cloud will be reduced.

3. PROPOSED SYSTEM

To overcome the problems present in the existing system new recovery mechanism such as the remote recovery mechanism is proposed. In this algorithm, recover the disk such as cloud based upon the parity values. Because of all the data's are present like binary format in the cloud. The primary objective is to minimize the amount of data read from the surviving disks for recovery and hence the overall time of the recovery operation, while the recovery solution can be quickly determined.

In replace recovery algorithm, a hill-climbing (greedy) approach is used to optimize the recovery solution. It starts with a feasible recovery solution and incrementally replaces the current solution with another one that reads less data. After, validate that it provides near-optimal recovery for different variants of FMSR codes. Also, it is shown to achieve polynomial complexity. Note that the replace recovery can be

extended for the setting where disks are heterogeneous with different performance costs. This implies that replace recovery can be applied based on the current performance costs of surviving disks, while existing enumeration recovery is infeasible in doing so due to its exponential complexity.

In cloud computing environment there is a possibility of losing of data due to user error or system fault. In such unexpected situation, need to retrieve the lost data from the source.

The main goal of this project is,

- Easily regenerating or retrieving the loosed data when unexpected failures are occurred.
- Quick to read the location of loosed data.
- Reduce the overall time of searching and retrieving than FMSR code.

3.1 File process

The first process of this project is to select the files for loading process. Here, the files are to be uploaded in to different server. The indexing information of each and every file can be maintained in main server but the files are stored in cloud storage. In this process, the cloud server can contains all information about the servers.

File upload can be used in a number of different ways, depending upon the requirements of the application. Each file item has a number of properties that might be of interest for the application. If one file has to be uploaded means the file storage information are stored in main server. The details are IP Address, Port Number then status of the particular server. In case the file has to be stored in one server the main server first note down status of the server is the file name, then display status of the file. Status means if the file has been corrupted or faulty identified means the status has been displayed in the main server. Every movement of the file can be displayed in the main server status.

Two kinds of files are following: One is device file and another one is log file. The device file is an interface for a device driver that appears in a file system as if it were an ordinary file. This file can have the information of IP address and port number of every server connected to the cloud server. Log file have the information such as connectivity status of every server, file name, file size, Timing.

3.2 System process

Process is the actual execution of those instructions and

it may be made up of multiple threads of execution that execute instructions concurrently. Concurrency is a property of systems in which several computations are executing simultaneously, and likely interacting with each other. Several processes may be associated with the same program. For example, opening up several details of the same program often means more than one process is being executed.

The log file and device are internal process of the system. Log file is a file that records either the events which happen while an operating system or other software runs, or the personal messages between peculiar users of communication software. The act of keeping a log file is called logging. In the transparent case, log messages are written to a single log file. Device or special file is an interface for a device driver that appears in a file system as if it were an ordinary file. They grant software to interact with a device driver using standard input/output system calls, which clarify many tasks and unifies user-space I/O mechanisms but they can also be used to access specific resources on those devices.

In System process, the files can be downloaded by some user from different servers. While downloading the files first need to give the request to the server. According to the application, the file can be downloaded from different server but the storage process is not known to the users. The directory indexing information of each and every files are stored in main server because to reduce the amount of storage and easily retrieving the destroyed file. The files are retrieved from peculiar server. At the time of downloading, if any one of the servers may be shutdown.

3.3 Greedy approach

Greedy approach mainly used for storage process. It starts with a feasible recovery solution, and incrementally replaces the current solution with another one that reads less data. This approach mainly used at the time of destroying the files from the server. Before presenting replace recovery algorithm, primitive function is needed to determine if one server is valid to resolve the data symbols after being replaced with other parity symbols in another server. For each parity symbol, bit encoding vector is defined that specifies how the strip of lost data symbols is encoded to the parity symbol.

A greedy algorithm is a mathematical process that

looks for simple, easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit. Greedy algorithms work by recursively constructing a set of objects from the smallest possible constituent parts. Recursion is an approach to problem solving in which the solution to a particular problem depends on solutions to smaller instances of the same problem. The advantage to using a greedy algorithm is that solutions to smaller instances of the problem can be straightforward and easy to understand. Greedy algorithms are often used in ad hoc mobile networking to efficiently route packets with the fewest number of hops and the shortest delay possible.

3.4 Recovery process

The simplified recovery model states that there exists a recovery solution that contains exactly parity symbols for regenerating lost data symbols for each server failure. Computationally efficient replace recovery algorithm that seeks to minimize the number of read symbols for single disk failure recovers.

In recovery process, replace recovery algorithm is used to recover the failed server information in the cloud. The directory indexing information of each and every file is stored in every server. Using this algorithm, recover the files which can be stored in the failure server. This replace recovery algorithm can have some objectives like search efficiency, effective recovery performance and adaptable to heterogeneous network system because this algorithm finds a recovery solution with polynomial complexity. In this process, the main server can contains all servers directory indexing information. Cloud backup can easily retrieved when any one cloud moves to the failure due to some reasons such as collision, intrusion and fraudulency.

