Systematic Erasure Codes with Optimal Repair Bandwidth and Storage

QING LIU and DAN FENG, Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China
HONG JIANG, Department of Computer Science and Engineering, University of Nebraska-Lincoln
YUCHONG HU and TIANFENG JIAO, Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China

Erasure codes are widely used in distributed storage systems to prevent data loss. Traditional codes suffer from a typical repair-bandwidth problem in which the amount of data required to reconstruct the lost data, referred to as the repair bandwidth, is often far more than the theoretical minimum. While many novel codes have been proposed in recent years to reduce the repair bandwidth, these codes either require extra storage and computation overhead or are only applicable to some special cases.

To address the weaknesses of the existing solutions to the repair-bandwidth problem, we propose Z Codes, a general family of codes capable of achieving the theoretical lower bound of repair bandwidth versus storage. To the best of our knowledge, the Z codes are the first general systematic erasure codes that jointly achieve optimal repair bandwidth and storage. Further, we generalize the Z codes to the GZ codes to gain the Maximum Distance Separable (MDS) property. Our evaluations of a real system indicate that Z/GZ and Reed-Solomon (RS) codes show approximately close encoding and repairing speeds, while GZ codes achieve over 37.5% response time reduction for repairing the same size of data, compared to the RS and Cauchy Reed-Solomon (CRS) codes.

CCS Concepts: • **Information systems → Storage recovery strategies**; • **Hardware → System-level fault tolerance**; • **Mathematics of computing → Coding theory**; • **Computer systems organization → Reliability**;

Additional Key Words and Phrases: Distributed storage system, erasure codes, failure tolerance, repair bandwidth

An earlier version of this work was presented at the 34th International Symposium on Reliable Distributed Systems (SRDS'15) (Liu et al. 2015b)

This work is supported in part by National Basic Research Program of China (973 Program) (2011CB302301); National Natural Science Foundation of China (61025008, 61232004, 61173043); National High Technology Research and Development Program of China (863 Program) (2013AA013203); National Key Technology R&D Program of China (2011BAH04B02); The Fundamental Research Funds for the Central Universities (2013TS043); Natural Science Foundation of Hubei Province (2015CFB192).

Authors' addresses: Q. Liu, D. Feng (corresponding author), Y. C. Hu, and T. F. Jiao, Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China, 1037 Luoyu Road, Wuhan 430074, China; emails: {qing, dfeng, yuchonghu, tfjiao}@hust.edu.cn; H. Jiang, the bulk of the research was done at University of Nebraska-Lincoln, and he is currently affiliated with Department of Computer Science and Engineering, University of Texas at Arlington; email: jiang@cse.unl.edu.

https://doi.org/10.1145/3109479

ACM Reference format:
Qing Liu, Dan Feng, Hong Jiang, Yuchong Hu, and Tianfeng Jiao. 2017. Systematic Erasure Codes with Optimal Repair Bandwidth and Storage. *ACM Trans. Storage* 13, 3, Article 26 (September 2017), 27 pages.
https://doi.org/10.1145/3109479

1 INTRODUCTION

Erasure codes are widely used in distributed storage systems to recover from data loss in the event of server breakdown. These codes incorporate data redundancy in a space-efficient manner to tolerate data loss by reconstructing the lost data and are *systematic* in that the native data are kept unchanged after encoding and can be accessed without decoding. An (m, k) code with the *Maximum Distance Separable* (MDS) encodes k data fractions, which are equally divided from the native data, into $m + k$ fractions, and any k fractions can retain the native data. Typical examples of these codes include Reed-Solomon (RS) codes (Reed and Solomon 1960) and Cauchy Reed-Solomon (CRS) codes (Blömer et al. 1995).

However, such traditional erasure codes face a known repair-bandwidth problem (Dimakis et al. 2010) that becomes increasingly more important in a distributed environment where bandwidth is typically expensive in terms of both performance and power consumption. That is, in a storage system of data size M with k data nodes and m parity (i.e., redundant) nodes that are interconnected by a network of limited bandwidth, each node stores data of size $\frac{M}{k}$. The repair of one node's failure requires disk access or network bandwidth of size M, which is k times the size of the lost data ($\frac{M}{k}$). A given erasure code with a larger k in a storage system means higher storage efficiency, but accomplishing with more disk I/Os and repair network traffic.

Disk I/O and network bandwidth are valuable resources in distributed storage systems and the repair-bandwidth problem has a negative performance impact on both of them. First, a repair causes excessive disk I/Os and traffic flow over the network, which can severely interfere with normal services of the system; second, a repair requires a long time to reconstruct the lost data, which indirectly weakens the reliability of the system by exposing the system to a long window of vulnerability in which additional node failures can lead to unrecoverable data loss. A former study on a production cluster of 3,000 nodes in Facebook showed that repair network traffic for recovering RS-based failures would exhaust the cluster network links, if no preventive measures are taken (Sathiamoorthy et al. 2013). Another study, also on a Facebook's cluster, revealed that recovering a single lost data block is the most common case, since about 98% failures are the single failure in real scenarios (Rashmi et al. 2013). In this article, we define repair bandwidth as the amount of data accessed by the disk I/O and transferred over the network for repairing a single failure.

Storage capacity is also an expensive resource in terms of the maintenance cost and power consumption for the storage systems. The minimum storage of a node for an (m, k) code is $\frac{M}{k}$, so any k nodes can retain the native data of size M if the code keeps the MDS property. However, Dimakis et al. (2010) pointed out that repair bandwidth can be reduced by increasing the storage capacity of each node or making more nodes participate in the repair process (also known as enlarging the *repair locality*, which is the number of nodes to help to repair). If no extra data are stored with the repair locality being unchanged, then no repair bandwidth can be saved. Furthermore, the theoretical minimum storage and the minimum repair bandwidth cannot be achieved at the same time, and there is a lower-bound trade-off curve between the two (Dimakis et al. 2010), as plotted in Figure 1. Although codes with the minimum storage cannot simultaneously achieve the minimum repair bandwidth, their theoretical repair bandwidth lower bound, which is called *optimal repair bandwidth* (Rashmi et al. 2015), can be achieved by increasing repair locality to $m + k - 1$. Thus,

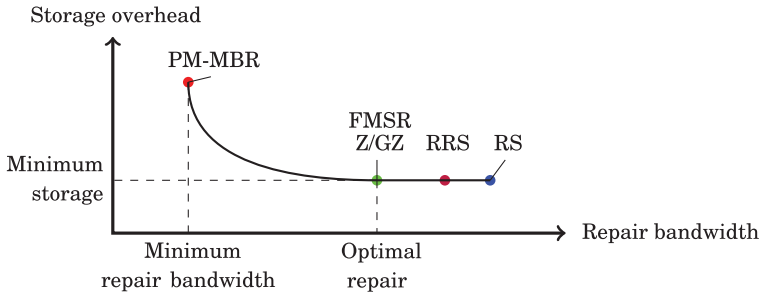


Fig. 1. Theoretical lower-bound trade-off curve of storage overhead and repair bandwidth.

the optimal repair bandwidth for the minimum-storage codes can be calculated as

$$(m + k - 1)M/(mk). \tag{1}$$

Recently, many novel repair-bandwidth-efficient codes have been proposed to reduce the repair bandwidth, but at the expenses of (1) extra storage capacity, (2) additional computation overhead, (3) lacking the MDS property, or (4) being applicable only to some special cases. Both Simple Regenerating Codes (SRC) (Papailiopoulos et al. 2012) and Local Reconstruction Codes (LRC) (Huang et al. 2012) can reduce the repair bandwidth and repair locality at the same time, but they consume additional storage resources to store the extra parity information, and the LRC codes also lose the MDS property. The Functional Minimum Storage Regenerating (FMSR) codes are not systematic and only store parity information after encoding, thereby resulting in a high computation cost (Chen et al. 2014). The Rotate Reed-Solomon (RRS) codes (Khan et al. 2012) also require additional computation with limited repair bandwidth being saved. The Zigzag codes (Tamo et al. 2013) can achieve the optimal repair bandwidth for repairing the single failure of a data node. However, they have no general construction and the finite fields where some Zigzag codes (Tamo et al. 2013) are constructed are not binary fields. The product-matrix based regenerating codes (Rashmi et al. 2013, 2015) exist only when the *code rate* (the ratio of the data size and size of total data after encoding) is not greater than $\frac{1}{2}$, namely, $m \geq k$.

To address the above weaknesses in the existing codes, we present in this article a family of novel erasure codes, called the Z codes. The Z codes not only can achieve the theoretical optimal repair bandwidth under the minimum storage for a single data node's failure but also have the following desirable properties that make them suitable for distributed storage systems. (1) *The minimum storage property*: the Z codes consume exactly the same storage capacity as the RS and CRS codes. (2) *Low computation overhead*: since the Z codes are XOR codes, all coding operations can be performed by fast bitwise XOR operations; (3) *The systematic property*: the native data remain unchanged after encoding, allowing accesses to the native data without decoding; (4) *Generality*: the code rate of the Z codes can be arbitrarily high with a flexible parameter set (m, k), meaning that the Z codes can be constructed for arbitrary numbers of data nodes and parity nodes as long as $m \geq 2$ and $k \geq 2$. In Table 1, we compare the storage cost and the repair bandwidth for several repair-bandwidth-efficient codes with $m = 2$ and $k = 10$. From the table, we can see that the Z codes have the least storage cost, the same as the RS codes and the FMSR codes, as well as the least repair bandwidth, the same as the FMSR codes. We call our codes the Z codes, because they achieve the ultimate desirable features analogous to the last letter "Z" in the alphabet. Further, we generalize the Z codes to the GZ codes by constructing the former in the Galois Field GF(2^w) ($w > 1$), which enables the GZ codes to obtain the MDS property practically.

Table 1. Storage Cost for Storing 1 TB Data and Repair Bandwidth for Reconstructing 1 TB Lost Data for Codes with $m = 2$ and $k = 10$ (l and f Are Configuration Parameters for SRC and LRC)

Codes	Storage Cost	Repair Bandwidth	Computation Overhead
3-Replicas	3 TB	1 TB	Very low
RS	1.2 TB	10 TB	High
SRC ($f = 5$)	1.44 TB	5 TB	High
LRC ($l = 2$)	1.3 TB	5 TB	High
RRS	1.2 TB	7.5 TB	High
FMSR	1.2 TB	4.5 TB	Very High
Z	1.2 TB	4.5 TB	Low

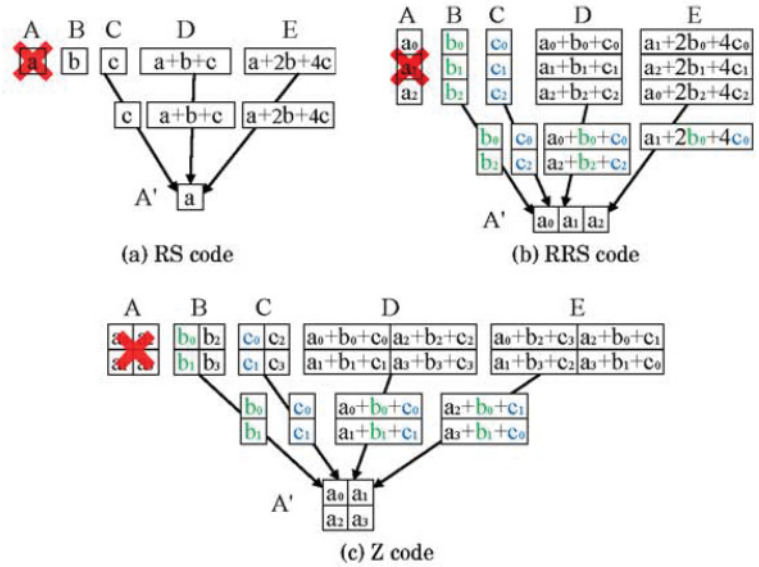


Fig. 2. An example of how the three codes, RS, RRS, and Z, of $k = 3$ and $m = 2$ repair the lost data in node A. Repair bandwidth of node A decreases as the number of blocks in each node (r) increases. Note that all three codes have the minimum storage, so if parity nodes is less than two ($m < 2$), no repair bandwidth can be saved by allowing more nodes to participate the repair process.

