IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 1, Feb-Mar, 2015
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Case-Study On Functional Minimum Storage Regenerating Codes For Fault Tolerance Cloud Storage

Rishabh Anand

Manager, HCL Technologies Ltd –IOMC, India

Abstract: 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 the Internet." Cloud storage offers an on-demand data out sourcing service model, and is gaining popularity due to its elasticity and low maintenance cost.
Recent advances have given rise to popularity and successes of cloud computing. Cloud storage enables users to remotely store their data and enjoy the on-demand remote backup service. To provide fault tolerance for cloud storage recent studies proposed to stripe data across multiple cloud vendors. If cloud suffers from a permanent failure and losses all its data, we need to repair the lost data with the help of the other surviving clouds to preserve data redundancy. NCCloud is a proof of concept prototype of a network coding based file system that aims at providing fault tolerance and reducing the storage capacity when storing files using multiple cloud storage. NCCloud is a proxy based file system that interconnects multiple storage nodes, which achieves cost effective repair for permanent single-cloud failure. It is built on top of a network coding-based storage scheme called functional minimum storage regenerating (FMSR) codes. Compared to traditional optimal erasure codes 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 node. NCCloud realizes regenerating codes in practical cloud storage system that does not require any encoding/decoding intelligence on the cloud storage nodes.

Keywords: Regenerating codes, Network coding, Fault tolerant system, Recapture, Encoding, Implementation, Reparation.

I. INTRODUCTION

With the rapid growth of data production in companies, the requirement of storage space grows very largely as well. This growth leads to the emergence of cloud storage. Cloud storage is a concept which is an extension and development from cloud computing. This system collects application software in order to work together and provide systems of data storage and business access features through grids or distributed file systems. Cloud storage is produced by distributed storage technology and virtualization technology, and it is the latest development of distributed storage technology. Cloud storage provides effective solutions for network mass data storage. Also this system provides on-demand pay services which reduces not only the threshold for the user but also the payment. A single-cloud storage provider encounters the problem such as a single point of failure [1] and vendor lock-ins [2]. As suggested in [1], [2], [3], a probable solution is to stripe data across multiple cloud vendors. However, if cloud suffers from a permanent failure, than the outsourced data on a failed cloud will become permanently unavailable. In order to safeguard our precious data against such failures, it is necessary to activate repair to maintain data redundancy and fault tolerance. A repair operation retrieves data from existing surviving clouds over the network and reconstructs the lost data in a new cloud. In our definition of repair we mean to retrieve data only from the other surviving clouds and reconstruct the data in new storage site.

II. PROBLEM STATEMENTS

Cloud storage provides an on demand remote backup solution. However, using single cloud storage provider raises unexpected permanent cloud. When a cloud fails permanently, will make the hosted data in the failed cloud no longer accessible, so it must repair and reconstruct the lost data in a different cloud or a storage site to maintain the required degree of fault tolerance and data redundancy. Today's cloud storage providers charge users for outbound, so moving an enormous amount of data across clouds can introduce significant monetary costs. It is important to reduce the repair traffic (i.e., the amount of data being transferred over the network during repair), and hence, the monetary cost due to data migration. A good erasure code technology can not only improve the availability and reliability of the system but can also improve the efficiency of data access.

III. PROBABLE SOLUTION

To minimize repair traffic, regenerating codes [4] have been proposed for storing data redundantly in a distributed storage system. Regenerating codes are built on the concept of network coding [5], in the sense that nodes perform encoding operations and send encoded data. During repair, each surviving node encodes its stored data chunks and sends the encoded chunks to a new node, which then regenerates the lost data. Regenerating codes require less repair traffic than traditional erasure codes [6] with the same fault-tolerance level.