According to the process of replace recovery algorithm the files can get from servers. This algorithm provides optimal recovery performance in the cloud and can replace the files from destroyed servers. Simplified recovery model states that there exists a recovery solution that contains exactly parity symbols for regenerating lost data symbols for each server failure. Computationally efficient replace recovery algorithm that seeks to minimize the number of read symbols for single disk failure recovers.

3.5 Algorithm: Replace recovery algorithm

Replace recovery algorithm can be used to recover the failure server's information in the cloud. Using this algorithm, recover the files which can be stored in the failure server. This replace recovery algorithm can have some objectives like search efficiency, effective recovery performance and adaptable to heterogeneous network system because this algorithm find a recovery solution with polynomial complexity. In this process, the main server can contains all servers directory indexing information. According to the process of replace recovery algorithm, can get the files from servers. This algorithm provides optimal recovery performance in the cloud.

4. SYSTEM FLOW DIAGRAM

Figure 2 shows the system flow diagram of recovering the removed files. Initially, users are uploading the files and transmit in a packet manner then the files are sending to different servers.

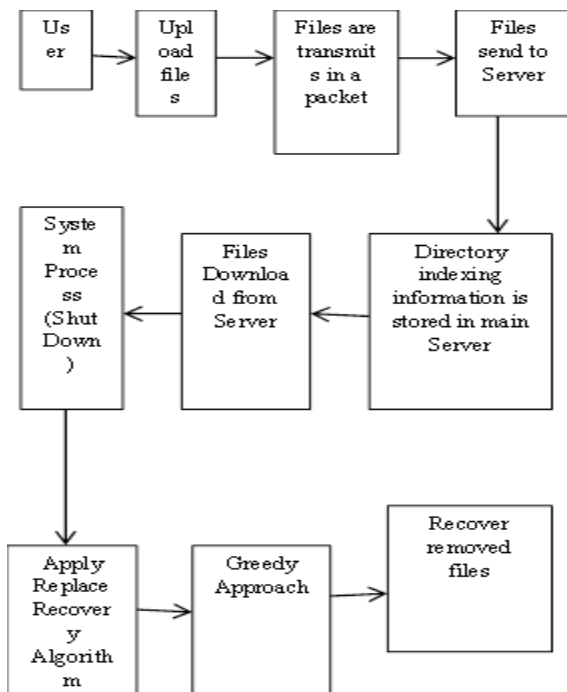


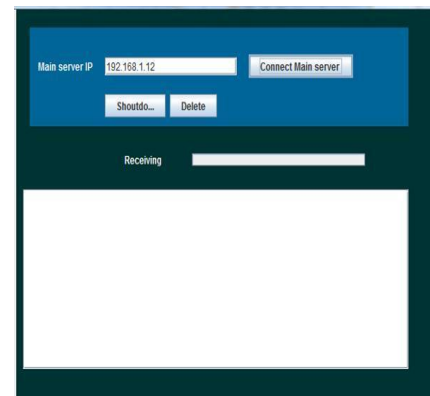
Figure 4: System flow diagram

The directory indexing information of each and every file can be maintained in main server. After that the files are downloading from Recover

the failed cloud information by using replace recovery algorithm with less memory space and less recovery time. Here the performance of cloud will be increased then easily find the location where the data had deleted by indexing information of failed cloud. Different servers, at the time of downloading the server may down. Replace recovery algorithm is used to recover the files from destroyed server which uses greedy approach to recover the optimized solution. Then finally the recovered files are recovered.

5. EXPERIMENTAL RESULTS

Recover the failed cloud information by using replace recovery algorithm with less memory space and less recovery time. Here the performance of cloud will be increased then easily find the location where the data had deleted by indexing information of failed cloud.



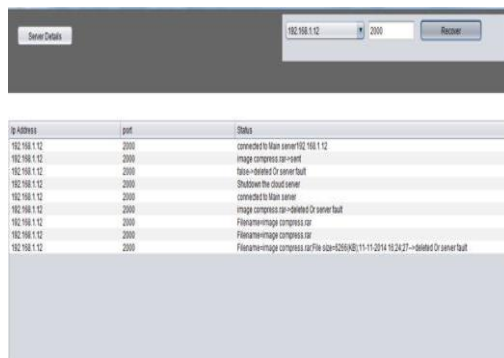
(a)

Figure (a) shows that the cloud server is connected to the main server with IP address. After connection only, the main server can receive the file from cloud server that files are receiving in a packet manner.



(b)

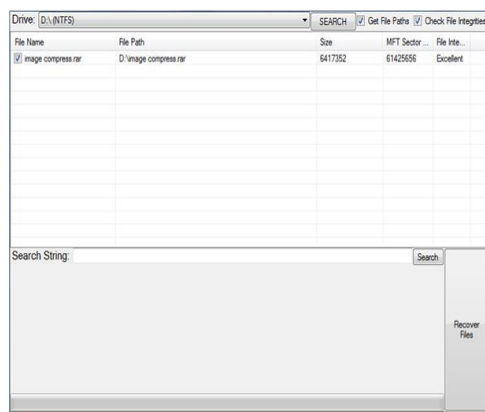
Fig (b) shows the uploading process of file, here the files are loaded in to different server. One file has to be uploaded means the file storage information are stored in main server. The details are IP Address, Port Number then status of the particular server.



