International Journal of Soft Computing 13 (1): 25-30, 2018
ISSN: 1816-9503

A Novel Shorten Erasure Based Reed Solomon Fault Tolerance Code for Road Traffic Data Fault Tolerance

[1]Md. Rafeeq, [2]C. Sunil Kumar and [3]N. Subhash Chandra
[1]Department of CSE, CMR Technical Campus (CMRTC), Hyderabad, Telangana, India
[2]Department of IT, Sreenidhi Institute of Science and Technology (SNIST), Yamnampet, Ghatkesar, Hyderabad, Telangana, India
[3]Department of CSE, CV Raman College of Engineering (CVRCE), Manglapalli, Ibrahimpatnam R.R (D), Telangana, India

Abstract: The massive growth in road traffic and subsequence generation of traffic related data insisting the researcher to proceed for the analytical research on the traffic prediction. However, the gigantic size of the data and chances of storage failure may cause the purpose inefficient. The advancement in technologies and high demand for fault tolerant storage solutions most of the cloud based commercial storage service providers are now equipped with erasure based Reed-Solomon fault tolerance mechanism. However, the additional cost for replication is still an overhead for service providers and customers. In this research, we propose a novel erasure based code and further optimization as shortening the proposedcode also for the digital storage formats. The research also results into a comparative study of cost analysis for commercial cloud based storage service providers. Finally, the research demonstrates the improvement in code shortening and making the performance higher.

Key words: Erasure, Reed-Solomon, code shortening, performance comparison, evolution application, response time comparison, Dropbox, Google Drive, Hightail, OneDrive, SugarSync

INTRODUCTION

In the past years, the high upcoming demand for storage with high performance and reliability were been understood (Mallikharjuna and Anuradha, 2015, 2016; Rao and Anuradha, 2016). The industry was approaching towards a phase where the lack of standardization of digital storage was limiting the applications to make storage more reliable for commercial storage providers. The major bottleneck for the standardization was the non-standard storage solutions used by different service providers. In the early 80's, the industry adopted cloud computing for distributed storage solutions. The effort was well recognized and multiple companies came together to form a consortium in order to frame the standardization for digital storage.

As far as data storage is concerned, there are multiple schemes are available to improve file and data compression. The other most influencing parameters for instance, a data file that is uploaded and accessed on the server may seriously be effected by the network bandwidth as well as the server workload. This will degrade the efficiency (Mallikharjuna and Anuradha, 2015, 2016). Moreover, the cloud storage services deals with a great scope and domain of the data being storage and retrieved along with the frequency of access varying depending on the mode of the operation performed on the data (Kubiatowicz *et al*., 2000). Offering unlimited storage container space might cause a high economic drawback on the cloud storage provider and as well as the users due to inefficient storage (Druschel and Rowstron, 2001). Hence, a technique or automation is needed to find the best suitable storage structure based on cost and other influencing factors. There are many free offerings of the cloud storage services; however, they may not suite the application requirement to the best always (Adya *et al*., 2002).

Two major companies, Philips and Sony took the major initiative to define the standard storage formats in digital media. The standard is well accepted today and been referred as compact storage format. This standard format is majorly used for achieving any data which also reduces the storage cost compared to the early storage formats. However, the compact storage format has limitations in order to achieve high availability. It is difficult to predict how a storage media gets corrupted. In the earlier studies we have understood the reasons for storage device failure. Henceforth, we realise the following errors for storage failures as:

- The additional noise affecting the storage during transmission or during retrieval
- Mishandling of the removable devices

Corresponding Author: Md. Rafeeq, Department of CSE, CMR Technical Campus (CMRTC), Hyderabad, Telangana, India

Table 1: Cost comparison for Dropbox (data load in GB)

Data load (GB)	Costs (US$)
100	99
200	99
300	99
400	499
500	499
1000	Not available
>1000	Not available

Table 2: Support for mobile based cloud applications in Dropbox

Client OS types	Support
Apple iPhone operating systems	Available
Android mobile operating systems	Available
Blackberry operating systems	Available
Microsoft mobile operating system	Available

