Erasure Codes for Distributed Storage

John Fritsche

January 30, 2024

Abstract

This survey paper focuses on erasure coding techniques utilized on distributed storage systems such as RAID. There is a specific emphasis on Reed-Solomon codes due to their ubiquity within distributed storage systems. Advances in coding techniques have substantial impacts on the cost and performance of storage systems. Several methods of utilizing Reed-Solomon codes are examined for their various efficiencies in computational complexity.

1 Introduction

The twenty first century has begun with a technological revolution that focuses on information stored in computer systems as data. This boom is in large part due to a shift in commerce made possible by the internet. For the second quarter of 2021 Retail E-Commerce totaled $222.5 billion dollars in sales [1]. The number of businesses offering products and services through the internet continues to rapidly grow. Social media applications thrive on massive user communities using content, mainly high quality video and photos, to connect with other users. Other services made possible by the Internet-of-Things (IoT) can regularly generate large amounts of data for everything from optimizing refrigeration to controlling lights with a mobile application. Every service on the internet requires information in the form of data. The internet, as a connection medium, would not be of much use if there were no information for users to process, create, or act upon. The central use of information in communications has lead to many advanced techniques of ensuring integrity of information. Data centers must offer cost efficient solutions to store, retrieve, and manage the data required for their customers' applications. The storage must be reliable, scalable, and low cost. Cost can be minimized by utilizing the smallest number of storage devices required to store the relevant data; however, various reliability methods can increase the number of drives required to ensure errors are mitigated. Economies of Scale allow for data centers to serve many customers for a reasonable price that ensures reliable service. This scalability has been delivered by using Distributed Storage Systems.

1.1 What is Distributed Storage?

Many computing systems require massive amounts of data to be stored and accessible to back-end services; however, many computing systems utilize wireless connections and use a client device that connects to centrally located data centers. This need for large amounts of data storage has given rise to the practice of distributing storage across many drives. This helps reduce cost and risk by allowing for the addition of more storage as the needs of customers grow. In 2010, Google's data warehouses typically had over ten thousand machines with six 1TB SSD drives per machine and

1 GB per second network connections [2]. This equates to 60 Petabytes of storage per warehouse. The current amount of storage is likely much higher. These drives must all work together to create a reliable storage service. Reliability of storage devices can greatly vary. Drives can fail for many reasons and experience errors during reading and writing, so the distributed storage paradigm has become very practical and popular to mitigate risk. Despite drive failures being common, large-scale distributed storage systems provide high partition-tolerance and availability by distributing both storage and computation over multiple nodes [3].

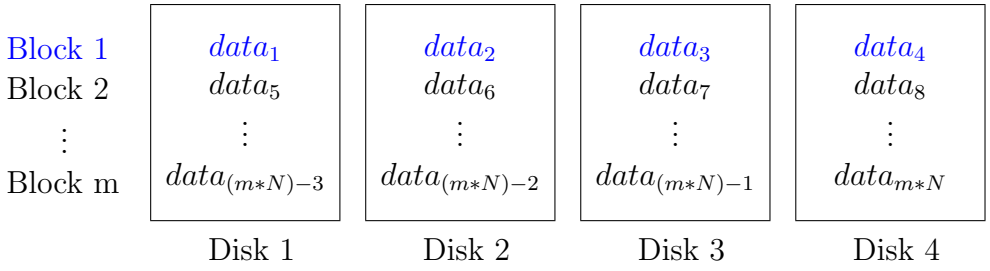

Figure 1: A diagram of RAID architecture with four disks, where m is the number of blocks and N is the number of disks. In this example $N = 4$

Redundant Array of Independent Disks (RAID) architectures, similar to the diagram shown in figure 1, are designed to allow distributed storage systems to recover from disk level failures. In this paper a matrix will be used to serve as a model for the RAID architecture and its contents. Blocks consist of common indices across rows of the disk array. Facebook's data center experiences over 98% of all their failure modes on a single block [4]. These blocks will be represented as rows within a matrix. The disks will be represented as columns within the matrix. Errors often occur at the data and disk level. The matrix representation is shown in figure 2.

	Disk 1	**Disk 2**	...	**Disk N**
Block 1	d_1	d_2	...	d_N
Block 2	d_{N+1}	d_{N+2}	...	d_{N+N}
Block 3	d_{2N+1}	d_{2N+2}	...	d_{2N+N}
⋮	⋮	⋮	⋱	⋮
Block m	$d_{(m-1)N+1}$	$d_{(m-1)N+2}$	...	d_{Nm}

Figure 2: A matrix model of the RAID architecture.

