Flexible Partial MDS Codes

Weiqi Li, Taiting Lu, Zhiying Wang, Hamid Jafarkhani
Center for Pervasive Communications and Computing (CPCC), University of California, Irvine, USA
{weiqil4, taitingl, zhiying, hamidj}@uci.edu

***Abstract*—The partial MDS (PMDS) code was introduced by Blaum et al. for RAID systems. Given the redundancy level and the number of symbols in each node, PMDS codes can tolerate a mixed type of failures consisting of entire node failures and partial errors (symbols failures). Aiming at reducing the expected accessing latency, this paper presents flexible PMDS codes that can recover the information from a flexible number of nodes according to the number of available nodes, while the total number of symbols required remains the same. We analyze the reliability and latency of our flexible PMDS codes.**

I. Introduction

Redundant Arrays of Independent Disks (RAID) architecture is widely applied in storage systems to prevent data loss and failures. By assigning one or more disks to parity, *maximum distance separable* (MDS) codes can be used to prevent disk failures. However, *Solid-State Drives* (SSDs) bring new challenges, where the system has to overcome a mixed type of failures consisting of entire node failures and partial errors (symbol failures). *Partial MDS* (PMDS) codes were first introduced in [1] to overcome this mixed type of failures. A code consisting of an $\ell \times n$ array is an (n, k, ℓ, s) PMDS code if it can tolerate $n-k$ node failures and s additional symbol failures in the code. The general construction is given in [2] to provide such an (n, k, ℓ, s) PMDS code with arbitrary k and s.

However, the PMDS codes are designed for a given redundancy level $n-k$, while in practical systems, the number of node failures is undetermined. The redundant storage nodes are wasted when the number of node failures is smaller than $n-k$, and the incurred latency is the same as the case of $n-k$ failures. In this paper, we propose and construct flexible PMDS codes satisfying Definition 1, and show that they achieve low latency while maintaining high reliability.

Definition 1. An (n, k, ℓ, s) flexible PMDS code is parameterized by the set $\{(k_j, \ell_j) : 1 \le j \le a\}$,

$$k_j \ell_j = k\ell, 1 \le j \le a, k_1 > ... > k_a = k, \ell_a = \ell, \quad (1)$$

such that by reading ℓ_j symbols in each node, we can tolerate $n-k_j$ node failures and s additional symbol failures, for all $1 \le j \le a$.

Notation. In this paper, for a positive integer i, we denote $[i] = \{1, 2, \ldots, i\}$.

II. Constructions

A general construction of PMDS codes is proposed in [2] for any k and s using Gabidulin code. In this section, we first introduce the construction in [2] and then show how to apply it to flexible PMDS codes.

An (N, K) Gabidulin code over the finite field $\mathbb{F} = GF(q^L), L \ge N$ is defined by the polynomial $f(x) = \sum_{i=0}^{K-1} u_i x^{q^i}$, where $u_i \in \mathbb{F}, i = 0, 1, ..., K-1$, is an the information symbol. The N codeword symbols are $f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_N)$ where the N evaluation points $\{\alpha_1, ..., \alpha_N\}$ are linearly independent over $GF(q)$. From any K independent evaluation points over $GF(q)$, the information can be recovered.

In [2, Construction 1], the (n, k, ℓ, s) codeword is an $\ell \times n$ matrix over $\mathbb{F} = GF(q^{k\ell})$ shown below:

$$\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{\ell,1} & C_{\ell,2} & \cdots & C_{\ell,n} \end{bmatrix}, \quad (2)$$

where each column is a node. Set $K = \ell k - s$. Here, $C_{m,i} \in \mathbb{F}, m \in [\ell], i \in [k]$ are the $K+s$ codeword symbols from a $(K+s, K)$ Gabidulin code, and for each row m, $m \in [\ell]$,

$$[C_{m,k+1}, ..., C_{m,n}] = [C_{m,1}, ..., C_{m,k}] G_{\text{MDS}}, \quad (3)$$

where G_{MDS} is the $k \times (n-k)$ encoding matrix of an (n, k) systematic MDS code over $GF(q)$ that generates the parity.