The Z codes can achieve the optimal repair bandwidth based on the principle that is *dividing native data of each node into more blocks, more repair bandwidth can be saved by means of selecting some specific blocks from the live nodes to help to repair*. We illustrate this principle with the following example in Figure 2, where we compare repairs of three codes for reconstructing data blocks in node A. r represents the number of block in each node: the $(2, 3)$-RS code with $r = 1$ (i.e., one block in each node), the $(2, 3)$-RRS code with $r = 3$ and the $(2, 3)$-Z code with $r = 4$. The first two codes are over the Galois field, resulting in more computation overheads than the Z code for encoding and repairing. Repair bandwidths of these three codes, normalized to the size of the native data, are 1 (RS), 0.778 (RRS), and 0.667 (Z), respectively.

Table 2. Notations and Their Definitions

Notations	Meaning
k	Number of data chunks in a stripe
m	Number of parity chunks in a stripe
n	Number of chunks in a stripe ($n = m + k$)
r	Number of blocks in a chunk
f	Index number of the faulty data chunk
d	Repair locality, Number of nodes participated to repair
D_i	The i-th data chunk ($0 \leq i < k$)
C_i	The i-th parity chunk ($0 \leq i < m$)
$D_{i,j}$	The j-th data block of data chunk D_i ($0 \leq j < r$)
$C_{i,j}$	The j-th parity block of parity chunk C_i ($0 \leq j < r$)
P	The generator matrix of an (m, k)-Z code
Q	The generator matrix of an $(m, k+1)$-Z code
$P_{i,j}$	submatrix of P ($0 \leq i < k$, $0 \leq j < m$)
P_i	The i-th block row of P, $P_i = [P_{i,0}, \ldots, P_{i,k-1}]$
M	Native data size (total size of all data chunks)

Our main contributions are summarized as follows:

—We propose the Z codes, a family of erasure codes with the optimal repair bandwidth under the minimum storage.
—We generalize the Z codes to the GZ codes to gain MDS property by sacrificing the property of XOR codes.
—We inductively prove the attainment by the Z/GZ codes the optimal repair bandwidth under the minimum storage for a single data node failure.
—In a real distributed storage system, we comprehensively evaluate the performance of the Z codes, in terms of the coding performance, disk access, and the response time.

The rest of the article is organized as follows. Section 2 introduces some basic background of all-purpose erasure codes and some repair-bandwidth-efficient codes; we present the Z codes in Section 3, including code construction, encoding, repairing and decoding; Section 4 introduces the GZ codes and how they attain the MDS property; Section 5 proves the attainment by the Z/GZ codes the theoretical optimal repair bandwidth under the minimum storage; we analyse the properties of Z/GZ codes in Section 6; Section 7 evaluates the performance of the Z/GZ codes in several key metrics; Section 8 proposes some open issues of Z/GZ codes with remarks on the future work; we conclude the article in Section 9.

2 BACKGROUND AND MOTIVATION

In this section, we introduce the necessary background on the erasure-coded distributed storage systems to motivate and facilitate the description of our Z codes. To facilitate these detailed description, we list some notations of erasure codes and their definitions in Table 2.

2.1 Erasure Coding in Distributed Storage Systems

To simplify our discussion, we focus on a stripe over n storage nodes as showed in Figure 3. A *stripe* (Plank 2005) is the minimum unit for encoding, decoding, and repairing, and it contains n chunks over n storage nodes, one chunk in each node. If the code is systematic, then n chunks can be partitioned into k data chunks equally divided from the native data and m parity chunks that

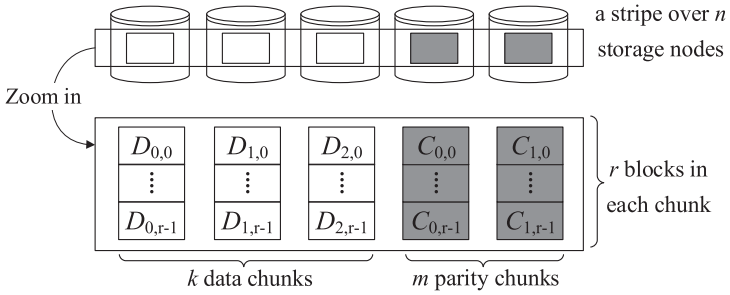


Fig. 3. A stripe of 5 chunks across 5 storage nodes; one chunk in each node is a set of r blocks.

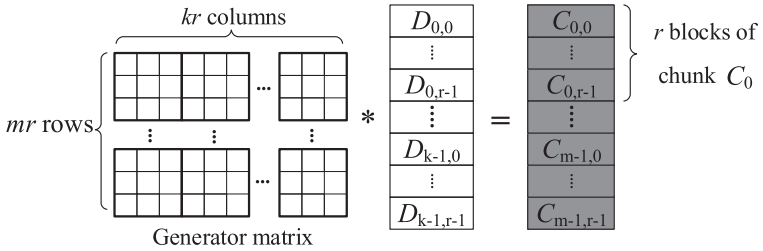


Fig. 4. Encoding data blocks into parity blocks through left multiplying all data blocks by the generator matrix.

store the redundant information of all data chunks. We let D_i/C_i denote the ith data/parity chunk. Each chunk is a set of r contiguously stored blocks of equal size. When $r = 1$, a chunk contains only one block, analogous to the RS codes. When $r > 1$, a chunk is a set of r blocks, analogous to the CRS codes, and $D_{i,j}/C_{i,j}$ denotes the jth data/parity block in the ith data/parity chunk.

2.2 Encoding with the Generator Matrix

Each linear code can be uniquely represented by a generator matrix, which defines how data blocks in a stripe are encoded into parity blocks (Plank 2005). Encoding is the process of left multiplying all data blocks by the generator matrix, as pictured in Figure 4. Since there are kr data blocks and mr parity blocks in a stripe, a generator matrix of the code has mr rows and kr columns correspondingly. Each row vector in the generator matrix corresponds to a parity block, which can be viewed as a linear combination of all data blocks with entries in the row vector as coefficients.

Entries of the generator matrix are symbols over the Galois field GF(2^w), and their values are between 0 and $2^w - 1$. When $w = 1$, the code is an XOR code with each entry being either 0 or 1. A block can be seen as an array of w-bit symbols, so the product of an entry and the data block can be divided into several multiplications of two symbols over the field GF(2^w).

2.3 State-of-the-art Repair-Bandwidth-Efficient Codes

The Regenerating codes (Dimakis et al. 2010) are a class of codes that can theoretically achieve the theoretical lower bound trade-off curve in Figure 1, and they can be realized by random linear codes (Dimakis et al. 2010; Liu et al. 2015a). The FMSR codes and the product-matrix-based regenerating codes are two typical regenerating codes that achieve the theoretical minimum with respect to storage overhead or repair bandwidth. The FMSR codes (Chen et al. 2014) achieve the optimal repair bandwidth under the minimum storage for any single failure. However, there are

two issues to be considered before applying them in distributed storage systems. First, repairs of the FMSR codes are functional, so the lost data are not really restored, instead some alternative data are regenerated to maintain the data redundancy. Second, the FMSR codes are not systematic, which means that the native data are altered after encoding and a read request to the data will trigger a decoding operation that involves extra data access and computation (Chen et al. 2014).

The Product-Matrix-MSR codes (PM-MSR) (Rashmi et al. 2013) and the Product-Matrix-MBR (PM-MBR) (Rashmi et al. 2015) are two types of product-matrix-based regenerating codes, which separately achieve the minimum storage and the minimum repair bandwidth of the lower-bound trade-off in Figure 1. The PM-MSR codes can achieve the optimal repair bandwidth under the minimum storage, but they are non-systematic codes, and their code rate is no more than $\frac{1}{2}$. Another trade-off that the PM-MSR codes make for the optimal repair bandwidth is excess disk I/Os, because each node helping to repair has to access more size of data from the local storage than it transfers (Rashmi et al. 2015). The PM-MBR codes are systematic and they achieve the minimum repair bandwidth for not only repair network traffic but also disk I/Os. Nevertheless, the PM-MBR codes have to store some extra data in each node and their code rates are also no more than $\frac{1}{2}$.

Besides the Regenerating codes, other repair-bandwidth-efficient codes have been proposed recently, such as the LRC, SRC, and DRESS codes (Pawar et al. 2011). Similar to the PM-MBR codes, these codes trade the storage resources for repair bandwidth reduction. The SRC codes store extra data in each storage node (Papailiopoulos et al. 2012) and the LRC codes (Huang et al. 2012) introduce additional local parity blocks that confines the repair for a single failure locally. The DRESS codes are based on replication in which at least two replicas are kept for each block, so the code rate is always less than $\frac{1}{2}$.

2.4 Code Construction and Comparison

We summarize the RS, CRS, and some state-of-the-art repair-bandwidth-efficient erasure codes in Table 3 in aspects of storage cost, repair efficiency, computation complexity, and so on. The generator matrices of the RS and CRS codes are based on the known Vandermonde matrix or Cauchy Matrix (Plank and Ding 2005), and those of Z/GZ codes are based on a recursive construction of permutation matrix, which we discuss in detail in Section 3. While the generator matrices of the PM-MSR and PM-MBR codes are designed by elaborately concatenating symmetric matrices, zero matrix, and other matrices (Rashmi et al. 2013). The FMSR codes perform encoding and repairing by random linear combinations, so their generator matrices are randomly generated and satisfy some certain conditions. The SRC and LRC codes are hybrid coding schemes, which combine two or more specific erasure codes for a trade-off between the storage capacity and the repair bandwidth. The LRC codes apply Taking the LRC codes, for example, they apply one global coding scheme for all k data chunks and another local coding scheme for each group of l data chunks ($l(l < k)$ represents the number of data chunks in a group), so a single failure of data chunk inside the group can be repaired locally. The cost is the extra storage for local parity block generated by the local coding scheme.

Various erasure codes achieve trade-offs among storage overhead, computation, repair cost, repair locality, and so on. In the same way, all listed repair-bandwidth-efficient codes trade some beneficial properties for the repair bandwidth reduction. For example, the SRC, LRC, and PM-MBR codes lose the minimum storage, and they either store extra data on each node or require additional local parity blocks. The PM-MSR codes and the FMSR codes are not systematic, so native data are removed after encoding. In addition, the PM-MSR codes are I/O inefficient and make a limit on the parameters of m and k. The Z codes make their own trade-offs, such as the sub-packetization that divides data of each node into a great number of pieces, and we discuss them in Section 8.

Table 3. Properties of Some Up-to-Date Repair-Bandwidth-Efficient Codes, Compared with RS and CRS Codes

Codes	Storage Capacity	Repair Bandwidth	Repair Locality	Code Rate	MDS	XOR	Sys.†	I/O Efficient	Constraint Condition
RS	$n\frac{M}{k}$	M	k	$\frac{k}{n}$	√	×	√	√	
CRS	$n\frac{M}{k}$	M	k	$\frac{k}{n}$	√	√	√	√	
opt-CRS	$n\frac{M}{k}$	$< M$	$> k$	$\frac{k}{n}$	√	√	√	√	$m \geq 2$
RRS	$n\frac{M}{k}$	$\frac{(k+\lceil \frac{k}{m} \rceil)}{2}\frac{M}{k}$	$k+1$	$\frac{k}{n}$	√	×	√	√	$r = 2, 4, 8; m = 2, 3$
Z	$n\frac{M}{k}$	$\frac{(n-1)}{m}\frac{M}{k}$	$n-1$	$\frac{k}{n}$	×	√	√	√	$m \geq 2$
GZ	$n\frac{M}{k}$	$\frac{(n-1)}{m}\frac{M}{k}$	$n-1$	$\frac{k}{n}$	√	×	√	√	$m \geq 2$
PM-MSR	$n\frac{M}{k}$	$\geq \frac{(n-1)}{m}\frac{M}{k}$	$\geq 2k-2$	$\frac{k}{n} \leq \frac{1}{2}$	√	×	×	×	$m \geq 2; n \geq 2k-1$
PM-MBR	$\frac{2(n-1)}{2n-k-1}n\frac{M}{k}$	$\frac{2(n-1)}{2n-k-1}\frac{M}{k}$	$n-1$	$\frac{2n-k-1}{2(n-1)}\frac{k}{n}$	√	×	√	√	
FMSR	$n\frac{M}{k}$	$\frac{(n-1)}{m}\frac{M}{k}$	$n-1$	$\frac{k}{n}$	√	×	×	√	$m = 2$
SRC	$\frac{f+1}{f}n\frac{M}{k}$	$(f+1)\frac{M}{k}$	$2f$	$\frac{f}{f+1}\frac{k}{n}$	√	×	√	√	$f \geq 2; m \geq 2$
LRC‡	$(l+g+k)\frac{M}{k}$	$l\frac{M}{k}$	$l < k$	$\frac{k}{l+g+k}$	×	×	√	√	$l \mid k$

† Systematic. If a code keeps the native data after encoding, then it is systematic.
‡ l and g indicates the numbers of local parity nodes and global parity nodes, respectively, and $l|k$ means that k can be equally divided by l.

