ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 13, April 2016

DATA RECOVERY AND RECONSTRUCTION SYSTEM USING BI-METHODOLOGY ERASURE CODE IMPLEMENTATION

Lakshmi A[#1], Prof.C. Prakash Narayanan[*2]

[#1]*Final year, M.E, Computer Science And Engineering, P.S.V College of Engineering and Technology, Krishnagiri (D.T) - 635108*

[*2]*Assistant Professor,Computer Science And Engineering, P.S.V College of Engineering and Technology, Krishnagiri (D.T) - 635108*

laxbe12@gmail.com
cprakashmca@gmail.com

***Abstract*— A key design goal of erasure-coded storage clusters is to minimize reconstruction time, which in turn leads to high reliability by reducing vulnerability window size. PULL-Rep and PULL-Sur are two existing reconstruction schemes based on PULL-type transmission, where a renewal node initiates reconstruction by sending a set of read requests to ongoing nodes to retrieve surviving blocks.This paper presents a technique for constructing a code that can correct up to three errors with a simple, regular encoding, which admits very efficient matrix inversions.In order to support this variety of use cases on the ever increasing amount of data, a flexible infrastructure that scales up in a cost winning manner, is critical. In this paper we introduce a new set of codes for erasure coding called Local Reconstruction Codes (LRC). We present an algorithm that finds the optimal number of codeword symbols needed for recovery for any XOR-based erasure code and produces recovery schedules that use a minimum amount of data.**

Keywords*--- Erasure-coded storage cluster, *data discovery,* PUSH-type transmission, *map-reduce,* PULL-type transmission, *distributed systems.

I. INTRODUCTION

Erasure coded algorithm, we propose two PUSH-based reconstructionschemes—PUSH-Rep and PUSH-Sur—to improve reconstructionperformance in a distributed storage cluster. At

the heart of this study is the proactive PUSH techniquethat evenly distributes network and I/O loads among surviving nodes to shorten reconstruction times.The following three factors motivate us to propose thePUSH-based reconstruction technique for erasure-coded, clustered storage.This paper presents a new decentralized coordination algorithmfor distributed disk systems using deterministic erasurecodes. A deterministic erasure code, such as Reed-Solomon [12] or parity code, is characterized by two parameters,*m* and *n*.1 It divides a logical volume into fixed-size*stripes*, each with *m* stripe units and computes *n*−*m* parityunits for each stripe (stripe units and parity units havedecrypting locally is clearly impractical, due to the huge amount the same size). the storage system consists of several bricks where each brick or node is a sealed unit consisting of a controller, power supply, networking interfaces and disk drives. Several components in thenode represent single points of failure. In order to achievereliability goals associated with high-end enterprise-classstorage systems, redundancy has to be distributed acrossthe collection of nodes to tolerate both drive and nodeFor small values of and reasonably reliable devices, one checksumdevice is often sufficient for fault-tolerance. Thisis the ―RAID Level 5‖ configuration, and the codingtechnique is called *" +1-parity."* [4, 5, 6]. With +1-parity, the –though byte of the checksum device is calculated to be the bitwise exclusive-or (XOR) of the r -though byte of each data device. If anyone of the +1 device fails, it can be reconstructed as the XOR of the remaining devices. +1-parity is attractive becauseof its simplicity. It requires one extra storage device, and one extra write operation per write to any single device. Its maindisadvantage is that it cannot recover from more than one simultaneous failure.These are degraded reads to temporarily unavailabledata and recovery from single failures. Although erasurecodes tolerate multiple simultaneous failures, singlefailures represent 99.75% of recoveries [44].Two recentresearch projects have demonstrated how the RAID-6 codes RDP and EVENODD may recover from singledisk failures by reading significantly smaller subsets ofcodeword symbols than the previous standard practice ofrecovering from the parity drive [51, 49]. Our contributionsto recovery arrangement generalize these results toall XOR-based erasure codes, analyze existing codes todifferentiate them based on recovery performance, andexperimentally verify that reducing the amount of datausgiving recovery translates directly into improved performancefor cloud file systems, but not for typical RAID array configurations.Erasure coding provides potential storage andnetwork savings to replication. For example, an m-of-nerasure coding scheme encodes unit data into n *fragments*of size 1m such that any m of them reconstructsthe original data. While 3-way replication and 3-