The "exclusive or" (XOR) function is used to derive parity bits, because it is very computationally efficient. The XOR function represents modulo two arithmetic which is fundamental to computer systems and is implemented via circuits. Using an XOR function, one can quickly create check values that ensure errors are either detected or corrected. These check values are known as parity bits. They can be used for data reconstruction depending on how many bits and/or disks fail. Simple parity bits are constructed using a row wise XOR sum as shown in equation 1.

$$[H]p_1 = \oplus_{c=1}^{N} d_c \tag{1}$$

Some paritys use approaches involving a combination of various data and parity bits. The EVEN-ODD code developed by IBM in 1994, implemented on RAID5, uses two parity disks to tolerate two disk failures [5]. Parity bits are best constructed using a system of dependent equations where

many of the terms are known and reside across many unique disks. This creates resiliency to disk failure.

These algorithms for construction of parity bits and data reconstruction are called Error-Correction Codes (ECCs). They correct data errors by intentionally adding redundant information. These codes can detect and correct varying numbers of errors based on the number of redundant bits utilized and their corresponding parity bits [6]. Storage drives can experience erasures and error bits. If the symbol is in error it may change it's value from one known value to another. An erasure can have too much noise to map to a known value. When erasures are present the Forward Error-Correction (FEC) code may be able to determine the true value of the erasure symbols. There are many algorithms designed for erasure coding. Erasure codes are Forward Error-Correction (FEC) codes that utilize the concept of erasures rather than errors. Each with various strengths and weaknesses. In the remainder of this paper we will explore several methods of Erasure Coding.

2 Reed Solomon Codes

In June of 1960 Irving Reed and Gustave Solomon published "Polynomial Codes Over Certain Finite Fields" [7]. It established an algorithm that is fundamental to most modern storage and communication systems. Reed-Solomon (RS) codes are Maximum Distance Separable (MDS) which means they have the best possible minimum code distance and for the message size, k, and codewords size, n, it can correct the highest number of erasures. RS codes satisfy the Singleton bound shown in equation 2.

$$d \leq n - k + 1 \tag{2}$$

The number of erasures a RS code can correct is one less than the minimum distance, d, of the code as shown in equation 3.

$$\text{Number of Correctable Erasures} = n - k \tag{3}$$

RS codes work on both errors and erasures, and utilize a generator matrix G that is a Vandermonde matrix as shown in equation 4. The elements of the generator matrix are distinct non-zero elements of the Galois field $GF(p^t)$, where p is prime.

$$G = \begin{bmatrix} 1 & 1 & \dots & 1 \\ a_1^1 & a_2^1 & \dots & a_n^1 \\ a_1^2 & a_2^2 & \dots & a_n^2 \\ \vdots & \dots & \ddots & \vdots \\ a_1^{k-1} & a_2^{k-1} & \dots & a_n^{k-1} \end{bmatrix} \tag{4}$$

The operations required for decoding codewords must invert this Vandermonde matrix and is computationally very expensive [8]. For more details on the implementation of RS codes see the RS tutorial by J. S. Plank [9]. The computational expense of this inverse taken over a finite field has led to research in optimizing the implementation of RS codes. Cauchy Reed-Solomon codes have higher performance than Standard RS codes in all cases [8].

2.1 Cauchy Reed-Solomon Codes

In 1995, Blomer et al published a paper that uses Cauchy matrices to increase the computational efficiency of the inversion operation, and uses discrete logarithms to transform non-binary polynomial arithmetic into efficient XOR-based computations [10].

Plank and Xu discovered the construction of the Cauchy matrix should use the minimal amount of ones and can have a performance impact on average of 10% and as much as 83% [8]. Cauchy RS codes have become widely used across the world with Google's Colossus using RS(9,6) and Facebook storage using RS(14,10) [2, 11].

2.2 Systematic Reed-Solomon Codes

The systematic form of a RS code can be used when data storage within an unencoded state is desired. The inversion of the systematic generator Vandermonde matrix, as shown in equation 5, is a very computationally costly operation.

$$G = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & 0 & a_1^1 & a_2^1 & \dots & a_n^1 \\ 0 & 0 & \dots & 0 & a_1^2 & a_2^2 & \dots & a_n^2 \\ \vdots & \dots & \ddots & \vdots & \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & a_1^{k-1} & a_2^{k-1} & \dots & a_n^{k-1} \end{array}\right] \tag{5}$$

Mattoussi et al. proved the use of a Hankel matrix, equation 6, can be as much as 157 times faster than the Vandermonde method [12]. This technique uses the upper triangle of the Hankel matrix to get a rectangular sub-matrix of non-singular values and remove the need for Vandermonde matrix inversion[12]. The Hankel matrix is defined over GF(p) and has coefficients that are constant along the diagonals [12].