Analogous to other minimum-storage Regenerating codes (such as the FMSR codes) that can achieve the optimal repair bandwidth, two factors guarantee the optimal repair bandwidth for the Z codes. First, repair locality reaches its maximum and it means all $n-1$ live nodes participate in the repairing; second, each of the $n-1$ remaining nodes only needs to selectively access and transfer blocks at a rate of $\frac{1}{m}$. Therefore, the amount of data required in each node is $\frac{1}{m}\frac{M}{k}$ and the repair bandwidth is $(n-1)\frac{M}{mk}$, exactly the same as the optimal repair bandwidth in Equation (1).

3 Z CODES

In this section, we introduce the construction of the Z codes inductively and present their encoding, repairing, and decoding.

3.1 Code Construction

As we mentioned in Subsection 2.2, each erasure code has its own generator matrix for encoding, repairing, and decoding. We introduce the generator matrix construction of the Z code in two steps: First, we give the construction of Z codes on the condition of two data nodes ($(m, 2)$-Z code); second, we present an inductive method that deduces the generator matrix of the $(m, k+1)$-Z code from the generator matrix of the (m, k)-Z code. Through these two steps, we can generate generator matrices of all Z codes with arbitrary with arbitrary parameters of m and k.

(1) *Z Codes with $k = 2$*: The generator matrix of the $(m, 2)$-Z code is an $m \times 2$ block matrix (denoted by P), and each block (denoted by $P_{i,j}$) is an $m \times m$ bit matrix, as showed in Equation (2),

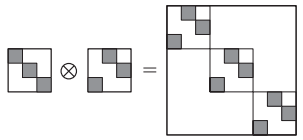


Fig. 5. A tensor product of I_3 and $I_3^{2+}(I_3 \otimes I_3^{2+})$. Each entry 1 in the I_3 is replaced by an I_3^{2+} and each entry 0 is replaced by a 3×3 zero matrix. Note that the shaded boxes in the generator matrix represent 1s and others are 0s and we follow the same plotting convention in the rest of the article.

$$P = \begin{bmatrix} P_{0,0} & P_{0,1} \\ P_{1,0} & P_{1,1} \\ \vdots & \vdots \\ P_{m-1,0} & P_{m-1,1} \end{bmatrix}_{m\times 2} = \begin{bmatrix} I_m & I_m \\ I_m & I_m^{+} \\ \vdots & \vdots \\ I_m & I_m^{(m-1)+} \end{bmatrix}_{m\times 2}, \tag{2}$$

where I_m is an $m \times m$ identity matrix, and $I_m^{i+}(i < m)$ is an $m \times m$ permutation matrix representing i steps' left cyclic shift from I_m. Therefore, I_m equals to I_m^{0+}. We call I_m^{i+} *cell matrix*, since it is the fundamental submatrix of P. The number of different *cell matrices* is m. For example, when $m = 3$, all *cell matrices* are as follows:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} I_3^{1+} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} I_3^{2+} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

(2) *Z Codes with an arbitrary k*: We first present an inductive construction method that generates the generator matrix of the $(m, k+1)$-Z code from the generator matrix of the (m, k)-Z code. Let P and Q denote generator matrices of the (m, k)-Z code and the $(m, k+1)$-Z code separately, a two-step method of constructing Q from P is as follows:

(1) Replicate the last block column of P (that is $[P_{0,k-1}\ P_{1,k-1} \cdots P_{m-1,k-1}]^\intercal$) and tile it onto the P along the horizontal dimension. The result is denoted as an intermediate matrix T:

$$T = \begin{bmatrix} P_{0,0} & \cdots & P_{0,k-1} & P_{0,k-1} \\ P_{1,0} & \cdots & P_{1,k-1} & P_{1,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m-1,0} & \cdots & P_{m-1,k-1} & P_{m-1,k-1} \end{bmatrix}_{m\times(k+1)}.$$

(2) Construct Q from T as follows, where each submatrix $Q_{i,j}$ in Q is a tensor product of a submatrix $T_{i,j}$ in T and a cell matrix I_m^{j+}:

$$Q = \begin{bmatrix} Q_{0,0} & \cdots & Q_{0,k-1} & Q_{0,k} \\ Q_{1,0} & \cdots & Q_{1,k-1} & Q_{1,k} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{m-1,0} & \cdots & Q_{m-1,k-1} & Q_{m-1,k} \end{bmatrix}_{m\times(k+1)}$$

$$\text{and} \begin{cases} Q_{i,j} = T_{i,j} \otimes I_m,\ i = 0, \ldots, m-1, j = 0, \ldots, k-1 \\ Q_{i,k} = T_{i,k} \otimes I_m^{j+},\ i = 0, \ldots, k-1, \end{cases}$$

where $\otimes$ denotes the tensor product. Note that the tensor product of matrices is also called the Kronecker product, and an example of the tensor product is given in Figure 5. From P to T, blocks of the last column in P are replicated. From T to Q, the number of blocks remains the same, but

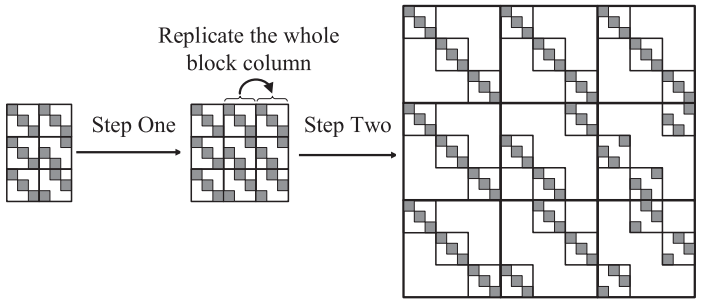
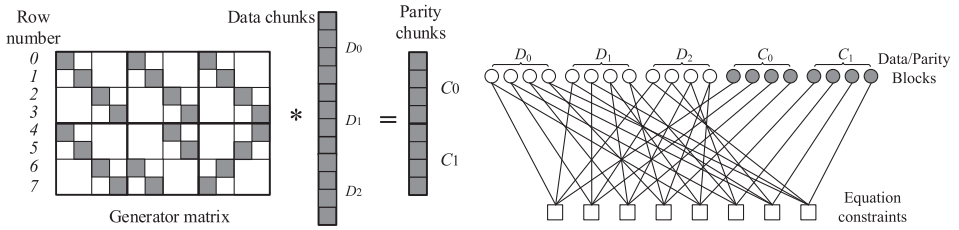


Fig. 6. Constructing the generator matrix of the (3, 3)-Z code from the generator matrix of the (3, 2)-Z code. The last block column is duplicated in the first step and each submatrix is replaced by the tensor product of the submatrix and a specific *cell matrix* in the second step.



(a) Encoding with generator matrix. (b) Tanner graph

Fig. 7. Generator matrix encoding of (2, 3)-Z code (a) and its Tanner graph expression (b).

each block in Q is a tensor product of the block with the same position in T and a cell matrix. Since we have already introduced the generator matrix's construction of the $(m, 2)$-Z code, we can construct the generator matrix for any (m, k)-Z code inductively. An example of constructing the generator matrix of the (3, 3)-Z code from the generator matrix of the (3, 2)-Z code is given in Figure 7. For an (m, k)-Z code, the number of blocks in a chunk (r) is a function of m and k, $r = m^{k-1}$. The generator matrix P actually has a general expression as Equation (3), which can be deduced from the above inductive construction:

$$P = [P_{i,j}]_{m\times k},\ P_{i,j} = (I_m^{i+})^{\otimes j} \otimes (I_m)^{\otimes(k-1-j)} \text{ with } 0 \le i < m,\ 0 \le j < k, \tag{3}$$

where i, j are the block row index number and the block column index number, respectively, and the power $(I_m^{i+})^{\otimes j}$ of matrix I_m^{i+} denotes the tensor product of j copies of I_m^{i+}.

In all, the generator matrix of an (m, k)-Z code is a $m \times k$ block matrix, with each block $P_{i,j}$ being a submatrix of a tensor product of $k - 1$ cell matrices. Since cell matrices are all permutation matrices, so $P_{i,j}$ is also a permutation matrix. Thus, the numbers of 1s in each row and each column of generator matrix P are k and m, respectively.

3.2 Encoding

Encoding is the procedure of generating m parity chunks by left multiplying data blocks by the generator matrix P, exactly the same as the encoding process of other matrix-based codes (Plank 2005). We illustrate an encoding example of the (2, 3)-Z code in Figure 7(a) and its Tanner-graph expression in Figure 7(b). The Tanner graph is a bipartite graph that is commonly used for representing the LDPC codes (Tanner 1981). There are two distinctive sets of vertices in a Tanner graph, namely, vertices representing data and parity blocks and those representing equation constraints,

which are connected by edges. An equation constraint is connected to some blocks by edges in the graph, whose summation (XOR) is equal to 0. *Each equation constraint in a Tanner graph corresponds to a row vector in the generator matrix of the same code.* For example, that an equation constraint connects to nodes of $D_{0,0}$, $D_{1,2}$, $D_{2,3}$, and $C_{1,0}$ represents the equation:

$$D_{0,0} \oplus D_{1,2} \oplus D_{2,3} \oplus C_{1,0} = 0,$$

where $\oplus$ denotes XOR. This equation constraint corresponds to the encoding process of generating parity block $C_{1,0}$, which is the result of left multiplying data blocks with the 4-th row vector of the generator matrix.

3.3 Repairing

We focus on the case of a single data node's failure,[1] which in the article refers in particular to the failure of a data node that stores the data information, excluding the failure of a parity node. We make this trade-off in the consideration of different reconstruction priority of a parity chunk and a data chunk. If a parity chunk is lost, then data chunks of the same stripe are still readable, which allows the reconstruction of the parity chunk to be done at a later time or in the background. This is usually not allowed for a lost data chunk, because once a data chunk is lost, a read request to the lost data causes a degraded read in which the lost data must be reconstructed as quickly as possible to satisfy the read request.

With the help of the Tanner graph of the Z code, we introduce a four-step method to determine $\frac{r}{m}$ blocks of each chunk in a node for repairing a faulty data chunk D_f as follows. (1) Select any one block of D_f and any one of the equation constraints the block is connected with, then remove other equation constraints the block is also connected with; (2) for each removed equation constraint, remove all blocks it is connected with, except for the data blocks from D_f; (3) for each removed block, remove all equation constraints it is connected with; (4) repeat (2) and (3) until no more blocks and equation constraints can be removed. The remaining blocks are the blocks required to repair the faulty node, with the remaining equation constraints as equations to solve all data blocks of D_f.