IP Address	port	Status
192.168.1.12	2000	connected to main server 192.168.1.12
192.168.1.12	2000	image compress server
192.168.1.12	2000	data deleted to server fault
192.168.1.12	2000	Shutdown the cloud server
192.168.1.12	2000	connected to main server
192.168.1.12	2000	image compress server deleted to server fault
192.168.1.12	2000	File name image compress.rar
192.168.1.12	2000	File name image compress.rar File size=6265KB; 11-11-2014 10:20:27 -> deleted to server fault

(c)

Fig (c) shows the uploaded files may destroy due to some reasons such as fraudulency or disaster. While downloading, the file may be failed and the intimation will send to main server



File Name	File Path	Size	MFT Sector	File Info
image compress.rar	D:\image compress.rar	6417352	61425556	Excellent

Search String:

Fig. 5.Results of the proposed system: (a) Connect to the main server with IP address; (b)Uploading the files into cloud; (c) File will be deleted while downloading; (d) Recover the deleted files.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The fault-tolerant storage is provided by a proxy based storage system in this proposed paper. Here, the functional minimum-storage regenerating (FMSR) code is used for double tolerant over multiple cloud storage. In the proposed concept, some recovery algorithm is

used for recovering the cloud data when the cloud is damaged. This algorithm recovers the data in a cloud and also replaces the same content. This algorithm is very useful for recovering important data from the natural disasters, loss and damage. The experimental results shows that the proposed algorithm is efficiently recover the backup during the corruption and the cost is also reduced.

6.2 Future work

In proposed system remote recovery method is used to obtain loosed data in the failure disk. The main objective is to reduce the size of file while reading the recovery operation, and recovery process is done in a quickly manner. In future work, new log based recovery method is introduced for retrieving the data from the failure disk. In this algorithm, read only the index of the failure file system instead of reading the whole file. If the data will be blast, log based recovery algorithm can easily redo the previous operation. It can easily retrieving the data and write these data to the new disk with in a minimum time. This algorithm can easily fetch the data so the overall time consumption is too low and take minimum time for recovery and reduces the cost of processing.

7. ACKNOWLEDGEMENT

The author would like to thank the respective Head of the Institution, Head of the Department and Faculty members for giving valuable advices and providing technical support.

REFERENCES

- [1] Henry C.H. Chen, Yuchong Hu, Patrick P.C. Lee, and Yang Tang, "NCCloud: A Network-Coding-Based Storage System in a Cloud-of- Clouds," IEEE TRANSACTIONS ON COMPUTERS, VOL. 63, NO. 1, JANUARY 2014
- [2] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A.G. Dimakis, R.Vadali, S. Chen, and D. Borthakur, "XORing Elephants: Novel Erasure Codes for Big Data," Proc. VLDB Endowment, vol. 6, pp. 325-336, 2013.
- [3] B. Schroeder and G.A. Gibson, "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?" Proc. Fifth USENIX Conf. File and Storage Technologies (FAST '07), Feb. 2007.
- [4] K. Shum, "Cooperative Regenerating Codes for Distributed Storage Systems," Proc. IEEE Int'l Conf. Communications (ICC '11), June 2011.

- [5] K. Shum and Y. Hu, "Exact Minimum-Repair-Bandwidth Cooperative Regenerating Codes for Distributed Storage Systems," Proc. IEEE Int'l Symp. Information Theory (ISIT '11), July 2011.
- [6] 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.
- [7] R. Wauters,, "Online Backup Company Carbonite Loses Customers' Data, Blames and Sues Suppliers," <http://techcrunch.com/2009/03/23/online-backup-company-carbonite-loses-customersdata-blames-and-sues-suppliers/>, Mar. 2009.
- [8] R. Kossman, "Cloud Case Studies: Data Storage Pros Offer First-Hand Experiences," <http://searchcloudstorage.techtarget.com/feature/Cloud-case-studies-Data-storage-pros-offer-first-handexperiences/>, 2013.
- [9] M. Vrabie, S. Savage, and G. Voelker, "Cumulus: Filesystem Backup to the Cloud," Proc. USENIX Conf. File and Storage Technologies (FAST '09), 2009.
- [10] M. Vukolić, "The Byzantine Empire in the Intercloud," ACM SIGACT News, vol. 41, pp. 105-111, Sept. 2010.
- [11] Z. Wang, A. Dimakis, and J. Bruck, "Rebuilding for Array Codes in Distributed Storage Systems," Proc. IEEE GlobeCom Workshops, 2010.
- [12] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, "A Hybrid Approach to Failed Disk Recovery Using RAID-6 Codes: Algorithms and Performance Evaluation," ACM Trans. Storage, vol. 7, no. 3, article 11, 2011.