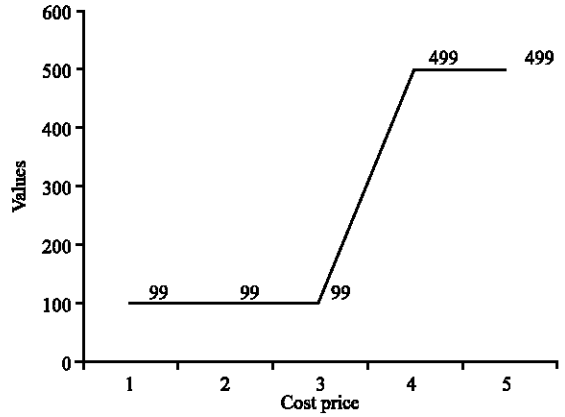

Fig. 1: Cost comparison for Dropbox

The most important improvement in the recent time for fault tolerance in digital media storages is the Reed-Solomon code. The basic benefit of the Reed-Solomon codes is to rearrange, so that, the timely restoration can be achieved for storage devices. Thus, in this research, we concentrate on further enhancement of the erasure based fault tolerance mechanism.

Commercial cloud storage services: As the choice of storage services from cloud is not limited and most of those are configured to give best advantages for specific type of data and operation, we compare most of the services here (Adya *et al*., 2002; Haeberlen *et al*., 2005; Tang, 2008).

Dropbox: The Dropbox is a storage service which is available for client side access for windows systems, Linux systems, Macintosh systems, Blackberry mobile operating systems android mobile operation systems and finally the iPhone operating systems. The free basic account comes with a paltry 2 GB of storage. For document based applications this is huge. The storage service is good choice for applications using the container for read only data (Table 1 and 2). Here, we provide a graphical representation of the cost price comparison (Fig. 1).

Table 3: Cost comparison for Google Drive (coast 60-9600)

Data load (GB)	Costs (US$)
100	60
200	120
300	120
400	240
500	240
1000	600
>1000	1200-9600

Table 4: Support for mobile based cloud applications in Google Drive

Client OS types	Support
Apple iPhone operating systems	Available
Android mobile operating systems	Available
Blackberry operating systems	Not Available
Microsoft mobile operating system	Not Available

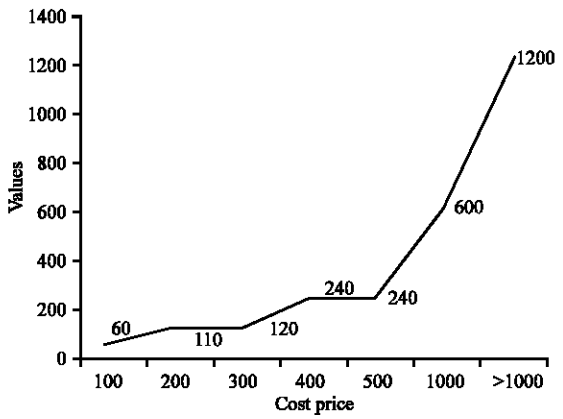

Fig. 2: Cost comparison for Google Drive

Google Drive: The most popular cloud storage service is drive storage from Google. The basic account comes with 15 GB of storage for a new customer account or an existing account created with Google email. The highest rated benefit of the Google Drive is the service can be also be integrated with other existing Google services for storing various types of data from other services (Table 3 and 4). Here, we provide a graphical representation of the cost price comparison (Fig. 2).

Hightail: The previous version of business cloud storage of Hightail was popular by name of YouSendIt. The basic reason for creating the name was the core of the features that Hightail provides. Hightail is majorly known for sharing files which can be digitally signed for verifications. The core technology behind this provider is link sharing where the sender can upload a file and the link to that same file can be shared with the recipient. The recipient can click on the link to download the same. This service is popular for business users as it provides the private cloud storage and the desktop version of the client which can be used for syncing local files to the cloud storage (Table 5 and 6).

OneDrive: The OneDrive was previously popular as SkyDrive. The functionalities are mostly same as Dropbox.

Table 5: Cost comparison for Hightail

Data load (GB)	Costs (US$)
100	Free
200	Free
300	Free
400	Free
500	Free
1000	Free
>1000	195

Table 6: Support for mobile based cloud applications in Hightail

Client OS types	Support
Apple iPhone operating systems	Available
Android mobile operating systems	Not available
Blackberry operating systems	Not available
Microsoft mobile operating system	Not available

Table 7: Cost comparison for OneDrive (data load)

Data load (GB)	Costs (US$)
100	50
200	100
300	Not available
400	Not available
500	Not available
1000	Not available
>1000	Not available

Table 8: Support for mobile based cloud applications in OneDrive

Client OS types	Support
Apple iPhone operating systems	Available
Android mobile operating systems	Available
Blackberry operating systems	Available
Microsoft mobile operating system	Available