Taking the $(2, 3)$-Z code as an example, we plot the $(2, 3)$-Z code's optimal repairs for the failure of each data node in Figure 8. For a better view, we faded the colors of the data/parity blocks and equation constraints that would not be used for repairing. To repair the faulty data chunk D_f $(f = 0, 1, 2)$, each remaining chunk contributes 2 blocks, so the repair bandwidth is 8 blocks. These 8 selected blocks and 4 lost blocks ($D_{f,0}$, $D_{f,1}$, $D_{f,2}$, and $D_{f,3}$) are associated with 4 equation constraints, which are only associated with these 8 selected blocks and 4 lost blocks. Thus, D_f can be repaired, since each lost data block is reconstructed by solving an equation constraint with the lost block as the only unknown argument. Compared with the RS or CRS codes that require all blocks from each of $k = 3$ chunks, the $(2, 3)$-Z code saves the repair bandwidth by an amount of $3 \times 4 - 8 = 4$ blocks.

3.4 Decoding

Since the Z codes are systematic, meaning that they retain the native data after encoding, data can be directly obtained without decoding as long as no data chunks are lost. If any data chunk in a stripe is lost, then the Z codes cannot recover the native data by downloading any k remaining chunks like other MDS codes. The best practice for Z codes is to selectively choose some blocks from more than k nodes, and these blocks' row vectors in the generator matrix form a full-rank

[1]We use the phrases "failure of a data node" and "failure of a data chunk" interchangeably, since "failure of a data node" causes a subsequent repair by rebuilding all chunks of the failed node.

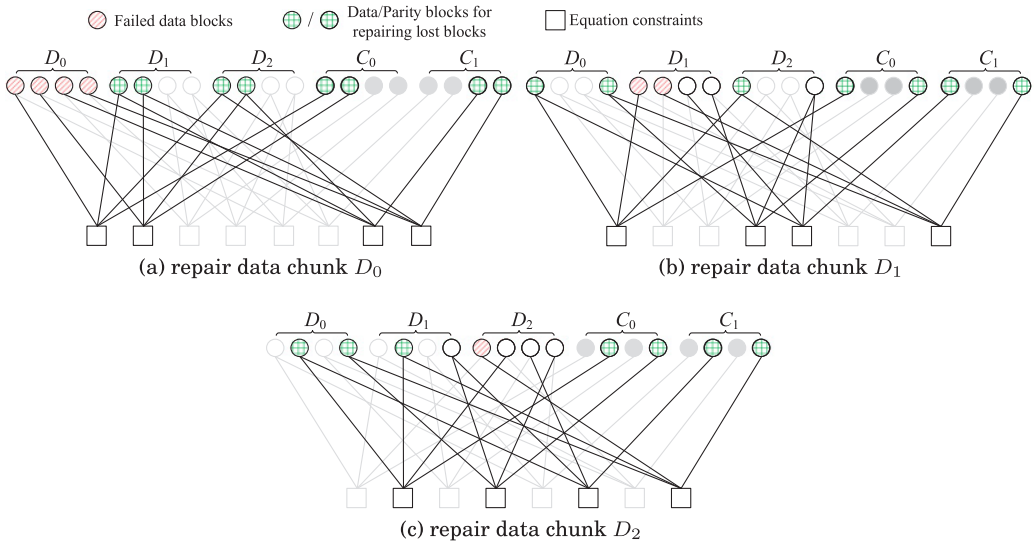


Fig. 8. Tanner graphs representing repairs of D_0, D_1, and D_2 with the optimal repair bandwidth for the $(2, 3)$-Z code.

square matrix. Then the data can be reconstructed by calculating the inversion of the square matrix and those blocks as the traditional decoding method (Plank 2005). More thoroughly, we will introduce a method to generalize the Z codes into Galois-field-based Z (GZ) Codes in the next section, which enables the GZ codes to attain the MDS property, so the native data can be decoded from any k data/parity chunks.

4 GALOIS-FIELD-BASED Z CODES

The Z codes we presented above are XOR codes that are constructed over a Galois field GF(2). In this section, we generalize Z codes to Galois-field-based Z (GZ) codes by constructing them over a bigger Galois field GF(2^w) with $w > 1$, in an effort to gain the MDS property. Essentially, the layout of non-zero entries in the generator matrix decides the associations between blocks and equation constraints in the Tanner graph, where such associations ensure the optimal repair bandwidth of the Z codes. Therefore, if we substitute 1s in the generator matrix with non-zero symbols over a bigger Galois field, the MDS property will likely be attained with the optimal repair bandwidth maintained.

Let PG represent the (m, k)-GZ code's generator matrix, which is also a block matrix, defined by Equation (4):

$$PG = [l_{i,j} \cdot P_{i,j}]_{m \times k}, \quad 0 \le i < m, \ 0 \le j < k. \tag{4}$$

Recall $P_{i,j}$ is the submatrix of the generator matrix of the (m, k)-Z code P. Each submatrix $PG_{i,j}$ is the result of replacing each entry 1 in $P_{i,j}$ with $l_{i,j}$. For different submatrices of PG, there are different multiplicative scalar $l_{i,j}$ and all $l_{i,j}$s form an $m \times k$ matrix, written as L. Given the determinate construction of a (m, k)-Z code, the generator matrix of the (m, k)-GZ code is more flexible. Developers can choose their own matrix L, as long as L makes the GZ code possess the MDS property. Existing MDS matrices, such as Vandermonde matrices (Plank and Ding 2005) and Cauchy matrices, can be used to generate L. In the remainder of the article, we generate the matrix L with the standard Vandermonde matrix for all GZ codes.

Table 4. Verifying the MDS Property of the GZ Codes Over GF(2^w). (√ Signifies that the MDS Property Is Satisfied, – Means that the MDS Property Is not Checked Because of r's Size Limit)

	k (w = 8/16/32)											
	2	3	4	5	6	7	8	9	10	11	12	13
$m = 2$	√	√	√	√	√	√	√	√	√	√	√	√
$m = 3$	√	√	√	√	√	√	√	√	–	–	–	–
$m = 4$	√	√	√	√	√	√	–	–	–	–	–	–

Except for the XOR encoding property, the GZ codes retain the desirable properties of the Z codes, including the optimal repair bandwidth under the minimum storage. This is because generator matrices of the Z and GZ codes with the same m and k have the same layout, as well as the same Tanner graphs, which guarantee the optimal repair as we will prove in the next section.

Over some commonly used Galois fields (GF(2^8), GF(2^{16}), and GF(2^{32})), we conduct some experiments to verify the MDS property of the GZ codes by checking all combinations of m concurrent failures and making sure that all combinational failures are tolerated. The inspective results are given in Table 4, and we limit the size of r to 7,000, to avoid excessively large generator matrices. The results show that all GZ codes fulfilled the MDS property in all cases. Both GF(2^8) and GF(2^{16}) are good choices in practice, and these fields are big enough to construct the GZ codes with high coding speed, according to the previous research (Plank et al. 2013b).

5 PROOF OF THE OPTIMAL REPAIR BANDWIDTH

We prove the Z/GZ codes' attainment of optimal repair bandwidth under minimum storage in this section, and our proof has two parts. In the first part, we propose a sufficient and necessary condition that guarantees that a data node's failure can be repaired with optimal repair bandwidth; we then inductively demonstrate that each data node's failure can be optimally repaired by Z codes in the second part.

Although we only prove the optimal repair bandwidth of Z codes, the (m, k)-GZ code can optimally repair a data node's failure as the (m, k)-Z code, due to the similar code constructions and identical Tanner graphs.

To facilitate the proof, we first introduce a sufficient and necessary condition of a general erasure code with $r > 1$ for optimally repairing the failure of a single data node in Theorem 5.1. If a code can repair the failure of the fth data node with the optimal repair bandwidth, then we say that the code with generator matrix P can optimally repair the failure of the fth data node, written as $P \xrightarrow{opt} f$. We select $\frac{r}{m}$ row vectors from each P_i to help to repair, except the failed node, and all row vectors form a $r \times kr$ matrix, which is named *constraint matrix*, since each row vector in the constraint matrix represents a constraint equation of the Tanner graph.

Theorem 5.1. *Let P be a generator matrix of an (m, k) code with $r > 1$ as follows:*

$$P = \begin{bmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,k-1} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m-1,0} & P_{m-1,1} & \cdots & P_{m-1,k-1} \end{bmatrix}_{m \times k}.$$

Then $P \xrightarrow{opt} f$ if and only if there exists a constraint matrix S satisfying the following Optimally Repairable Property (ORP) for the f: equally dividing S into k $r \times r$ blocks by column ($S_i, i = 0, \ldots, k-1$), S_f is a non-singular matrix, while other submatrix $S_i (i \neq f)$ has only $\frac{r}{m}$ non-zero columns.

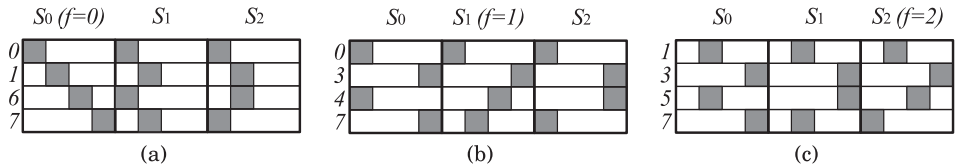


Fig. 9. Constraint matrices satisfying the ORP for repairing D_0 (a), D_1 (b) and D_2 (c). Numbers on the left are row indexes of the row vectors in the generator matrix P.

Proof. Assume that there is a constraint matrix S satisfying the ORP. Each row vector of S corresponds to a parity block, and these r parity blocks are denoted by C_s. Thus, there are r equations about C_s as follows:

$$D_0S_0 \oplus D_1S_1 \oplus \cdots \oplus D_{k-1}S_{k-1} = C_s.$$

Since S_f is a non-singular matrix, there are r linearly independent equations of all data blocks of D_f. Because there are $\frac{r}{m}$ non-zero columns in each S_i $(i \neq f)$, only $\frac{r}{m}$ data blocks from each data chunk D_i $(i \neq f)$ are combined in all equations. Therefore, we can reconstruct data blocks of D_f by solving these r equations, and each chunk of a stripe contributes $\frac{r}{m}$ blocks. Thus, $P \xrightarrow{opt} f$.

Assume that $P \xrightarrow{opt} f$, then a set of data blocks and parity blocks (denoted as C'_s), $\frac{r}{m}$ blocks from each chunk, can reconstruct all data blocks in D_f. Parity blocks (C'_s) in the set correspond to $m\frac{r}{m} = r$ row vectors of P, which form an $r \times kr$ constraint matrix S'. Equally dividing S' into k $r \times r$ blocks by column $(S'_i, i = 0, \ldots, k-1)$, the parity blocks of the set can be written as the matrix product of data chunks and S':

$$D_0S'_0 \oplus D_1S'_1 \oplus \cdots \oplus D_{k-1}S'_{k-1} = C'_s,$$

since each chunk, except for D_f, contributes $\frac{r}{m}$ blocks to the repair, there are $\frac{r}{m}$ non-zero columns in $S'_i (i \neq f)$. Meanwhile, S'_f must be invertible to retain all data blocks of D_f. □

Considering that each submatrix $P_{i,j}$ of a Z code is a permutation matrix, then the non-singular matrix S_f or S'_f, whose row vectors are selected from $P_{i,f}$s, must be a permutation matrix. *Thus, for the Z codes, S_f is not merely non-singular, but also a permutation matrix for the ORP in Theorem 5.1.*

An ORP-satisfying constraint matrix for f can identically specify all data/parity blocks that optimally reconstruct D_f. That is, row vectors of the constraint matrix correspond to parity blocks and the non-zero column vectors of S_i $(i \neq f)$ correspond to the data blocks that the ith node helps to repair. Therefore, we prove $P \xrightarrow{opt} f$ for the Z codes by finding an ORP-satisfying constraint matrix. In Figure 9, we demonstrate three ORP-satisfying constraint matrices of each data node failure for the $(2, 3)$-Z code, whose generator matrix is given in Figure 7(a).