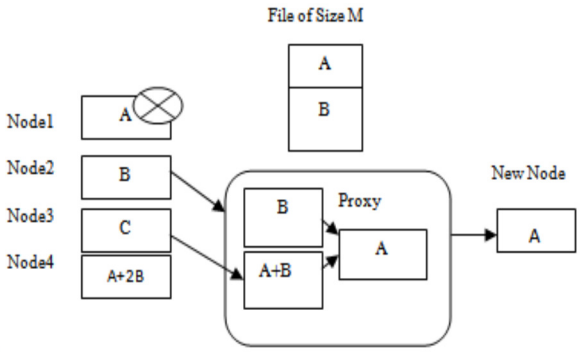

Figure 1. RAID-6

IV. DIFFERENT CODING SCHEME

Implementation based on the RAID 6 Reed-Solomon [7], as shown in Figure 1. We divide the file into two native chunks (i.e., A and B) of size M=2 each. We add two code chunks formed by the linear combinations of the native chunks. Suppose now that Node 1 is down. Then, the proxy must download the same number of chunks as the original file from two other nodes (e.g., B and A + B from Nodes 2 and 3, respectively). It then reconstructs and stores the lost chunk A on the new node. The total storage size is 2M, while the repair traffic is M. Let consider double fault tolerant implementation of EMSR code [8], as shown in Figure 2.We divide file into four chunks and allocate the native and code chunks. Suppose Node 1 is down. To repair it, each surviving node sends the XOR summation of the data chunks to the proxy, which then reconstruct the lost chunks.EMSR codes the storage size are 2M (same as RAID 6 codes), while the repair traffic is 0.75M, which is 25% of saving.EMSR codes leverage the notion of network coding as the nodes generate encoded chunks during repair.

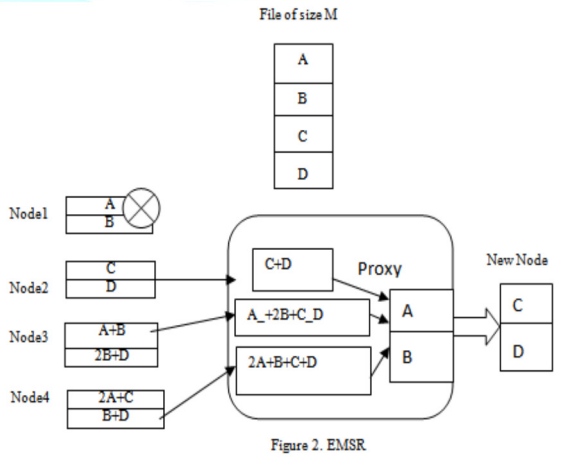

Figure 2. EMSR

The double fault tolerant implementation of FMSR codes as shown in Figure 3.Here we divide the file into four native chunks, and construct eight distinct code chunks formed by different linear combinations of the native chunks. Each native chunk has the same size M/4 as native chunks. Any two nodes can be used to recover the original four native chunks. If Node 1 is down, the proxy collects one code chunks from each surviving node ,so it downloads three code chunks is size M=4 each. Then the proxy generates two combinations of the three code chunks and writes in the new node.