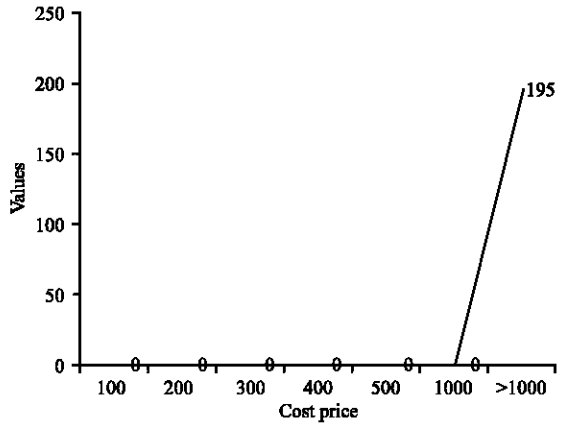

Fig. 3: Cost comparison OneDrive

The most important factor for this storage service is that the client version is available for windows systems, Linux systems, Macintosh systems, Blackberry mobile operating systems android mobile operation systems and finally the iPhone operating systems. Moreover, the supports for social media plug-ins are also available here. This feature makes the application more compatible with other applications to access data directly. Here, we provide a graphical representation of the cost price comparison (Fig. 3).

Table 9: Cost comparison for SugerSync (data load)

Data load (GB)	Costs (US$)
100	99
200	250
300	250
400	250
500	250
1000	550
>1000	Pay per use

Table 10: Support for mobile based cloud applications in SugerSync

Client OS types	Support
Apple iPhone operating systems	Available
Android mobile operating systems	Available
Blackberry operating systems	Available
Microsoft mobile operating system	Available

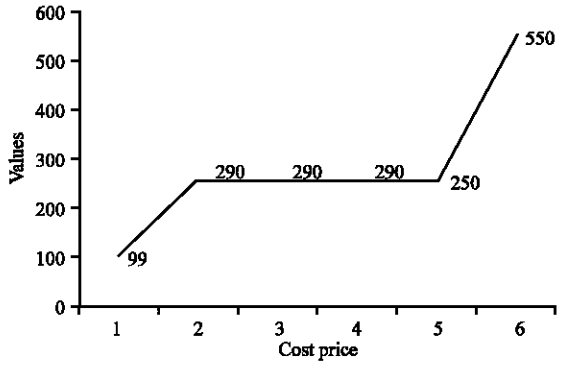

Fig. 4: Cost cmparison for SugarSync

SugarSync: The SugarSync is majorly popular among business users for its effective and fast online backup solutions. The service can also be used for complete folder and individual file syncing with multiple applications and multiple users. Moreover, the service provides a unique function to share the stored content over multiple devices at same point of time but with different permission levels. The most important factor for this storage service is that the client version is available for android mobile operation systems and also the iPhone operating systems (Table 9 and 10). Here, we provide a graphical representation of the cost price comparison (Fig. 4).

Reed-Solomon code for fault tolerance: The most important factor that makes Reed-Solomon framework to implement is the simplicity. Here, in this research we consider the scenario to compare the performance of Reed-Solomon and proposed encoding technique. We consider there will be K storage devices each hold n bytes of data such that:

$$D = \sum D_1, D_2, D_3, \ldots, D_k \quad (1)$$

where D is the collection of storage devices. Also, there will be L storage devices each hold n bytes of check sum data such that:

$$C = \sum C_1, C_2, C_3, ..., C_L \quad (2)$$

where C is the collection of Checksum devices. The checksum devices will hold the calculated values from each respective data storage devices.

The goal is to restore the values if any device from the C collection fails using the non-failed devices. The Reed-Solomon deploys a function G in order to calculate the checksum content for every device in C. Here, for this study, we understand the example of the calculation with the values as K = 8 and L = 2 for the devices C_1 and C_2 with G_1 and G_2, respectively (Shao and Cao, 2009).