We prove that Z codes' optimal repair bandwidth inductively, and our proof can be divided into the following steps:

(1) $P \xrightarrow{opt} f$ with $k = 2$; (Theorem 5.2),
(2) $P \xrightarrow{opt} f$ with $k = 3$; (Theorem 5.3, Corollary 5.4 and Lemma 5.5),
(3) $P \xrightarrow{opt} f$ with $k \geq 3$ and $f < k - 2$; (Inductively in Theorem 5.3),
(4) $P \xrightarrow{opt} k - 1$ with $k \geq 3$; (Inductively in Corollary 5.4),
(5) $P \xrightarrow{opt} k - 2$ with $k > 3$; (Inductively in Theorem 5.7),

Since Q indicates the generator matrix of the $(m, k + 1)$-Z code, the third step is equivalent to

(3) $Q \xrightarrow{opt} f$ $(f < k - 1)$ on the basis of $P \xrightarrow{opt} f$ $(f < k - 1)$ when $k \geq 3$; (Theorem 5.3).

Theorem 5.2. *Let P be the generator matrix of the $(m, 2)$-Z code as defined in Equation (2), then $P \xrightarrow{opt} f$ with $f = 0, 1$.*

Proof. When $f = 0$, we select the ith row vector from each P_i and all row vectors form a constraint matrix, which is given below:

$$SA = \begin{bmatrix} \hat{e}_0 & \hat{e}_0 \\ \hat{e}_1 & \hat{e}_0 \\ \vdots & \vdots \\ \hat{e}_{m-1} & \hat{e}_0 \end{bmatrix}_{m \times 2},$$

where $\hat{e}_i$ is a $1 \times m$ unit vector with 1 in the ith position and 0s elsewhere. The first block column of SA forms an $m \times m$ permutation matrix, while the second block column forms an $m \times m$ matrix, where only entries in the first column vector are 1s. Therefore, SA satisfies the ORP and $P \xrightarrow{opt} 0$.

When $f = 1$, we select the first row vector from each block row P_i as row vectors of a constraint matrix, which is

$$SB = \begin{bmatrix} \hat{e}_0 & \hat{e}_0 \\ \hat{e}_0 & \hat{e}_{m-1} \\ \vdots & \vdots \\ \hat{e}_0 & \hat{e}_1 \end{bmatrix}_{m \times 2}.$$

The second block column of SB forms an $m \times m$ permutation matrix, while the first block column forms an $m \times m$ matrix, where only entries in the first column are 1s. Therefore, $P \xrightarrow{opt} 1$. In conclusion, $P \xrightarrow{opt} f$ with $f = 0, 1$. □

Theorem 5.3. *Let T be the intermediate matrix when converting P to Q. Provided that $P \xrightarrow{opt} f(f < k-1)$, then $T \xrightarrow{opt} f(f < k-1)$ and $Q \xrightarrow{opt} f(f < k-1)$ as long as $k \geq 3$.*

Proof. Assume that $P \xrightarrow{opt} f(f < k-1)$, so there exists a constraint matrix S^P satisfying the ORP. Therefore, we construct a new constraint matrix S^T for T by selecting row vectors with the same index as row vectors of S^P in P. Obviously, S^T also satisfies the ORP, then $T \xrightarrow{opt} f$.

In the tensor extension from T to Q, each entry 1 in T is replaced by an $m \times m$ specific *cell matrix*, and an entry 0 in T is replaced by an $m \times m$ null matrix. We make the same tensor extension for all entries in S^T as they are done in the tensor extension from T to Q and the resultant matrix is denoted as S^Q. First, S^Q is a constraint matrix of Q, since numbers of rows of these two matrices are amplified m times simultaneously; second, S^Q also satisfies the ORP in that S^T also satisfies the ORP for $f < k-1$. In conclusion, as long as $P \xrightarrow{opt} f(f < k-1)$, then $T \xrightarrow{opt} f(f < k-1)$, and $Q \xrightarrow{opt} f(f < k-1)$. □

Corollary 5.4. *Let the generator matrix A be a block matrix as follows:*

$$A = \begin{bmatrix} A_{0,0} & A_{0,1} & \dots & A_{0,k-1} \\ A_{1,0} & A_{1,1} & \dots & A_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m-1,0} & A_{m-1,1} & \dots & A_{m-1,k-1} \end{bmatrix}_{m \times k},$$

where each submatrix $A_{i,j}$ is an $r \times r$ permutation matrix, and there is a block matrix

$$B = \begin{bmatrix} A_{0,0} \otimes I_m & \dots & A_{0,k-2} \otimes I_m & A_{0,k-1} \otimes I_m \\ A_{1,0} \otimes I_m & \dots & A_{1,k-2} \otimes I_m & A_{1,k-1} \otimes I_m^{+} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m-1,0} \otimes I_m & \dots & A_{m-1,k-2} \otimes I_m & A_{m-1,k-1} \otimes I_m^{(m-1)+} \end{bmatrix}_{m \times k},$$

then $B \xrightarrow{opt} k-1$.

Proof. From the first row of B to the last, we choose the first row vector every m rows as row vectors of the constraint matrix, which is presented as follows:

$$S = \begin{bmatrix} A_{0,0} \otimes \hat{e}_0 & \dots & A_{0,k-2} \otimes \hat{e}_0 & A_{0,k-1} \otimes \hat{e}_0 \\ A_{1,0} \otimes \hat{e}_0 & \dots & A_{1,k-2} \otimes \hat{e}_0 & A_{1,k-1} \otimes \hat{e}_1 \\ \vdots & \vdots & \vdots & \vdots \\ A_{m-1,0} \otimes \hat{e}_0 & \dots & A_{m-1,k-2} \otimes \hat{e}_0 & A_{m-1,k-1} \otimes \hat{e}_{m-1} \end{bmatrix}_{m \times k}.$$

With each non-zero entry $A_{i,j}$ replaced by the product of the entry and the vector $\hat{e}_k$ and zero entry replaced by the null vector, we get the result of the tensor product $A_{i,j} \otimes \hat{e}_k$. Since submatrix $A_{i,j}$ is a permutation matrix, the last block column is also a permutation matrix. Meanwhile, $A_{i,j} \otimes \hat{e}_k$ has non-zero columns with a ratio of $\frac{1}{m}$. Therefore, the constraint matrix S satisfies the ORP, and $B \xrightarrow{opt} k-1$. □

If we treat the block matrix A in the Corollary 5.4 as the intermediate matrix T, then the matrix B is just the generator matrix Q of the $(m, k+1)$-Z code. Hence, $Q \xrightarrow{opt} k$ and $P \xrightarrow{opt} k-1$.

Lemma 5.5. *Let the generator matrix A be a block matrix as follows:*

$$A = \begin{bmatrix} I_m \otimes I_m & I_m \otimes I_m & I_m \otimes I_m \\ I_m \otimes I_m & I_m^{+} \otimes I_m & I_m^{+} \otimes I_m^{+} \\ \vdots & \vdots & \vdots \\ I_m \otimes I_m & I_m^{(m-1)+} \otimes I_m & I_m^{(m-1)+} \otimes I_m^{(m-1)+} \end{bmatrix}_{m \times 3},$$

then $A \xrightarrow{opt} 1$.

Proof. A_i is the ith block row of A:

$$A_i = \begin{bmatrix} I_m \otimes I_m & I_m^{i+} \otimes I_m & I_m^{i+} \otimes I_m^{i+} \end{bmatrix}.$$

In A_i, we select the $(im+i)$th $(i = 0, \dots, m-1)$ row vectors, and m row vectors forms an $m \times 3m^2$ matrix as follows:

$$L_i = \begin{bmatrix} \hat{e}_0 \otimes \hat{e}_0 & \hat{e}_{(-i)\%m} \otimes \hat{e}_0 & \hat{e}_{(-i)\%m} \otimes \hat{e}_{(-i)\%m} \\ \hat{e}_1 \otimes \hat{e}_1 & \hat{e}_{(1-i)\%m} \otimes \hat{e}_1 & \hat{e}_{(1-i)\%m} \otimes \hat{e}_{(1-i)\%m} \\ \vdots & \vdots & \vdots \\ \hat{e}_{m-1} \otimes \hat{e}_{m-1} & \hat{e}_{(m-1-i)\%m} \otimes \hat{e}_{m-1} & \hat{e}_{(m-1-i)\%m} \otimes \hat{e}_{(m-1-i)\%m} \end{bmatrix},$$

where $i\%m$ represents i modulo m and the result is always non-negative. We construct a constraint matrix S for $f = 1$ by aligning all L_is by row, which is an $m^2 \times 3m^2$ matrix given below:

$$S = [L_0\ L_1\ \dots\ L_{m-1}]^{\intercal}.$$

We divide S equally by column into 3 blocks of size $m^2 \times m^2$, that is, $S = [S_0\ S_1\ S_2]$. S_0 and S_2 have only m non-zero columns, because all their row vectors belong to the set $\{\hat{e}_i \otimes \hat{e}_i,\ i = 0,$

$1, \ldots, m-1\}$. S_1 is a permutation matrix, since every row vector in the square matrix S_1 is unique. Therefore, constraint matrix S satisfies the ORP and $A \xrightarrow{opt} 1$. □

The matrix A of Lemma 5.5 is actually the $(m, 3)$-Z code's generator matrix P, and $P \xrightarrow{opt} 2$ according to Corollary 5.4. When the faulty chunk is D_0 for the $(m, 3)$-Z code, that is, $f = 0$, we can also find a constraint matrix S satisfying the ORP as Theorems 5.2 and 5.3, then $P \xrightarrow{opt} 0$. Consequently, $P \xrightarrow{opt} f$ with $f = 0, 1, 2$.

Lemma 5.6. *Let the generator matrix A be a block matrix given below:*

$$A = \begin{bmatrix} \overbrace{I_m \otimes I_m & \ldots & I_m \otimes I_m}^{k-1 \text{ identical columns}} & I_m \otimes I_m & I_m \otimes I_m \\ I_m \otimes I_m & \ldots & I_m \otimes I_m & I_m^+ \otimes I_m & I_m^+ \otimes I_m^+ \\ \vdots & \vdots & \vdots & \vdots & \\ I_m \otimes I_m & \ldots & I_m \otimes I_m & I_m^{(m-1)+} \otimes I_m & I_m^{(m-1)+} \otimes I_m^{(m-1)+} \end{bmatrix}_{m \times (k+1)},$$

where the first $k-1$ block columns are all the same as $[I_m \otimes I_m \ldots I_m \otimes I_m]^\intercal$, then $A \xrightarrow{opt} k-1$.

The matrix A in Lemma 5.6 can be seen as the matrix A in Lemma 5.5 with the first column being duplicated $k-2$ times. Thus, we can find a constraint matrix satisfying the ORP for $f = k-1$ by selecting the row vectors that have the same indices as row vectors in the constraint matrix of the matrix A in Lemma 5.5. On the basis of the above, we finally prove the optimal repair bandwidth for the failure of the second to last data node.

Theorem 5.7. *Let P and Q be the generator matrices of the (m, k)-Z code and $(m, k+1)$-Z code separately, then $P \xrightarrow{opt} k-2$ when $k > 3$.*

Proof. For the $(m, 2)$-Z code and the $(m, 3)$-Z code, we have proven that $Q \xrightarrow{opt} k-1$ in Theorem 5.2 and Lemma 5.5.

When $k > 2$, each submatrix $Q_{i,j}$ of Q is a tensor product of k *cell matrices*, and the tensor product of the last two cell matrices is exactly the block of A in Lemma 5.6. We define $A_{i,j}$ as the submatrix, which is a tensor product of two cell matrices, of A in Lemma 5.6. Thus, each $Q_{i,j}$ is a tensor product of two matrices, that is, $Q_{i,j} = H_{i,j} \otimes A_{i,j}$, where $H_{i,j}$ is the tensor product of $k-2$ *cell matrices*. Since $A \xrightarrow{opt} k-1$, there exists an $m \times (k+1)m^2$ *constraint matrix* S^A satisfying the ORP and S^A can be divided into $m \times (k+1)$ blocks (denoted by $S^A_{i,j}$). Let

$$S^Q = [S^Q_{i,j}]_{m \times (k+1)} = [H_{i,j} \otimes S^A_{i,j}]_{m \times (k+1)}.$$

First, S^Q is a *constraint matrix* of Q; second, S^Q satisfies the ORP on the basis of S^A satisfying the ORP. Thus, $Q \xrightarrow{opt} k-1$ and $P \xrightarrow{opt} k-2$. □

6 ANALYTICAL EVALUATIONS

We analyze the storage efficiency, repair bandwidth, update cost, and computation complexity of the Z/GZ codes in this section.