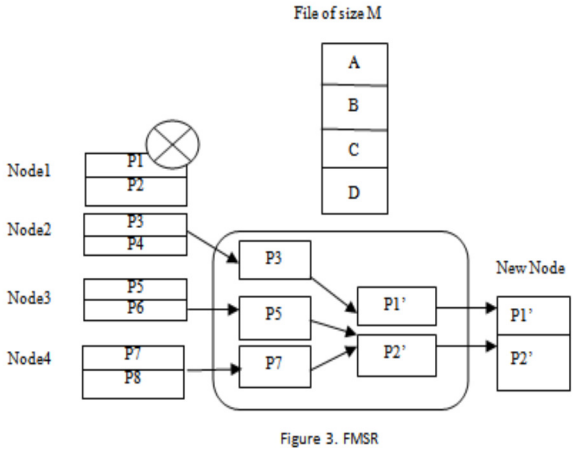

Figure 3. FMSR

V. TECHNIQUES

FMSR maintains double fault tolerance and eliminate the need to perform encoding operations within storage nodes during repair, while preserving the benefits of network coding in reducing repair traffic. Here we will consider fault tolerant storage based on a type of maximum distance separable (MDS) codes. Given a file object of size M, we divide it into equal size native chunks, which are linearly combined to form code chunks. When an (n,k) MDS code is used ,the native/code chunks are then distributed over nodes, each storing chunks of total size M/k, such that the original file object may be reconstructed from the chunks contained in any n-k nodes. We call this fault tolerance feature the MDS property. One of the extra feature of FMSR codes is that reconstructing the chunks stored in the failed node can be achieved by downloading less data from the whole file.

One key challenge for deploying regenerating codes in practice is that most existing regenerating codes require storage nodes to be equipped with computation capabilities for performing encoding operations during repair. On the other hand, to make regenerating codes portable to any cloud storage service, it is desirable to assume only a thin-cloud interface, where storage nodes only need to support the standard read/write functionalities. This motivates us to explore, from an applied perspective, how to practically we can deploy regenerating codes in multiple-cloud storage, if only the thin-cloud interface is assumed.

VI. MOTIVATION FOR FMSR CODE

NCCloud is proxy-based design [9] that interconnects multiple cloud repositories, as shown in Figure 4. The proxy serves as an interface between client applications and the clouds. If a cloud experiences a permanent failure, the proxy activates the repair operation, as shown in Figure 5. That is, the proxy reads the essential data pieces from other surviving clouds, reconstructs [10][11] new data pieces, and writes these new pieces to a new cloud. One property of FMSR codes is that we do not require lost chunks to be exactly reconstructed but instead in each repair, we regenerate code chunks that are not necessarily identical to those originally stored in the failed node, as long as the MDS property [12] holds. It support a two-phase checking scheme, which ensures that the code chunks [13] on all nodes always satisfy the MDS property, and hence data availability, even after iterative repairs.

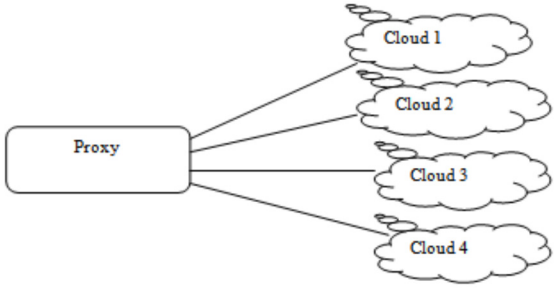

Figure 4. Normal operation

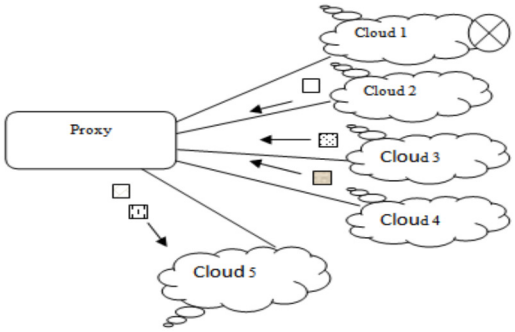

Figure 5. REPAIR OPERATION

VII. RELATED WORK

For implementing FMSR code, for particular file object specify 3 operations (i) File upload operations with no failure, (ii) File download operations with node failure, (iii) Repair operations during node failure

a. File upload: In the upload operations, NCCloud generates code chunks for the file based on FMSR codes. The code chunks will be temporarily stored in the local file system instead of being uploaded to the server

b. File download: In the download operations, the idea is to treat FMSR code as standard Reed Solomon codes, and the creating an inverse matrix technique is to decode the original data.

c. Repair: The repair operations of failed node include three steps. First file transmission of the existing blocks from survival nodes to NCCloud, second one generation for lost blocks of the failed node in NCCloud and third one transmission of the generated blocks from NCCloud to the new node [3].If there is more than one failed node, then apply the repair operation for each failed node one by one.