The core functionalities of Reed-Solomon is to break the collection of storage devices in number of words (Zhang *et al*., 2013; Bellorado and Kavcic, 2010; Garcia-Herrero *et al*., 2011; Jiang and Narayanan, 2008). Here, in this example, we understand the each number of words is of u bits randomly. Hence, the words in each device can be assumed as v where v is defined as:

$$v = (n \text{ bytes}) \times \left(\frac{8 \text{ bits}}{\text{byte}}\right) \times \left(\frac{1 \text{ word}}{u \text{ bits}}\right) \quad (3)$$

Furthermore, v is defined as:

$$V = \frac{8n}{u} \quad (4)$$

Henceforth, we understand the formulation for checksum for each storage device as:

$$C_i = W_i \times (D_1, D_2, D_3, ..., D_k) \quad (5)$$

where the coding function W is defined to operate on each word. After the detail understanding of the erasure fault tolerance scheme, we have identified the limitations of the applicability to the cloud storage services and propose the novel scheme for fault tolerance in this research in the next study.

MATERIALS AND METHODS

Proposed novel fault tolerance scheme: With the understanding of the limitations of existing erasure codes to be applied on the cloud based storage systems as the complex calculations with erasure codes will reduce the performance of availability measures significantly. Thus, we make an attempt to reduce the calculation complexities with simple mathematical operations in the standard erasure scheme.

The checksum for storage devices are considered as C_i from the Eq. 5. We propose the enhancement as the following formulation for checksum calculation:

$$C_i = W_i \times (D_1, D_2, D_3, ..., D_k) = W_i (D_1 \oplus D_2 \oplus D_3, ..., \oplus D_k) \quad (6)$$

Here, the XOR operation being the standard mathematical operation most suitable for logical circuits used in all standard hardware makes it faster to be calculated. Also, we redefine the function to be applied on each word for the storage devices D as following:

$$W = \begin{bmatrix} w_{1,1} & ,\cdots, & w_{1,L} \\ . & ,\cdots, & . \\ . & ,\cdots, & . \\ . & ,\cdots, & . \\ w_{K,1} & ,\cdots, & w_{K,L} \end{bmatrix}_{K \times L} \quad (7)$$

The proposed matrix will be stored on one of the devices and will be recalculated only once. As the modified checksum formulation is an XOR operation, thus which will automatically notify in case of any change. Furthermore, we optimize the proposed code framework in the next study.

Optimizing proposed novel fault tolerance scheme: The Reed-Solomon code is expressed by the power of coefficient denoted by n for the data blocks where n is expressed as:

$$n = 2^m - 1 \quad (8)$$

And the code blocks are represented as:

$$k = 2^m - 1 - 2t$$

Where:
m = The number of bits per data
t = The capability of correcting errors

In general the Reed-Solomon code considers an 8 bit data and 2 bit code, the error correcting code can be represented as 255,251 code.

Here, in this part of the research, we try to optimize the code length further to reduce the replication cost. The steps of the optimization algorithm are explained here:

Optimization algorithm:

Step 1; first, we consider the effective code in (255,251) block where the code is consisting of zero and non-zero codes
Step 2; then, we find the number of zero codes in the segment. For instance the numbers of zero codes are 227 in the code block. These codes will not have any effect in the error correction and fault tolerance mechanism
Step 3; then, we find the effective block of the code as (28, 24) for a 2 bit error correction code
Step 4; hence, as a final outcome of the optimization technique, we got the optimized code block

RESULTS AND DISCUSSION

To simulate and understand the improvement in the outcomes we implement the Reed-Solomon code with the

Table 11: Initial data block

Decimal	----------BCD----------			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Table 12: Addition table

Addition	a^0	a^1	a^2	a^4	a^5	a^6	a^7
0|0	a^0	a^1	a^2	a^3	a^4	a^5	a^6
a^0	a^0	a^4	a^8	a^{14}	a^1	a^{10}	a^{13}
a^1	a^1	a^{40}	a^5	a^9	a^0	a^2	a^{11}
a^2	a^2	a^8	a^{50}	a^6	a^{10}	a^1	a^3
a^3	a^3	a^{14}	a^9	a^{60}	a^7	a^{11}	a^2
a^4	a^4	a^1	a^0	a^{10}	a^{70}	a^8	a^{12}
a^5	a^5	a^{10}	|a^2	a^1	a^{11}	a^8	a^9
a^6	a^6	a^{13}	a^{11}	a^3	a^2	a^{12}	a^{90}

Table 13: Multiplication table

Multiplication	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7
0|0	0	0	0	0	0	0	0	0
a^0|0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7
a^1|0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8
a^2|0	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9
a^3|0	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
a^4|0	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}
a^5|0	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}
a^6|0	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}

Table 14: Fault tolerance results

Parameters	Generic RS	Proposed optimized RS
Initial polynomial	a^1 a^3 a^5	a^1 a^3 a^5
Encoded data	a^5 a^3 a^1 a^6 a^4 a^2 a^0	0 00 a^6 a^4 a^2 a^0
Fault tolerance code	a^5 a^3 a^1 a^6 a^4 a^2 1	a^6 a^4 a^2 1
Optimization reduction	0%	57%

enhancement and optimization proposed in this research. We accept any random data as the initial data block for the testing (Table 11). Based on the modified fault tolerance scheme, we realise the addition and multiplication table (Table 12 and 13).