It is proved in [2, Lemma 2] that t_m symbols in row $m, m \in [\ell]$, is equivalent to evaluations of $f(x)$ with $\sum_{m=1}^{\ell} \min(t_m, k)$ evaluation points that are linearly independent over $GF(q)$. Thus, with any $n-k$ node failures and s symbol failures, we have $t_m \le k$ and

$$\sum_{m=1}^{\ell} \min(t_m, k) = \sum_{m=1}^{\ell} t_m = \ell k - s = K. \quad (4)$$

Then, with the K linearly independent evaluations of $f(x)$, we can decode all the information symbols.

Next, we show how to construct flexible PMDS codes. The main idea is that we divide our code into multiple layers, and each layer applies a similar construction of (2) with a different dimension.

Theorem 1. We can construct an (n, k, ℓ, s) flexible PMDS code over $GF(q^N)$ parameterized by $\{(k_j, \ell_j) : 1 \le j \le a\}$

satisfying (1), assuming there exists an (N, K) Gabidulin code over $GF(q^N)$, $N = \sum_{j=1}^{a} k_j(\ell_j - \ell_{j-1})$, $K = \ell k - s$, and a set of (n, k_j) systematic MDS codes over $GF(q)$.

Proof: The proof is by construction.

Encoding: the codeword is

$$\begin{bmatrix} C_{1,1,1} & C_{1,1,2} & \cdots & C_{1,1,n} \\ C_{1,2,1} & C_{1,2,2} & \cdots & C_{1,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1,\ell_1,1} & C_{1,\ell_1,2} & \cdots & C_{1,\ell_1,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{a,1,1} & C_{a,1,2} & \cdots & C_{a,1,n} \\ C_{a,2,1} & C_{a,2,2} & \cdots & C_{a,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{a,\ell_a-\ell_{a-1},1} & C_{a,\ell_a-\ell_{a-1},2} & \cdots & C_{a,\ell_a-\ell_{a-1},n} \end{bmatrix}, \quad (5)$$

where Layer $j, j \in [a]$ is the $(\ell_{j-1}+1)$-th row to the ℓ_j-th row and $\ell_0 = 0$. Each column is a node.

We first encode the K information symbols using the (N, K) Gabidulin code. Then, we set the first k_j codeword symbols in each row: $C_{j,m_j,i}, j \in [a], m_j \in [\ell_j - \ell_{j-1}], i \in [k_j]$, as the codeword symbols in the (N, K) Gabidulin code. The remaining $n - k_j$ codeword symbols in each row are

$$[C_{j,m_j,k_j+1}, ..., C_{j,m_j,n}] = [C_{j,m_j,1}, ..., C_{j,m_j,k_j}]G_{n,k_j},$$

where G_{n,k_j} is the encoding matrix of the (n, k_j) MDS code over $GF(q)$.

Decoding: For $n - k_J$ failures, we access the first ℓ_J rows (the first J layers) from each node. The code structure in each layer is similar to the general PMDS code in [2, Construction 1], from [2, Lemma 2] we know that for a union of t_{m_j} symbols in Row m_j of Layer j, $j \leq J$, they are equivalent to evaluations of $f(x)$ with $\sum_{j=1}^{J} \sum_{m_j=1}^{\ell_j-\ell_{j-1}} \min(t_{m_j}, k_j)$ linearly independent points over $GF(q)$ in $GF(q^N)$. Thus, with $n - k_J$ node failures and s symbol failures, we have $t_{m_j} \leq k_J \leq k_j$ for $j \in [J]$, and

$$\sum_{j=1}^{J} \sum_{m_j=1}^{\ell_j-\ell_{j-1}} \min(t_{m_j}, k_j) = \sum_{j=1}^{J} \sum_{m_j=1}^{\ell_j-\ell_{j-1}} t_{m_j} = \ell_J k_J - s = K.$$