VIII. ACKNOWLEDGMENT

No one walks alone and when one is walking on the journey of life just where I start to thank those who joined me, walked besides me, and helped me along the way.Over the years, those that I have met and worked with have continuously motivated me to write this paper. So at last, here it is. So, perhaps this paper and its pages will be seen as "thanks" to the tens of thousands of you who have helped bringing out this paper in the form what is today.

Finally, thanks to my mother and to rest of my family for their patience and support during the long hours of writing this book and above all there is the one almighty whose humble children we are. It is his blessings we cherish and pray for. It is the blessing I wish for you.

My special acknowledgment to all the authors whose books, journals and paper are referred while writing this paper.

IX. CONCLUSION

Throughout this paper, we give an overview of NCCloud proxy based file system that connect multiple storage nodes that practically address the reliability of today's cloud backup storage. NCCloud not only provides fault tolerance in storage but also minimize the storage capacity NCCloud implements a practical version of the FMSR codes, which regenerates a new parity chunks during repair subject the required degree of data redundancy.FMSR implementation eliminates the encoding requirement of storage nodes during repair and ensuring that new set of code chunks after each round of repair preserves the required fault tolerance

REFERENCES

[1]. 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.

[2]. 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.

[3]. 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 '09), 2009.

[4]. K. Shum, "Cooperative Regenerating Codes for Distributed Storage Systems," Proc. IEEE Int'l Conf. Communications (ICC '11) June 2011

[5]. A.G. Dimakis, P.B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," IEEE Trans. Information Theory, vol. 56, no. 9, pp. 4539-4551,Sept. 2010.

[6]. J.S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-Like Systems," Software Practice & Experience vol. 27, no. 9, pp. 995-1012, Sept. 1997.

[7]. I. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," J. the Soc. Industrial and Applied Math., vol. 8, no. 2, pp. 300-304, 1960

[8]. 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.

[9]. Y. Hu, C.-M. Yu, Y.-K. Li, P.P.C. Lee, and J.C.S. Lui, "NCFS: On the Practicality and Extensibility of a Network-Coding-Based Distributed File System," Proc. Int'l Symp. Network Coding (NetCod '11), 2011

[10]. Y. Hu, P.P.C. Lee, and K.W. Shum, "Analysis and Construction of Functional Regenerating Codes withUncoded Repair for Distributed Storage Systems," Proc. IEEE INFOCOM, Apr. 2013.

[11]. K. Rashmi, N. Shah, and P. Kumar, "Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction," IEEE Trans. Information Theory,vol. 57, no. 8, pp. 5227-5239, Aug. 2011.

[12]. B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-Based Distributed

Storage Systems, "Proc. ACM Workshop Cloud Computing Security Workshop (CCSW '10), 2010

[13]. Z. Wang, A. Dimakis, and J. Bruck, "Rebuilding for Array Codes in Distributed Storage Systems," Proc. IEEE GlobeCom Workshops, 2010.

[14]. 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.

[15]. S. Ni-Na and Z. Hai-Yan, "On ProvidingIntegrity for Dynamic Data Based on theThird-party Verifier in Cloud Computing,"2011 First International Conference onInstrumentation Measurement,Computer Communication and Control, pp. 521–524, Oct. 2011.

Er. Rishabh Anand received his Bachelor's degree B.E. (Hons) in Electronics and Communication Engineering from Maharishi Dayanand University, Rohtak in 2006 and M.Tech. from Veer Bahadur Singh Purvanchal University, Jaunpur in 2014. His areas of interests include Cloud Computing,Information Security and Cyber Laws, Complier Design, Business Intelligence, Data Warehousing and Data Mining, Software Project Management, Software Engineering, Information Storage Management, Digital Image Processing, Distributed Operating System, Distributed Databases, Wireless and Mobile Computing and Advanced Computer Networks. He is prolific author of 20 engineering books. He is a currently working in MNC as a Manager. He is ITILV3 (F) and Prince2 (P) Certified Professional.