6.1 Optimal Repair Bandwidth under the Minimum Storage

An (m, k)-Z/GZ code consumes the minimum storage capacity, equal to that of a RS code with the same parameter set (m, k). That is, for each stripe over n nodes, the Z/GZ codes store a data chunk of size $\frac{M}{k}$ in each node.

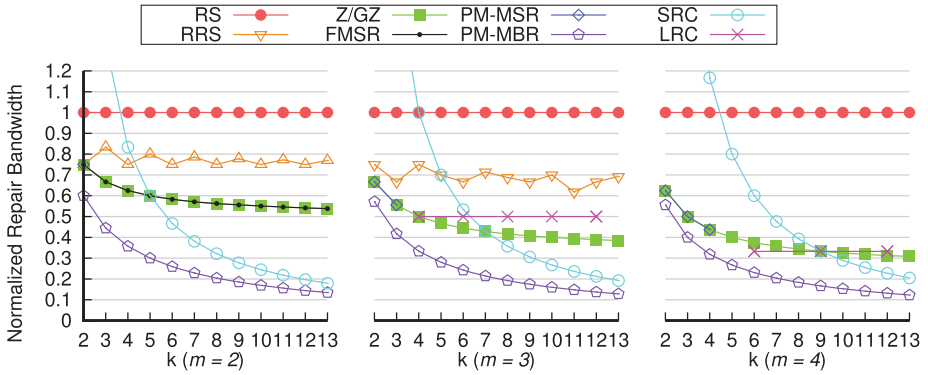


Fig. 10. Repair bandwidth under a single data node failure of some repair-bandwidth-efficient codes as a function of m and k, normalized to the size of the native data.

Under the minimum storage usage, the Z/GZ codes achieve the optimal repair bandwidth for a single data node failure, as given in Equation (1). Figure 10 plots the repair bandwidth under a single data node failure of some repair-bandwidth-efficient codes as a function of k and m, normalized to the size of the native data. Such repair bandwidth is calculated as the ratio of the amount of data required for repairing a data chunk to the amount of the native data in a stripe M. Compared to the RS codes, which must retrieve the whole chunk from any k remaining nodes, the amount of repair bandwidth saved by the Z/GZ codes is

$$k\frac{M}{k} - (n-1)\frac{1}{m}\frac{M}{k} = \frac{(m-1)(k-1)}{mk}M.$$

For the same m value, the Z/GZ codes with a larger k requires less repair bandwidth, while the Z/GZ codes with a larger m requires less repair bandwidth for the same k. In addition, the repair of a parity chunk by the Z/GZ codes costs the same repair bandwidth as the RS codes, with no repair bandwidth saved, since repairing a parity chunk is exactly the same process of re-encoding the parity chunks with all data chunks being involved.

Besides the Z/GZ codes, as plotted, the FMSR and PM-MSR codes can also achieve the optimal repair, as well as the minimum storage usage, but their parameters' limitations make them only applicable on some special cases. The PM-MBR, SRC, and LRC codes are not the minimum-storage codes, so more repair bandwidth could be reduced for the same parameter set (m, k). In Figure 10, we use one global parity chunk for all data chunks (i.e., $g = 1$) and one local parity chunk for data chunks of each group (i.e., $k/l = m - 1$) for the LRC codes, while we also make the SRC codes have the same storage capacity as the PM-MBR codes of the same parameter set (m, k), so their repair bandwidth are comparable.

In Figure 11, We plot the normalized repair bandwidths of the Z/GZ codes and other repair-bandwidth-efficient codes. The X axis is the storage overhead, defined as the ratio of parity data size and the native data size. With the growth of k, the proportion of the parity size in a stripe decreases, and the storage overhead decreases as well. We make $f = k$ for the SRC codes to avoid excessive storage overhead. When $m = 2$, the Z/GZ codes can achieve the equivalent normalized repair bandwidth as the FMSR codes with the same storage overhead, while the Z/GZ codes can save more repair bandwidth as m increases. Under the condition of the same storage overhead, the GZ codes can greatly reduce more repair bandwidth than the SRC and RRS codes, and don't have to lose the MDS property as the LRC codes.

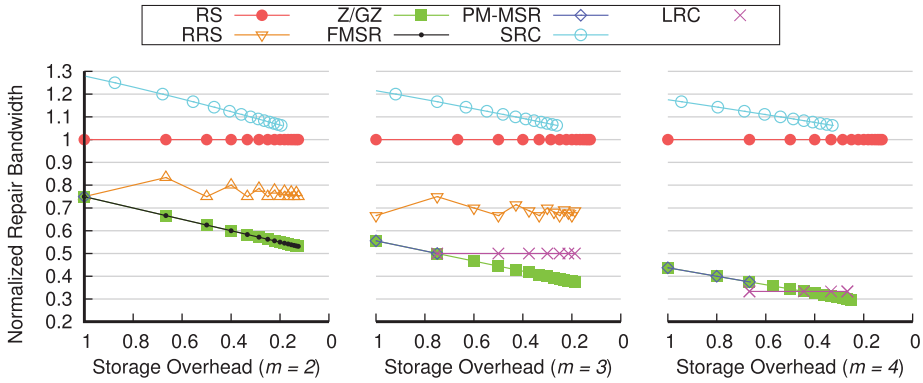


Fig. 11. Normalized repair bandwidth vs. storage overhead of some repair-bandwidth-efficient codes. From the left to the right, k increases from m to 16 and the storage overhead also decreases. (For the SRC codes, we let $f = k$, while for the LRC codes, we let $g = 1$ and $l = 2, 3$. Since the storage overheads of the PM-MBR codes are always greater than 1, they are not plotted.)

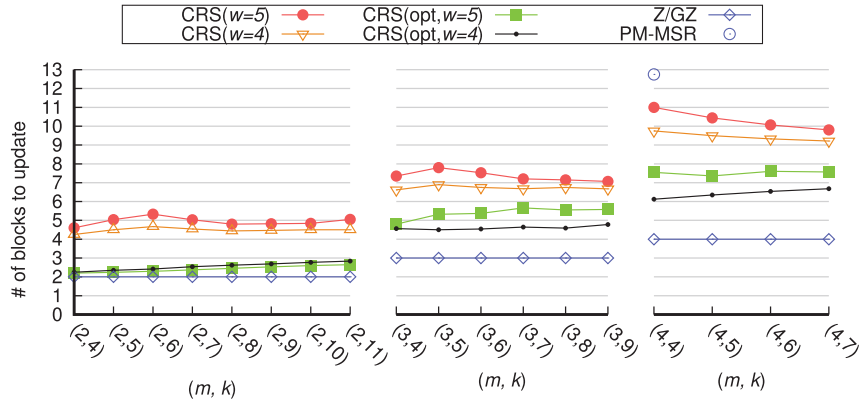


Fig. 12. Update cost of some CRS, Z/GZ, and PM-MSR codes.

6.2 Update Cost

Once a data block is updated, several parity blocks must be updated for consistency. We define the update cost as the number of parity blocks to update when a data block is modified. The update cost of the Z/GZ codes is m, since each data block is linearly combined once in a parity block of a stripe. Thus, when a data block is modified, only m parity blocks are updated. Figure 12 compares the update cost of some Z/GZ codes, CRS codes, optimized CRS codes (Plank and Lihao 2006) and PM-MSR codes, where the Galois field for generating Cauchy matrices is GF(2^4) ($w = 4$) or GF(2^5) ($w = 5$). The optimized CRS codes are a kind of CRS codes with minimum 1s in the generator matrices by exhaustive search. Since the LRC and SRC codes are two coding schemes that combine two or more specific coding methods, so their update costs are relative to what kind of coding method they are using in practice. Obviously, the Z/GZ codes have the least update cost among all cases. Although the update cost of the opt-CRS code is greatly reduced, it is still higher than that of the Z/GZ codes of the same m and k.

6.3 Computation Complexity of Encoding and Repairing

As other linear codes over a Galois field, encoding and repairing of erasure codes can be decomposed into basic operations (Plank et al. 2013b): $xR_1 \oplus R_2$s, where x is an entry in the generator matrix, and R_1 and R_2 are two regions of data, so the basic operation represents the XORing of two regions of data with one of them being multiplied by a factor of x. $R_1 \oplus R_2$ is the specific case of $xR_1 \oplus R_2$ as $x = 1$, and we regard them as being of the same complexity. On the basis of the above, we measure the encoding and repairing complexity by the number of $xR_1 \oplus R_2$ operations to generate a parity block or recover a data block.

Take the encoding of an (m, k)-Z/GZ code as an example, each parity block is a linear combination of k data blocks. Since there are k non-zero entries in each row of the generator matrix, the encoding can be carried out by $k + 1$ $xR_1 \oplus R_2$ operations. Similarly, the generator matrix of an (m, k)-RS code is an $m \times k$ matrix and there are also k non-zero entries in each row (Plank 2005), so encoding can be carried out by $k + 1$ $xR_1 \oplus R_2$ operations as well.

We separately discuss the complexity of repairing a parity chunk and a data chunk for an (m, k)-Z/GZ code. *The complexity of repairing a parity block is $k + 1$ $xR_1 \oplus R_2$ operations*, since repairing of a parity block is the process of re-encoding it. From Section 3.3, we have shown that each lost data block of the faulty chunk is linked to a constraint equation, which is also associated with k other different blocks. Thus, the lost data block is reconstructed by solving the constraint equation. Therefore, *the complexity of repairing a data block equals the number of edges that a constraint equation associates with, which is $k + 1$*. An (m, k)-RS code has the same encoding complexity and repairing complexity, since the repairing is the process of calculating the combination of any k blocks with entries in the repair matrix as coefficient. *In conclusion, the Z/GZ codes and RS codes have the same encoding and repairing complexity*, which is experimentally verified in the next section.

7 EXPERIMENTAL EVALUATIONS

Our experimental evaluations include two parts. First, we evaluate performances of some repair-bandwidth-efficient codes in a machine from aspects of the in-memory encoding/repairing speed, the amount of data needed to repair a single data chunk and parity updating cost for a data block. Second, we take network and disk I/O into account and measure upload/repair response times of four systematic MDS codes in a real distributed storage system of 7 nodes, which are connected to a 1 Gigabit network. Each node is a machine with an Intel Xeon E5620 2.4GHz CPU and 12GB RAM. To make a more direct comparison, we fix all storage overhead of codes at the minimum, and do not consider the LRC, PM-MBR, and SRC codes that require extra storage.

7.1 Encoding/Repairing Speed

In a single machine with a 3.5GHz Intel Core i7-4770K CPU, we empirically evaluate the in-memory encoding and repairing speeds of several codes, which are measured by the amount of data encoded or rebuilt per second based on the following equation:

$$\text{Encoding Speed} = \frac{\text{Native data size}(M)}{\text{Time cost}},$$

$$\text{Repairing Speed} = \frac{\text{Size of data of a node}(\frac{M}{k})}{\text{Time cost}}.$$

All codes employed in this experiment are the minimum-storage codes and data to be encoded or to be repaired are all the same size, their speeds can be fairly compared.