Then, the information symbols can be decoded from K linearly independent evaluations of $f(x)$. ∎

III. Latency & Reliability Analysis

In this section, we analyze the latency and reliability. We assume each node has a failure probability p, and transmitting one symbol takes t time slots. To simplify the model, we assume the total number of additional symbol failures is smaller than s and it also takes t time slots to transmit a failed symbol. Thus, for a given (n, k, ℓ, s) PMDS code, downloading the required information takes ℓt time slots, and we can downloads all the required information successfully when there are at least k available nodes. We write the probability of at least k out of N nodes are available as:

$$P_{n,k} = \sum_{i=k}^{n} \binom{n}{i} (1-p)^i p^{n-i}. \quad (6)$$

For our (n, k, ℓ, s) flexible PMDS codes with a set $\{(k_j, \ell_j) : 1 \leq j \leq a\}$ satisfying (1), we can transmit the smallest amount per node, i.e., ℓ_j, when the number of failed nodes is less than $n - k_j$. In this case, we have latency $\ell_j t$ with probability $P_{n,k_j} - P_{n,k_{j-1}}$. Thus, we have the latency

$$T = \sum_{j=1}^{a} (P_{n,k_j} - P_{n,k_{j-1}}) \ell_j t, \quad (7)$$

where we set $P_{n,k_0} = 0$. Compared to a fixed (n, k, ℓ, s) PMDS code, we have the same probability $P_{n,k}$ to transmit information successfully while saving the average latency.

TABLE I
An example of $(5, 3, 4, 2)$ flexible PMDS code with $\{(k_1, \ell_1), (k_2, \ell_2)\} = \{(4, 3), (3, 4)\}$. In the case we only have $*$ as failures, we can use the first 4 nodes to decode, each node access the first 3 symbols. In the case both $*$ and $\triangle$ are failures, we can decode from node $1, 3, 4$, each node access 4 symbols.

$C_{1,1,1}$	$\triangle$	$C_{1,1,3}$	$*$	$*$
$C_{1,2,1}$	$\triangle$	$C_{1,2,3}$	$C_{1,2,4}$	$*$
$C_{1,3,1}$	$\triangle$	$*$	$C_{1,3,4}$	$*$
$C_{2,1,1}$	$\triangle$	$C_{2,1,3}$	$C_{2,1,4}$	$*$

TABLE II
The probability of success and average latency for downloading all the required information. we set $n = 5, p = 0.1$ and compare our flexible PMDS code with fixed PMDS codes

	Fixed $(k, \ell) = (4, 3)$	Fixed $(k, \ell) = (3, 4)$	Flexible $\{(k_1, \ell_1), (k_2, \ell_2)\} = \{(4, 3), (3, 4)\}$
Probability of success	91.85%	99.14%	99.14%
Average latency	$3t$	$4t$	$3.05t$

In Table II, we show a comparison of our flexible PMDS code with fixed PMDS codes. For fixed PMDS codes, we can see a tradeoff between the reliability and latency. To reduce the latency from $4t$ to $3t$, we need to increase k from 3 to 4, compromising the system reliability. As a result, the probability of success is reduced from 99.14% to 91.85%. However, our flexible PMDS codes can tolerate failures in the worst case while having the same latency in most of the cases. With parameters $\{(k_1, \ell_1), (k_2, \ell_2)\} = \{(4, 3), (3, 4)\}$, we can achieve 99.14% probability of success while the average latency is only $3.05t$.

References

[1] M. Blaum, J. L. Hafner, and S. Hetzler, "Partial-MDS codes and their application to RAID type of architectures," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4510–4519, 2013.

[2] G. Calis and O. O. Koyluoglu, "A general construction for PMDS codes," *IEEE Communications Letters*, vol. 21, no. 3, pp. 452–455, 2016.