INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

Cloud Based Data Recovery and Reconstruction System using Bi Methodology Erasure Code Implementation

P. Praveen Kumar #1, K. Madhan *1

Mailam Engineering College, Mailam #1, *1

pkumartin@gmail.com #1

Abstract - Reconstruction time has been minimized in erasure coded cloud storage. In previous work, as per the Traditional Reconstruction Techniques, Master node sends the request to the Worker node dedicated for the Reconstruction Process. This process encounters lots of Bottleneck Problems. In the proposed method, we are implementing Two Techniques namely, PUSH–Rep & PUSH–Sur. In PUSH–Rep Reconstruction occurs using Replacement Nodes. Rebuilt blocks are sequentially written to the disks of replacement nodes. PUSH–Sur allows each surviving node to rebuild a subset of failed data, so all the surviving nodes accomplish the reconstruction in parallel. In modified work, we are deploying this Application in Cloud. Data is encrypted, separated and stored in different Cloud. Replica is created for data backup. Top Hash Key is stored in Separate Cloud as well in the Local Backup. We implement PUSH–Rep using reconstruction from Cloud Backup and PUSH–Sur reconstruction from Local Backup.

Keywords – Erasure code, replacement, reconstruction, Parallel

1 Introduction

Traditional reconstruction techniques in storage clusters advocate the pull model, where a master node initiates reconstruction by sending requests to worker nodes dedicated to the reconstruction process. The passive pull model inevitably encounters a transmission bottleneck problem that lies in rebuilding nodes. In this paper, we propose two PUSH-based reconstruction schemes

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

PUSH-Rep and PUSH-Sur—to improve reconstruction performance in a distributed storage cluster. At the heart of this study is the proactive PUSH technique that evenly distributes network and I/O loads among surviving nodes to shorten reconstruction times. The following three factors motivate us to propose the PUSH-based reconstruction technique for erasure-coded clustered storage. The high cost-effectiveness of erasure-coded storage, the severe impact of recovery time on reliability, and the deficiency of PULL-based reconstruction of input and output. 1. Erasure-coded storage clusters have increasingly become a cost-effective and fault-tolerant solution for archive storage data centers cloud storage and the like. Especially, Reed-Solomon (RS) codes are widely used in storage clusters to provide high data reliability. For example, Windows Azure Storage (WAS) adopts a variant of RS codes to implement a four-fault-tolerant cluster system. A detailed review on the RS-coded distributed storage is provided in Motivation 2. Ideally, erasure-coded storage clusters should protect against data loss caused by node failures, because high reliability is an indispensable requirement for building large-scale storage systems. The mean-time-to data- loss or MTTDL of a r-fault-tolerant storage system is inversely proportional to the power of recovery time of a storage node. Therefore, it is extremely important to speed up the reconstruction process, which in turn can improve system reliability by shrinking vulnerability window size. The existing reconstruction schemes adopt a PULL-transmission mode, where a rebuilding node initiates the reconstruction by sending read requests to fetch/pull surviving blocks. Such a PULL mode not only raises the TCP In cast problem due to its synchronized many-to-one traffic pattern, but also yields poor reconstruction performance. When it comes to a reconstruction which relies on replacement nodes, the network traffic of replacement nodes contributes to an excessively long reconstruction time. The problem with the reconstruction among surviving nodes is that each surviving node bears extra seek time owing to the non-contiguous disk access. This problem makes the low write bandwidth become a major reconstruction performance bottleneck. In this paper, we

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

introduce a PUSH-type transmission to speed up node-reconstruction performance. Our PUSH enables surviving nodes to accomplish reconstruction tasks in a pipelining manner. Each surviving node combines its local block with an intermediate block from another surviving node to partially generate an intermediate block forwarded to a subsequent node. Thus, PUSH can speed up the reconstruction process by maximizing the utilization of both network and I/O bandwidth of all the surviving nodes.