$$B_p = \begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_{q-2} & b_{q-1} \\ b_2 & b_3 & . & \dots & b_{q-1} & \\ b_3 & . & . & \dots & & \\ . & . & . & & & \\ b_{q-2} & b_{q-1} & & & & \\ b_{q-1} & & & & & \end{bmatrix} \tag{6}$$

Where the elements $b_i = \frac{1}{1-y^i}$ for $1 \leq i \leq p-1$ and y is an arbitrary primitive element of $GF(p)$ and y^i is over $GF(p)$. These properties allow every square sub-matrix to be non-singular [12].

3 Discussion

There are various trade offs that should be considered when deciding on what ECC to utilize. Storage cost is an important measure to optimize in order to maintain a valid business case. Erasure coding allows for the reduction in the amount of storage required to maintain information. This can reduce storage costs by over 50% through savings in hardware, facility size, and power costs [13]. The probability of encountering a failed storage drive increases with the number of drives accessed per read. More read operations also add to network overhead and I/O costs [13]. Khan et al. introduced Rotated Reed-Solomon codes to increase the performance of degraded reads for single disk failures [14]. The first parity bit used is a row-wise XOR and the second parity bit is a wrapping combination between multiple rows of data bits. This code leads to significant improvements for single-disk failure degraded reads [14].

Reliability is another major consideration. Microsoft does not use a RS code, but instead created a Local Reconstruction Code (LRC) (16,12) Pyramid Code for their Windows Azure Storage (WAS) [13]. This decision was due to a requirement to have three replicas of the data and and a storage overhead of 1.33x. A RS code of (6,3) would have a $1.5x$ storage overhead [13].

4 Conclusion

Erasure coding for distributed storage systems are algorithms that impact all users of cloud based storage services. Optimizing between space and complexity can have significant impact on large user communities. With modern advances in coding theory and machine learning combinations of multiple techniques show promise to deliver optimal solutions to all stakeholders of the system [4]. The original RS code has become one of the most popular algorithms in distributed storage systems; however, trade-offs must still be analyzed to ensure the solutions purpose is attained.

This survey paper focused on describing the differences between various erasure codes for distributed storage systems, with a particular focus on Reed-Solomon codes and its variations. RS codes are storage and encoding efficient, but have costly decoding complexity [7]. Cauchy RS reduces this decoding complexity by transforming non-binary information to a binary representation that can undergo computation using XOR operations [10]. Rotated RS codes minimized the I/O required during degraded reads by using a parity comprised of multiple rows [15]. Hankel RS utilized Hankel matrices to remove the Vandermonde matrix inversion to greatly increase the efficiency of computation [12].

References

[1] U. D. of Commerce, "U.s. census bureau news- quarterly retail e-commerce sales 2nd quarter 2021."

[2] A. Fikes, "Storage architecture and challenges," 2010.

[3] P. Ruan, T. T. A. Dinh, D. Loghin, M. Zhang, G. Chen, Q. Lin, and B. C. Ooi, "Blockchains vs. distributed databases: Dichotomy and fusion," 2021.

[4] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *13th USENIX Conference on File and Storage Technologies (FAST 15)*, (Santa Clara, CA), pp. 213–226, USENIX Association, Feb. 2015.

[5] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An optimal scheme for tolerating double disk failures in raid architectures," *SIGARCH Comput. Archit. News*, vol. 22, p. 245–254, Apr. 1994.

[6] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[7] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[8] J. Plank and L. Xu, "Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications," in *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, pp. 173–180, 2006.

[9] J. S. Plank, "A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems," *Software – Practice & Experience*, vol. 27, pp. 995–1012, September 1997.

[10] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An xor-based erasure-resilient coding scheme," 10 1999.

[11] P. K. Dikang Gu and G. J. Chen, "Saving capacity with hdfs raid," June 2014.

[12] F. Mattoussi, V. Roca, and B. Sayadi, "Complexity comparison of the use of vandermonde versus hankel matrices to build systematic mds reed-solomon codes," in *2012 IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 344–348, 2012.

[13] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *USENIX ATC 2012 (Winner of the Best Paper Award)*, USENIX, June 2012. Winner of the Best Paper Award.

[14] 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 *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, (USA), p. 20, USENIX Association, 2012.

[15] "Rethinking erasure codes for cloud file systems: Minimizing i/o for recovery and degraded reads," in *10th USENIX Conference on File and Storage Technologies (FAST 12)*, (San Jose, CA), USENIX Association, Feb. 2012.