Henceforth, we compare the results of the generic Reed-Solomon coding and the proposed fault tolerance technique (Table 14) based on the initial code.

CONCLUSION

In this study, the commercial cloud storage services are been compared based on the cost and performance factors. The result of the comparative measures provided the understanding of the demand for highly reliable and cost effective fault tolerance system. Henceforth, in this research, we study the core Reed-Solomon fault tolerance mechanism based on erasure codes. The research contributes towards the improved performance code for fault tolerance for digital storage devices rather than magnetic. Also,the research enhanced the performance of the proposed technique by applying the improvement in terms of optimization. The result of the proposed optimization technique is 57% reduction in the storage cost without negotiating with the fault tolerance reliability.

REFERENCES

Adya, A., W.J. Bolosky, M. Castro, G. Cermak and G. Cermak *et al*., 2002. Farsite: Federated, available and reliable storage for an incompletely trusted environment. Proceedings of the 5th symposium on Operating systems design and implementation, Volume, 36, December 9-11, 2002, Boston, Massachusetts, USA., pp: 1-14.

Bellorado, J. and A. Kavcic, 2010. Low-complexity soft-decoding algorithms for reed-solomon codes-Part I: An algebraic soft-in hard-out chase decoder. IEEE. Trans. Inf. Theory, 56: 945-959.

Druschel, P. and A. Rowstron, 2001. PAST: A large-scale, persistent peer-to-peer storage utility. Proceedings of the 8th Workshop on Hot Topics in Operating Systems, May 20-22, 2001, IEEE, Elmau, Germany, ISBN:0-7695-1040-X, pp: 75-80.

Garcia-Herrero, F., J. Valls and P.K. Meher, 2011. High-speed RS (255, 239) decoder based on LCC decoding. Circuits Syst. Signal Process., 30: 1643-1669.

Haeberlen, A., A. Mislove and P. Druschel, 2005. Glacier: Highly durable, decentralized storage despite massive correlated failures. Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation Volume 2, May 2-4, 2005, USENIX Association, Berkeley, California, USA., pp: 143-158.

Jiang, J. and K.R. Narayanan, 2008. Algebraic soft-decision decoding of reed-solomon codes using bit-level soft information. IEEE. Trans. Inf. Theory, 54: 3907-3928.

Kubiatowicz, J., D. Bindel, Y. Chen, S. Czerwinski, P. Eaton and D. Geels, 2000. Oceanstore: An architecture for global-scale persistent storage. ACM Sigplan Notices, 28: 190-201.

Mallikharjuna, K.R. and K. Anuradha, 2015. An efficient method for software reliability growth model selection using modified particle swarm optimization technique. Intl. Rev. Comput. Software, 10: 1169-1178.

Mallikharjuna, R.K. and A. Kodali, 2017. An efficient method for enhancing reliability and selection of software reliability growth model through optimization techniques. J. Software, 12: 1-8.

Mallikharjuna, R.K. and K. Anuradha, 2016. A new method to optimize the reliability of software reliability growth models using modified genetic swarm optimization. Intl. J. Comput. Appl., 145: 1-8.

Rao, K.M. and K. Anuradha, 2016. Performance evaluation of software reliability growth models using optimization techniques. Intl. J. Comput. Sci. Inf. Secur., 14: 355-362.

Shao, J. and Z. Cao, 2009. CCA-secure proxy re-encryption without pairings. Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography, Vol. 5443, March 18-20, 2009, Springer, Irvine, California, USA., pp: 357-376.

Tang, Q., 2008. Type-based proxy re-encryption and its construction. Proceedings of the 9th International Conference on Cryptology in India Vol. 8, December 14-17, 2008, Springer, Kharagpur, India, pp: 130-144.

Zhang, W., H. Wang and B. Pan, 2013. Reduced-complexity LCC reed-solomon decoder based on unified syndrome computation. IEEE. Trans. Very Large Scale Integr. Syst., 21: 974-978.