2 Related Work

In the previous work, as per the Traditional Reconstruction Techniques, Master node sends the request to the Worker node dedicated for the Reconstruction Process. This process encounters lots of Bottleneck Problems. Here it provides some drawbacks are, Waiting time is increased, Congestion occurring, Unreliable, Less data transmission rate Less effective

2.1 Proposed Work

In the proposed work, we are implementing Two Techniques namely, PUSH–Rep & PUSH–Sur. In PUSH–Rep Reconstruction occurs using Replacement Nodes. Rebuilt blocks are sequentially written to the disks of replacement nodes. PUSH–Sur allows each surviving node to rebuild a subset of failed data, so all the surviving nodes accomplish the reconstruction in parallel. In the modified work, we are deploying this Application in Cloud. Data is encrypted, separated and stored in different Cloud. Replica is created for data backup. Top Hash Key is stored in Separate Cloud as well in the Local Backup. We implement PUSH–Rep using reconstruction from Cloud Backup and PUSH–Sur reconstruction from Local Backup.

3 PULL-Based Reconstruction Scheme

Let us consider two existing reconstruction techniques that rely on the pull mode, where a rebuilding node first issues read requests to surviving nodes and then reconstructs a failed block using the requested blocks. The PULL-based reconstruction can be envisioned as a master-worker computing model, in which a master triggers a reconstruction procedure by sending a set of read requests, and then waits for the requests to be completed by workers. There are two classical reconstruction approaches in real-

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

world erasure-coded storage clusters: i) a designated master (e.g., a replacement node) fetches k surviving blocks and reconstructs a failed block, and ii) each surviving node plays the role of a master (i.e., acting as a rebuilding node) and all surviving nodes perform as workers, where write I/O s of rebuilt blocks are spread out over all the surviving nodes. From the angle of message communication, this 'Master Worker' pattern belongs to the category of PULL-type transmission. Throughout this paper, we refer to the reconstruction scheme using replacement nodes as PULL Rep; we term the solution of distributing reconstruction load among surviving nodes as PULL-Sur. In the case of PULL-Rep, all reconstruction reads are sequential requests that minimize disk seeking times; rebuilt blocks are sequentially written to disks of replacement nodes. Fig. 2a shows that k surviving blocks should be delivered to a replacement node (e.g., RN), which becomes a network bottleneck that slows down the entire reconstruction process. Furthermore, such a many- to-one ('M: 1') communication pattern may cause the severe In cast problem

4 Literature Review

[4] Describes about, Digital archives are growing rapidly, necessitating stronger reliability measures than RAID to avoid data loss from device failure. Mirroring, a popular solution, is too expensive over time. We present a compromise solution that uses multi-level redundancy coding to reduce the probability of data loss from multiple simultaneous device failures. This approach handles small-scale failures of one or two devices efficiently while still allowing the system to survive rare-event, larger-scale failures of four or more devices. In our approach, each disk is split into a set of fixed size diskless which are used to construct reliability stripes. To protect against rare event failures, reliability stripes are grouped into larger ""user-groups," each of which has a corresponding ""user-parity;" ¨user-parity is only used to recover data when disk failures overwhelm the redundancy in a single reliability stripe. ¨User-parity can be stored on a variety of devices such as NV-RAM and always-on disks to offset write bottlenecks while still keeping the number of active devices low. Our calculations of failure probabilities

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

found that the addition of ¨user-groups allowed the system to absorb many more disk failures without data loss. Through discrete event simulation, we found that adding ¨user-groups only negatively impacts performance when these groups need to be used for a rebuild. Since rebuilds using ¨user-parity occur very rarely, they minimally impact system performance over time. Finally, we showed that robustness against rare events can be achieved for fewer than 5% of total system cost. [16] Describes about, In this paper we describe Cumulus, a system for efficiently implementing file system backups over the Internet. Cumulus is specifically designed under a thin cloud assumption—that the remote datacenter storing the backups does not provide any special backup services, but Only provides a least-common-denominator storage interface (i.e., get and put of complete files). Cumulus aggregates data from small files for remote storage, and uses LFS-inspired segment cleaning to maintain storage efficiency. Cumulus also efficiently represents incremental changes, including edits to large files. While Cumulus can use virtually any storage service, we show that its efficiency is comparable to integrated approaches. [22] Describes about, In spite of the central role of key derivation functions (KDF) in applied cryptography, there has been little formal work addressing the design and analysis of general multi-purpose KDFs. In practice, most KDFs (including those widely standardized) follow ad-hoc approaches that treat cryptographic hash functions as perfectly random functions. In this paper we close some gaps between theory and practice by contributing to the study and engineering of KDFs in several ways. We provide detailed rationale for the design of KDFs based on the extract- then-expand approach; we present the first general and rigorous dentition of KDFs and their security that we base on the notion of computational extractors; we specify a concrete fully practical KDF based on the HMAC construction; and we provide an analysis of this construction based on the extraction and pseudorandom properties of HMAC. The resultant KDF design can support a large variety of KDF applications under suitable assumptions on the underlying hash function; particular attention and effort is devoted. [23]