The RS and CRS codes are two most commonly used erasure codes with first-rank coding speed in the last decades. Some existing repair-bandwidth-efficient codes, such as LRC codes and SRC

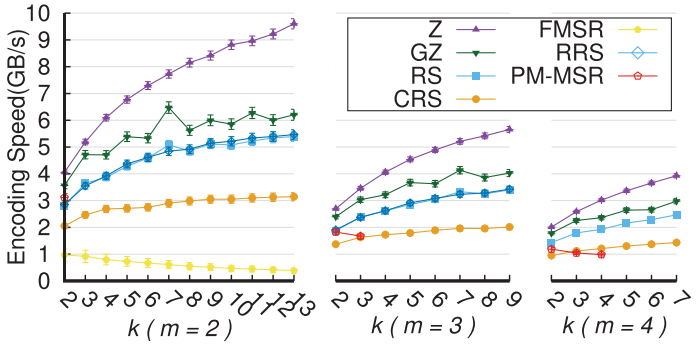


Fig. 13. Encoding speed of the Z, GZ, CRS, RS, FMSR, RRS, and PM-MSR codes for a 1GB file.

codes, are constructed over the RS or CRS codes, so the performances of the former are usually worse than those of the latter (Papailiopoulos et al. 2012; Huang et al. 2012). Our encoding/repairing speed evaluation method is exactly the same as that of the SD codes (Plank et al. 2013a) and STAIR codes (Li and Lee. 2014). For the GZ, RS, RRS, PM-MSR, and FMSR codes, their generator matrices are all generated over the Galois field GF(2^8), so encoding and repairing are over the same field. The fast Galois Field arithmetic library GF-complete provided by Plank et al. (2013b) is leveraged for fast Galois field arithmetic.

The experiments cover the situations where $m = 2, 3, 4$, which are the most common cases of failure tolerance. For each code, we test all combinations of (m, k)s with the restrictions of $r < 7,000$. We encode/repair a region of data with size up to 1GB, so the size of a block would not be too small for Z/GZ codes. Each test is run 50 times with the average speed as the final result. Encoding speeds are plotted in Figure 13. Observe that some codes are not achievable for all parameters. For example, the FMSR codes are only exist when $m = 2$ and the PM-MSR codes are only available when $k \leq m$. The Z codes gain the fastest encoding speed among all reference codes, benefiting from the fast bit-wise XOR. Due to the similar encoding manner, the RS codes and the RRS codes perform approximately the same in encoding speed, while speeds of GZ are also close to the RS codes, since they have the same theoretical computation complexity as we discussed in Subsection 6.3. The PM-MSR codes and the FMSR codes are non-systematic, so the number of non-zero entries in the generator matrix greatly increases when the parameter k increases, resulting in decreasing encoding speeds.

Figure 14 plots speeds of repairing a single data chunk of 1GB for some codes. We only measure the speed of repairing a data chunk, since the process of repairing a parity chunk is exactly the same as the process of re-encoding it. For all codes, their repairing speeds exhibit a decreasing trend as k increases, because more data blocks are involved in reconstructing the same size of data, incurring more computation. The Z/GZ codes and RS codes have the fast repairing speeds and the very close repairing speeds among them confirms that these three codes share the same computation complexity for repairing. Both the Z and GZ codes offer a speed on the order of GB/s, which suggests that computation would no longer be the performance bottleneck when applying the Z/GZ codes in a practical storage system.

7.2 Data Read for Repairing a Single Data Chunk

Figure 15 shows the total amount of data transferred over the network to reconstruct a single data chunk, and all codes are with the same parameters of $m = 4, k = 4$ or $m = 2, k = 4$. For the PM-MSR codes, we also plot the amount of data accessed from the disk, since they read a greater number

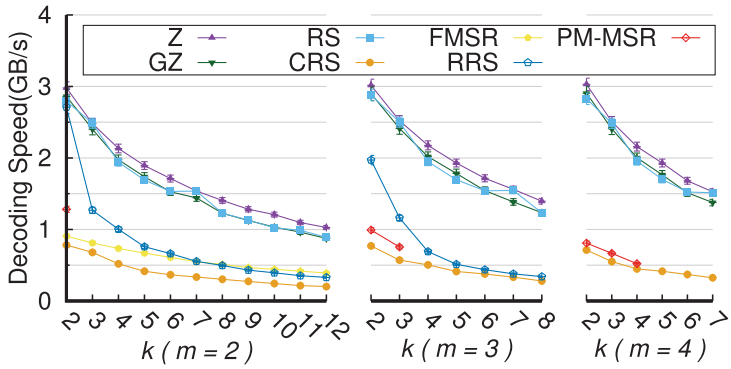


Fig. 14. Decoding speed of the Z, GZ, CRS, RS, FMSR, RRS, and PM-MSR codes for reconstructing 1GB data in a single node.

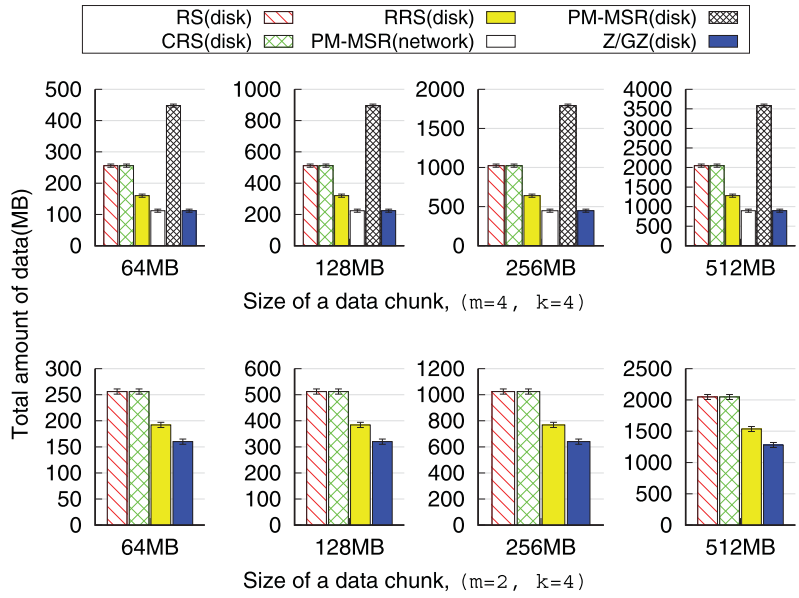


Fig. 15. Total amount of data required for repairing a single data chunk, either read from disks or transferred through the network.

of blocks from the disk than blocks transferred over the network. Except the PM-MSR codes, all other erasure codes transfer the same number of blocks over the network as they access from the disks. Two points can be seen from the results: First, both the RRS codes and the Z/GZ codes can reduce the total amount of data read, compared to the RS and CRS codes; second, the PM-MSR codes save the total amount of data transferred over the network at the cost of more disk accesses, and the total amount of data read from disks are even larger than the size of the native data.

7.3 Parity Updating Cost for a Single Data Block

We measure the update cost by counting how much amount of parity data are updated when a single data block is updated. In our experimental program, the size of a data/parity block is either

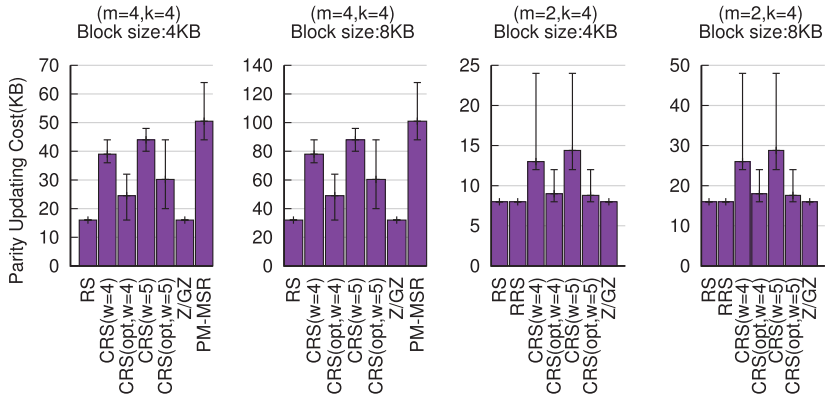


Fig. 16. Amount of parity data are updated when a data block is updated.

4KB or 8KB, and parameter sets of all codes are $(m = 4, k = 4)$ and $(m = 2, k = 4)$. One parameter constraint of the PM-MSR codes is $k \leq m$, so they are not plotted in the cases of $(m = 2, k = 4)$. The RRS codes are not plotted under the parameter set $(m = 4, k = 4)$ in this and latter subsections, because they are only available when $m = 2, 3$. For some codes, parity updating costs of different data blocks are not identical, and Figure 16 shows the average update costs of Z/GZ codes and some other codes. We also give the minimum and maximum update cost of each test case by means of error bars. The Z/GZ codes outperform other codes with the minimum and constant parity updating costs, since all updates of Z/GZ codes are evenly distributed in m parity nodes and every time only one parity block in each parity node is updated. Therefore, the Z/GZ codes suffer less update burdens when performing the update operation.

7.4 Response Time of Repairing

We applied four systematic MDS codes—GZ, RS, CRS and RRS codes—in a practical distributed storage system consisting of 7 nodes, where one node is metadata server and other 6 nodes are data servers. All nodes are connected via a 1,000Mbps network. We measure the response times of the upload operation and repair operation. The former is the time spent writing a file into the system, while the latter is the time spent repairing a lost data chunk. Since the Z and GZ codes of the same k and m combination have exactly the same theoretical repair bandwidth, resulting in the similar practical performance on the response time, we only consider the GZ codes in this subsection.

In Figure 17, we plot the response times of uploading a file and repairing a chunk of different sizes, with two parameter sets: $(k = 4, m = 2)$ and $(k = 4, m = 4)$. Each operation is performed 50 times with the average time as the result being plotted, and the data set consists of randomly generated files. Note that the four codes have the comparable response time for the upload operation, while the GZ codes gain a 37.5% to 48.6% reduction in the repair response time, compared to the RS and CRS codes. The response time of the RRS code is between those of the RS/CRS code and the GZ code. This clear advantage of the Z/GZ codes in repair response time stems from their significant reduction in repair bandwidth by attaining the optimal repair bandwidth, resulting in less data being transferred over the network.

In Figure 18, we further break down the response time of uploading or repairing a 64MB data chunk into two key components: network and disk-I/O, and computation for coding. Observe

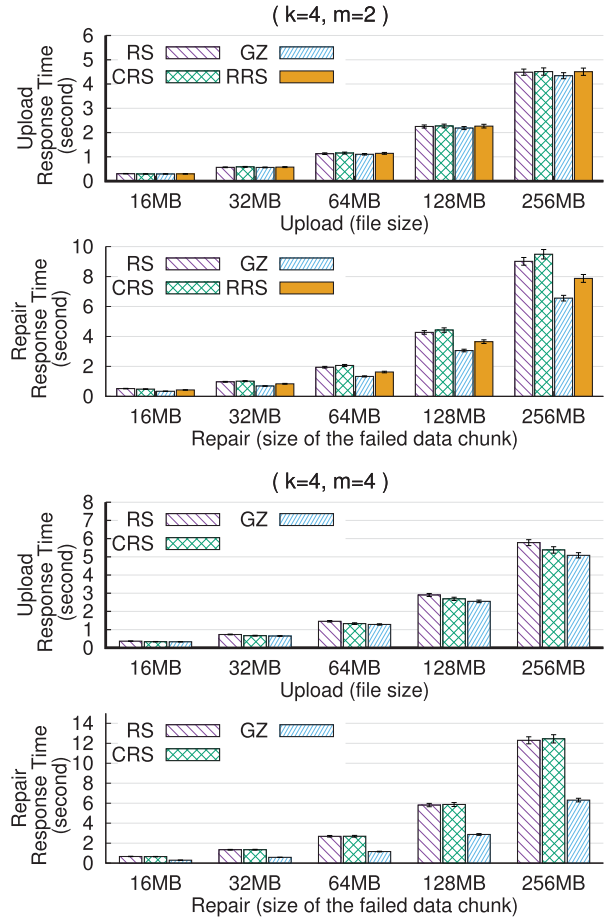


Fig. 17. Response times of the upload and repair operations when parameters are $(k = 4, m = 2)$ and $(k = 4, m = 4)$.