of-5 erasurecoding both tolerate 2 faults, the former requires 3×In a practical erasure code setting, this is the core computationfor reconstructing a failed disk, using exclusive-or (XOR) as the addition operator,but the technique presented in this paper apply more generally to computingstreams of sums in arbitrary monodies (like groups, only withoutinverses).modification part of the Project, Data is encrypted, splitter and stored in separate Servers. Before doing Erasure Code Technique Data is Splitter into smaller parts and converted into Binary Data. Data owner adds Parity bit into the Data in order to increase the security model of the implementation. We also have Third Party Auditor (TPA) for further Agent for Verifying Data Integrity. Before going to TPA data is hashed using SHA 256 Algorithm. We use Erasure Code implementation for Code Reconstruction Technique. Auditor is deployed to verify the data.

II. Related Work

Generally, we using Proof Of Retrievability (POR) and Proof Of Data Possession (PDP) in cloud computing for repair the corrupted data and restore the original data but it putting all data in a single server. Then it uses MRPDP and HAIL method for regenerating code has to minimize repair traffic. But not reading and reconstructing the whole file during repair as traditional erasure code.SOR creates a number of reconstruction processes associated with strips [15]; DOR makes every surviving disks busywith reconstruction reads at all time [16];Optimizing decoding operations. Cassidy and Hanger proposed a code-specific hybrid reconstruction algorithm to reduce XOR operations and improve decoding performance during recovery.

Hydrators makes all remaining nodes contribute to data rebuilding (i.e., bulk rebuilding), which maximizes I/O utilization [14]. The Per-file RAID offered by Panamas allows a metadata manager to rebuild files in parallel.

The simplest method for availability is to stripe (distribute) data over conventional, high-reliability array bricks. No redundancy is provided across bricks, but each brick could usean internal redundancy mechanism such as RAID-1 (mirroring) or RAID-5 (parity coding). The second common alternative is to mirror (i.e., replicate) data across multiplebricks, each of which internally uses either RAID-0 (no redundant striping) or RAID-5. This section compares erasurecoding to these methods and show that erasure codingcan provide a higher reliability at a lower cost.

A .Data Owner:

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.

B .Main Cloud Server:

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.

C *.Data Splitting 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.

D. Key Server:

The encryption keys are stored in appropriate key servers. So that we can increase the security of the cloud network. If the user wants retrieve the data , they‘ve to provide all the key that are stored in the appropriate key servers.

E. Party Bit Addition And Erasure Code

Once the data are stored in the corresponding data servers and the keys are stored in the key servers. Then we‘re adding the parity bits to the data, so that the data will be changed. Also we‘re applying the Erasure Code by using the XOR operation, while XORing the block data , the data will be converted in binary data.

F. Trusted Party Auditor:

Once added the parity added bits, then the data will be given to the Trusted Parity auditor. The Trusted Parity Auditor will generate the signature using change and response method. The data will be audited in this module, if any changes occurs it will provide the intimation regarding the changes.

G. Replica Server:

We‘ll maintain the separate Replica Cloud server. If suppose the data in the data server was lost, then the Main Cloud server will contact the Replica Cloud server and get the data from the Replica Cloud Server. By using this concept, we can get the data if any data loss occurs.

III. Reconstruction

Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

Partition detection and erasing routes:

Partitions are detected when a *reversal* reaches a node with no downstream links and all of its neighbors have the same *reflected reference level*, which it previously defined. A node that detects a partition initiates the process of erasing the invalid routes

IV. Recovery Optimization

Workload-based approaches to improving recovery are independent of the choice of erasure code and apply to minimum I/O recovery algorithm and rotated RS codes that we present.

Regenerating codes provide optimal recovery bandwidth [12] among storage nodes. This concept is different than minimizing I/O; each storage node reads all of

its available data and computes and sends a linear combination.

Regenerating codes were designed for distributed systems in which wide-area bandwidth limits recovery performance.

Cost during large-scale recovery:

A main penalty paid by erasure coding is decreased performance during recovery (when compared to replication). This might be

a serious concern when erasure coding is considered as the redundancy mechanism across data centres‘, or across availability domains within a large data centred ascending order. The best files are given as output to the main cloud server. The main cloud server retrieves top files and given as output to the user.

reconstruction models to predict performance of PUSH (i.e., PUSH-Rep, andPUSH-Sur) as well as the existing

counterparts (i.e., PULL-Rep,PULL-Sur). The accuracy of all the four performance models is validated against the results collected on a real-world storage cluster.