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

Describes about, The increasing popularity of cloud storage is leading organizations to consider moving data out of their own data centers and into the cloud. However, success for cloud storage providers can present a significant risk to customers; namely, it becomes very expensive to switch storage providers. In this paper, we make a case for applying RAID-like techniques used by disks and file systems, but at the cloud storage level. We argue that striping user data across multiple providers can allow customers to avoid vendor lock-in, reduce the cost of switching providers, and better tolerate provider outages or failures. We introduce RACS, a proxy that transparently spreads the storage load over many providers. We evaluate a prototype of our system and estimate the costs incurred and benefits reaped. Finally, we use trace-driven simulations to demonstrate how RACS can reduce the cost of switching storage vendors for a large organization such as the Internet Archive by seven-fold or more by varying erasure-coding parameters. [7] Describes about, Latent sector errors (LSEs) refer to the situation where particular sectors on a drive become inaccessible. LSEs are a critical factor in data reliability, since a single LSE can lead to data loss when encountered during RAID reconstruction after a disk failure. LSEs happen at a significant rate in the field [1], and are expected to grow more frequent with new drive technologies and increasing drive capacities. While two approaches, data scrubbing and intra-disk redundancy, have been proposed to reduce data loss due to LSEs, none of these approaches has been evaluated on real field data. This paper makes two contributions. We provide an extended statistical analysis of latent sector errors in the field, specifically from the view point of how to protect against LSEs. In addition to providing interesting insights into LSEs, we hope the results (including parameters for models we fit to the data) will help researchers and practitioners without access to data in driving their simulations or analysis of LSEs. Our second contribution is an evaluation of five different scrubbing policies and five different intra-disk redundancy schemes and their potential in protecting against LSEs. Our study includes schemes and policies that have been

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

suggested before, but have never been evaluated on field data, as well as new policies that we propose based on our analysis of LSEs in the field.

6 Methodologies

6.1 Owner in cloud

User is the person is going to see or download the data from the Cloud server. To access the data from the Cloud server, the users have to be registered with the cloud server. So that the user have to register their details like username, password and a set of random numbers. This is information will stored in the database for the future authentication. Data Owner: Data Owner is the Person who is going to upload the data in the Cloud Server. To upload the data into the Cloud server, the Data Owner have be registered in the Cloud Server. Once the Data Owner registered in cloud server, the space will be allotted to the Data Owner.

6.2 Cloud Server of main

Cloud Server is the area where the user going to request the data and also the data owner will upload their data. Once the user send the request regarding the data they want, the request will first send to the Cloud Server and the Cloud Server will forward your request to the data owner. The data Owner will send the data the data the user via Cloud Server. The Cloud Server will also maintain the Data owner and Users information in their Database for future purpose.

6.3 Partition of data and encryption

In this module, once the data was uploaded into the cloud server, the Cloud server will split the data into many parts and store all the data in the separate data servers. In techniques wasn't used in proposed system so that there might be a chance of hacking the entire data. Avoid the hacking process, we splitting the data and store those data in corresponding data server. We're also encrypting the data segments before storing into the data server.

6.4 Key production Server

The encryption keys are stored in appropriate key servers. So that we can increase the security of the cloud network. If

Copyright © 2015 IJARCSET. All rights reserved.