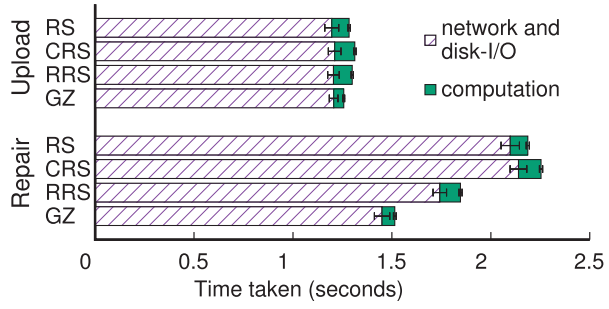


Fig. 18. Breakdown of the response times of uploading and repairing a 64MB file (or data chunk) when $k = 4$ and $m = 2$.

Table 5. Characteristics of the Two Traces

Workloads	Average read request size	Read ratio	Elapsed time
Financial	2.25 KB	23.2%	14.2 hours
Websearch	15.15 KB	99.9%	4.3 hours

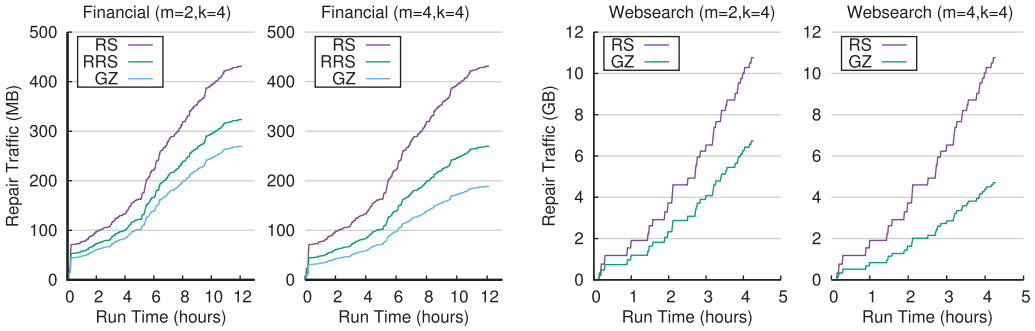


Fig. 19. Cumulative repair traffics of different erasure codes under the two workloads.

that most of the time are spent on the data transmission, while the computation time for coding (encoding or repairing) is less than 6%.

7.5 Repair Traffic with Real-World Workloads

We further evaluate practical repair traffic of Z/GZ codes in a erasure-coded distributed storage system, under two real-world workloads. Financial and Websearch are two I/O traces collected from applications in a large financial institution and a popular search engine (uma 2017), and their characteristics are summarized in Table 5. We randomly inject failures into storage nodes with a failure rate of 4% and aggregate repair traffics caused by degraded read requests, which are read requests of the workloads to the failure nodes. Figure 19 demonstrates repair traffics of the storage system as a functions of runtime, under different erasure codes and the two workloads. The Websearch has a higher proportion of read requests with a larger average data size than the Financial, so it has more repair traffic. Among all codes, the GZ codes encounter with the least repair traffic, which confirms their theoretical minimum repair bandwidths.

8 CONSTRAINTS AND FUTURE WORK

Based on the theoretical analysis and experimental evaluation, we present open research issues of Z/GZ codes: sub-packetization and inefficiency for multiple, concurrent failures. The large sub-packetization, which breaks data of each node into a great number of blocks, derives from the generator matrix's construction that utilizes the tensor product. The tensor product causes an exponential increase of a Z/GZ code's generator matrix when its parameters increase.

The large sub-packetization affects practical systems on two aspects: First, more blocks in each node need to be maintained. Although a great number of blocks don't increase the actual storage overhead, but it may place a big burden on data/metadata management if not well operated. In a practical storage system, contiguous blocks of a chunk in a storage node do not necessarily arrange as independent files by the local file system, and the best practice is to store one chunk as a file with the block size as a multiple of the minimum access unit size. However, such minimum access unit (like sector in disk array and page in SSD) of the system in return limits the number of blocks in a chunk, as well as the the maximum feasible parameters of Z/GZ codes. Thus, although Z/GZ codes

theoretically exist in any parameter set of (m, k), their parameters must adjust to the configurations of the practical storage system. Second, an oversize sub-packetization requires more storage for saving a bigger generator matrix and may interfere with the coding performance by making a dent in the data locality and I/O efficiency. Specifically, when the size of the generator matrix is big enough to compare with the block size, both types of data will race to seize the cache, thereby increasing the cache miss and slowing down the coding speed. Fortunately, generator matrices of Z/GZ codes are sparse and existing matrix-compression methods can be used to condense them. In the experimental evaluation in Section 7, we use a list-in-list compression method (Golub and Van Loan 2012), which stores one list for each row of a matrix, with each entry containing the column index and the value. One merit of this method is compression results of Z/GZ codes' generator matrices are also matrices without zero entries, so coding processes of Z/GZ codes have just little change.

The ability of tolerating the maximum concurrent failure events for Z/GZ codes is conditional and their minimum repair cost for multiple failures remains to be further studied. Z codes are non-MDS codes and can live with at most m simultaneous parity failures, but how many concurrent multiple failures of arbitrary nodes can be tolerated at least still need further research. The MDS property of GZ codes isn't theoretically proofed but is verified by the exhaustive validation in a reasonable range of parameters. In some cases of Z/GZ codes with $m > 2$, we found that 2 or more simultaneous failures can be recovered with less repair cost than the average, with the help of some existing repair schemes (Khan et al. 2012, 2011) that find a minimum or near-minimum solutions by the heuristic search. That is, all lost data of multiple nodes can be simultaneously retrieved with repair data size less than the size of the native data M in some specific conditions. As far as our experience goes, failures of multiple data nodes can be repaired with less repair cost than failures of multiple parity nodes for Z/GZ codes, and our future work is to find a determinate repair scheme to reduce the repair cost for multiple data nodes.

9 CONCLUSIONS

We propose a family repair-bandwidth-efficient erasure codes, called the Z codes for distributed storage systems. The Z codes not only achieve the optimal repair bandwidth for repairing a single data node's failure but also attain the minimum storage property as the RS and CRS codes. The Z codes have many other beneficial properties that make them suitable for distributed storage systems. Moreover, we generalize the Z codes to the GZ codes to attain the MDS property. The Z/GZ codes have comparable encoding and repairing performances with the RS codes and significantly outperform the CRS codes and other repair-bandwidth-efficient codes.

ACKNOWLEDGMENTS

The authors acknowledge the anonymous reviewers for their helpful and constructive suggestions that greatly improve the final version of the article. They also thank Rashmi K. Vinayak for providing the source codes of the PM-MSR and PM-MBR codes.

REFERENCES

UMass Trace Repository. 2017. Homepage. Retrieved from http://traces.cs.umass.edu/index.php/Main/HomePage.

Johannes Blömer, Malik Kalfane, Richard Karp, Marek Karpinski, Michael Luby, and David Zuckerman. 1995. An XOR-based erasure-resilient coding scheme. ICSI Technical Report TR-95-048. http://www.icsi.berkeley.edu/ftp/global/pub/techreports/1995/tr-95-048.

Henry C. H. Chen, Yuchong Hu, Patrick P. C. Lee, and Yang Tang. 2014. NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds. *IEEE Trans. Comput.* 63, 1 (Jan 2014), 31–44.

Alexandros G. Dimakis, P. Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. 2010. Network coding for distributed storage systems. *IEEE Trans. Info. Theory* 56, 9 (2010), 4539–4551.

Gene H. Golub and Charles F. Van Loan. 2012. *Matrix Computations*. Vol. 3. JHU Press.

Cheng Huang, Huseyin Simitci, Xu Yikang, Aaron Ogus, Brad Calder, Parikshit Gopalan, Li Jin, and Sergey Yekhanin. 2012. Erasure coding in windows azure storage. In *Proceedings of the USENIX ATC*.

Osama Khan, Randal Burns, James Plank, and Cheng Huang. 2011. In search of I/O-optimal recovery from disk failures. In *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*. USENIX Association, 6–6.

Osama Khan, Randal C. Burns, James S. Plank, William Pierce, and Cheng Huang. 2012. Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads. In *Proceedings of Usenix Conference on File and Storage Technologies (FAST'12)*.

Mingqiang Li and Patrick P. C. Lee. 2014. STAIR codes: a general family of erasure codes for tolerating device and sector failures in practical storage systems. In *Proceedings of Usenix Conference on File and Storage Technologies (FAST'14)*. 147–162.

Qing Liu, Dan Feng, Zhan Shi, and Min Fu. 2015a. General Functional Regenerating Codes with Uncoded Repair for Distributed Storage System. In *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid'15)*. 372–381. DOI: http://dx.doi.org/10.1109/CCGrid.2015.38

Qing Liu, Dan Feng, Zhan Shi, and Min Fu. 2015b. Z codes: General Systematic Erasure Codes with Optimal Repair Bandwidth and Storage for Distributed Storage Systems. In *Proceedings of the 2015 IEEE 34rd International Symposium on Reliable Distributed Systems (SRDS'15)*. IEEE.

Dimitris S. Papailiopoulos, Jianqiang Luo, Alexandros G. Dimakis, Cheng Huang, and Jin Li. 2012. Simple regenerating codes: Network coding for cloud storage. In *Proceedings IEEE INFOCOM*. IEEE, 2801–2805.

Sameer Pawar, Nima Noorshams, Salim El Rouayheb, and Kannan Ramchandran. 2011. DRESS codes for the storage cloud: Simple randomized constructions. In *Proceedings of the IEEE International Symposium on Information Theory Proceedings (ISIT'11)*. IEEE, 2338–2342.

James S. Plank. 2005. T1: erasure codes for storage applications. In *Proceedings of Usenix Conference on File and Storage Technologies (FAST'05)*. 1–74.

James S. Plank, Mario Blaum, and James L. Hafner. 2013a. SD codes: Erasure codes designed for how storage systems really fail. In *Proceedings of Usenix Conference on File and Storage Technologies (FAST'13)*.

James S. Plank and Ying Ding. 2005. Note: Correction to the 1997 tutorial on Reed–Solomon coding. *Software: Pract. Exp.* 35, 2 (2005), 189–194.

James S. Plank, Kevin M. Greenan, and Ethan L. Miller. 2013b. Screaming fast Galois Field arithmetic using Intel SIMD instructions. In *Proceedings of the 11th Usenix Conference on File and Storage Technologies (FAST'13), San Jose*.

James S. Plank and Xu Lihao. 2006. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications. In *Proceedings of the 5th IEEE International Symposium on Network Computing and Applications (NCA'06)*. IEEE, 173–180.

K. V. Rashmi, Preetum Nakkiran, Jingyan Wang, Nihar B. Shah, and Kannan Ramchandran. 2015. Having Your Cake and Eating It Too: Jointly Optimal Erasure Codes for I/O, Storage, and Network-bandwidth. In *Proceedings of the Usenix Conference on File and Storage Technologies (FAST'15)*. 81–94.

K. V. Rashmi, Nihar B. Shah, Dikang Gu, Hairong Kuang, Dhruba Borthakur, and Kannan Ramchandran. 2013. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster. *Proceedings of USENIX HotStorage* (2013).

I. S. Reed and G. Solomon. 1960. Polynomial codes over certain finite fields. *J. Soc. Industr. Appl. Math.* 8, 2 (1960), 300–304.

Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros G. Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. 2013. Xoring elephants: Novel erasure codes for big data. In *Proceedings of the Very Large Data Base Endowment (VLDB'13)*, Vol. 6. VLDB Endowment, 325–336.

Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. 2013. Zigzag codes: MDS array codes with optimal rebuilding. *IEEE Trans. Info. Theory* 59, 3 (2013), 1597–1616.

Michael R. Tanner. 1981. A recursive approach to low complexity codes. *IEEE Trans. Info. Theory* 27, 5 (Sep 1981), 533–547. DOI: http://dx.doi.org/10.1109/TIT.1981.1056404

Received July 2015; revised June 2017; accepted June 2017