3.1 Reconstruction Time

Let R<PULL-Rep;PUSH-Rep> be a ratio between the reconstruction times of PULL-Rep and PUSH-Rep. We can deriveR<PULL-Rep;PUSH-Rep> from Eqs. The read throughput is larger than the receiving bandwidth (120 versus 800 Mbps), so the ratio R<PULL-Rep;PUSH-Rep> is approximated as k.

V. Working Methods

The /Var/Lib/Mysql.

Regardless of the storage engine, every MySQL table you create is represented, on disk, by a .frm file, which describes the table‘s format (i.e. the table definition). The file bears the same name as the table, with a .frm extension. The .frm format is the same on all platforms but in the description of the .frm format that follows, the examples come from tables created under the Linux operating system.

```
/var/lib/mysql/db.frm        #Table definition
/var/lib/mysql/db.MYD        #MyISAM data file
/var/lib/mysql/db.MYI        #MyISAM Index file
/var/lib/mysql/ibdata1       #Innodb data file
```

To perform the recursion, let‘s suppose that the extended schedule for *n* nodes is (*T*0,*T*1, ...,*Tk*, ...). For any set *Tk* from the schedule, define Double(*Tk*) to be the result of replacing every occurrence of node number *i* by 2*i*+1, incrementing all subscripts by one, and adding in links 2*i*!*k* 2*i*+1, except if a previously doubled set contains an edge *x* !*k* 2*i*+1, or an edge 2*i* !*k x*, in which case we instead

replace all occurrences of *i* by 2*i*, and add link 2*i*+1!*k* 2*i*. In every tree so constructed, the first communication step connects the paired nodes 2*i* and 2*i*+1, and edges in the original schedule turn into edges between paired nodes in different pairs.

.

Algorithm for Reconstruction;

When data disk i fails, the algorithm is applied for F = {di,0, . . . , di,r−1}. When coding disk j fails, F ={cj,0, . . . , cj,r−1}. If a storage system rotates the identities of the disks on a stripe-by-stripe basis, then the average number of symbols for all failed disks multiplied by the total number of stripes gives a measure of the symbols required to reconstruct a failed disk.

We easily see inductively that the property of consecutive summation is preserved. Each doubled tree is individually a valid tree for computing a sum; half of the elements appear on the left with subscript *k*, and the other half participate in a tree which previously worked, appropriately relabeled. Suppose that some edges conflict between trees. This means that some earlier tree contains an edge *x* !*k*+*m y*, and this tree contains either *x* !*k*+*m z* or *z*!*k*+*m y*, for some *x*, *y*, *z*, and *m*. We know that *x* and *y* must not be paired; all edges between paired nodes happen in the first step. Thus, an edge between $x/2$ and $y/2$ existed in the original sequence. If $m > 0$, the same is true of x,z or z,y, contradicting our inductive hypothesis. Hence $m = 0$, but then we avoided collisions by construction.

This example Swing application creates a single window with "Hello, world!" inside:

```
// Hello.java (Java SE 5)
import javax.swing.*;


 public class Hello extends JFrame {
 public Hello() {
setDefaultCloseOperation(WindowConstants.EXIT_ON_CLO
SE);
add(new JLabel("Hello, world!"));
 pack();
 } public static void main(String[] args)
  { new Hello().setVisible(true);
  }}
```

VI. CONCLUSION

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.

REFERENCES

[1] S. Frolund, A. Merchant, Y. Saito, S. Spence, and A. Veitch, —A decentralized algorithm for erasure-coded virtual disks,‖ in Proc. Int. Conf. Dependable Systems Networks, 2004, pp. 125–134.

[2] M. Storer, K. Greenan, E. Miller, and K. Voruganti, —Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage,‖ in Proc. 6th USENIX Conf. File Storage Technol., 2008, p. 1.

[3] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sarma, R. Murthy, and H. Liu, —Data warehousing and analytics infrastructure at facebook,‖ in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 1013–1020.

[4] Z. Zhang, A. Deshpande, X. Ma, and E. Thereska, —Does erasure coding have a role to play in my data center?‖ Microsoft research MSR-TR-2010, vol. 52, 2010.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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