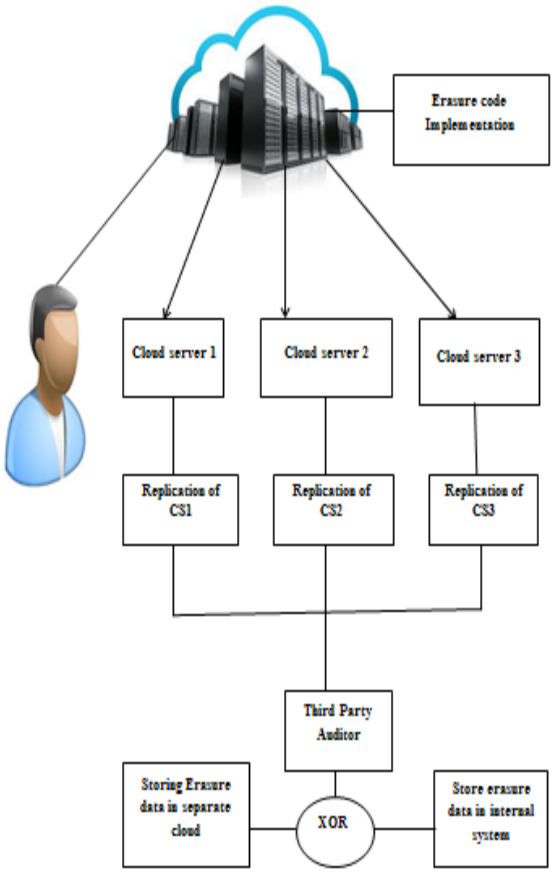
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

the user wants retrieve the data, they've to provide all the key that are stored in the appropriate key servers.

7 System Design

The architecture mainly based on the pull based technique, The PULL-based reconstruction can be envisioned as a master-worker computing model, in which a master triggers a reconstruction procedure by sending a set of read requests, and then waits for the requests to be completed by workers.



There are two classical reconstruction approaches in real-world erasure-coded storage clusters: i) a designated master (e.g., a replacement node) fetches k surviving blocks and reconstructs a failed block; and ii) each surviving node plays the role of a master (i. e., acting as a rebuilding node) and all surviving nodes perform as workers, where write I/O s of rebuilt blocks are spread out over all the surviving nodes. From the angle of message communication,

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

this 'Master Worker' pattern belongs to the category of PULL-type transmission. Throughout this paper, we refer to the reconstruction scheme using replacement nodes as PULL Rep. Also it depends on the The goal of the PUSH technique for node reconstruction is two-fold. First, PUSH aims to alleviate the reconstruction performance bottleneck caused by a replacement node's network bandwidth in PULL-Rep. Second, PUSH also aims to mitigate extra seeking times induced by the non-sequential disk accesses in PULL-Sur. In comparison to surviving nodes that passively respond to reconstruction reads in PULL, the surviving nodes in PUSH proactively participate in the entire reconstruction process.

8 Conclusion

From this, Cloud Based Data Recovery and Reconstruction System using Bi Methodology Erasure Code Implementation have been implemented. Nowadays a grand challenge for storage clusters is efficiently migrating data replicas to create an erasure-coded archive. To take this challenge, we are going to integrate the PUSH-type transmission into the archival migration in erasure-coded storage clusters. Moreover, since PUSH-based reconstruction schemes are sensitive to slow nodes, we plan to extend the PUSH-based reconstruction schemes for heterogeneous erasure-coded storage clusters by taking into account both load and heterogeneity of surviving nodes. To address these issues, we proposed the PUSH approach, in which a PUSH-type transmission is incorporated into node reconstruction. We developed two PUSH-based reconstruction schemes (i.e., PUSH Rep and PUSH-Sur). Compared to the PULL-based counterparts where surviving blocks are transferred in a synchronized 'M:1' traffic pattern, our PUSH-based reconstruction solutions support the '1:1' pattern, which naturally solves the In cast problem. We built performance models to investigate the reconstruction times of our PUSH-based schemes applied in large-scale storage clusters. We extensively evaluated the four schemes on a real-world cluster.

9 References

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran,

Copyright © 2015 IJARCSET. All rights reserved.

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

"Network coding for distributed storage systems," IEEE Trans. Inform. Theory, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[2] A. Kermarrec, N. Le Scouarnec, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in Proc. Int. Symp. Netw. Coding, 2011, pp. 1–6.

[3] A. Phanishayee, E. Krevat, V. Vasudevan, D. Andersen, G. Ganger, G. Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage systems," in Proc. 6th USENIX Conf. File Storage Technol., 2008, p. 12.

[4] Avani Wildani , Protecting Against Rare Event Failures in Archival Systems, April 2009

[5] B. Cassidy, J. Hafner, Space efficient matrix methods for lost data reconstruction in erasure codes," IBM Res., Armonk, NY, USA, Tech. Rep. RJ10415, 2007.

[6] B. Calder et al., "Windows azure storage: A highly available cloud storage service with strong consistency," in Proc. 23rd ACM Symp. Operating Syst. Principles, 2011, pp. 143–157.

[7]Bianca Schroeder, Understanding latent sector errors and how to protect against them, coding have a role to play in my data center?" Microsoft research MSR-TR-2010, vol. 52, 2010.

[8] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable performance of the panasas parallel file system," in Proc. 6th USENIX Conf. File Storage Technol.,vol.2,2008,pp.1–2.

[9] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in Proc. USENIX Annu. Tech. Conf., 2012, p. 2.

[10] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "Hydrastor: A scalable secondary storage," in Proc. 7th Conf. File Storage Technol., 2009, pp. 197–210.

Copyright © 2015 IJARCSET. All rights reserved.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

[11] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, 1960.

[12] J. Plank et al., "A tutorial on reed-solomon coding for fault-tolerance in raid-like systems," Softw. Practice Experience, vol. 27, no. 9, pp. 995–1012, 1997.

[13] K. Rao, J. Hafner, and R. Golding, "Reliability for networked storage nodes," IEEE Trans. Dependable Secure Comput., vol. 8, no. 3, pp. 404–418, May 2011.

[14] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in rdp code storage systems," ACM SIGMETRICS Per- form. Eval. Rev., vol. 38, no. 1, pp. 119–130, 2010.

[15] M. Aguilera, R. Janakiraman, and L. Xu, "Using erasure codes efficiently for storage in a distributed system," in Proc. Int. Conf. Dependable Syst. Netw., 2005, pp. 336–345.

[16]Michael Vrable,Cumulus: File system Backup to the Cloud,

[17] M. Holland, G. Gibson, and D. Siewiorek, "Fast, on-line failure recovery in redundant disk arrays," in Proc. 23rd Int. Symp. FaultTolerant Comput., 1993, pp. 422–431.

[18] M. Holland, G. Gibson, and D. Siewiorek, "Architectures and algorithms for on-line failure recovery in redundant disk arrays," Distrib. Parallel Databases, vol. 2, pp. 295–335, 1994.

[19] M. Manasse, C. Thekkath, and A. Silverberg, "A reed-solomon code for disk storage, and efficient recovery computations for erasure-coded disk storage," Proc. Inf., pp. 1–11, 2009.

[20] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in Proc. 10th USENIX Conf. File Storage Technol., 2012, pp. 251–264.

Copyright © 2015 IJARCSET. All rights reserved.

ISSN 2454-9924 Volume: 1 Issue: 1(2015)

[21] Q. Xin, E. Miller, T. Schwarz, D. Long, S. Brandt, and W. Litwin, "Reliability mechanisms for very large storage systems," in Proc. 20th IEEE/11th NASA Goddard Conf. Mass Storage Syst. Technol., 2003, pp. 146–156.

[22] T.J. Watson Research Center,Cryptographic Extraction and Key Derivation: The HKDF Scheme, 2010

[22] Q. Xin, E. Miller, and S. Schwarz, "Evaluation of distributed recovery in large-scale storage systems," in Proc. 13th IEEE Int. Symp. High Performance Distrib. Comput., 2004, pp. 172–181.

[23] RACS: A Case for Cloud Storage Diversity

Copyright © 2015 IJARCSET. All rights reserved.