Minimum Storage Rack-Aware Regenerating Codes with Exact Repair and Small Sub-Packetization

Hanxu Hou[†], Patrick P. C. Lee[§], and Yunghsiang S. Han[†]
[†] School of Electrical Engineering & Intelligentization, Dongguan University of Technology
[§] Department of Computer Science and Engineering, The Chinese University of Hong Kong

***Abstract*— Modern data centers often organize storage nodes in racks, in which the cross-rack communication cost is typically much higher than the intra-rack communication cost. Rack-aware regenerating codes have recently been proposed to achieve the optimal trade-off between storage redundancy and cross-rack repair bandwidth, subject to the condition that the original data can be reconstructed from a sufficient number of any non-failed nodes. In this paper, we present a coding framework that transforms any minimum-storage regenerating (MSR) code to a minimum-storage rack-aware regenerating (MSRR) code, such that the cross-rack repair bandwidth is minimized subject to the minimum storage redundancy. To this end, we can construct a family of exact-repair constructions for the MSRR codes for *all admissible parameters*. Furthermore, our constructions achieve low sub-packetization, which is critical for mitigating the I/O overhead during repair.**

***Index Terms*—Data centers, cross-rack repair bandwidth, rack-aware regenerating codes, exact-repair construction**

I. Introduction

Erasure coding is now widely adopted in modern storage systems to provide higher fault tolerance and lower storage redundancy over traditional replication. The most popular erasure codes are Reed-Solomon (RS) codes [1], which encode a data file of k *symbols* (i.e., the units for erasure coding operations) to obtain n symbols over a finite field (where $k < n$), and the n symbols are distributed across n different nodes. RS codes satisfy the *maximum distance separable (MDS)* property, i.e., the data file can be retrieved from any k out of n nodes. However, when a node fails, the conventional repair method of RS codes first reconstructs the data file and encodes it again to reconstruct the lost symbol. This amplifies both network bandwidth and I/O cost.

Regenerating codes (RC) [2] have been proposed to minimize the network bandwidth for repairing a single-node failure subject to a given storage cost. It is shown in [2] that RC achieves the optimal trade-off between the *repair bandwidth* (i.e., the total amount of symbols transferred when repairing a failed node) and the storage redundancy. Two specific RC constructions, namely *minimum storage regenerating (MSR)* codes and *minimum bandwidth regenerating (MBR)* codes, correspond to the minimum storage regeneration point and the minimum bandwidth regeneration point in the optimal trade-off curve, respectively. Follow-up studies [3]–[8] have proposed different constructions of MSR codes and MBR codes.

This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007), Start Fund of Dongguan University of Technology (No. GB200902-19, KCYXM2017025), Research Grants Council of HKSAR (GRF 14216316 and AoE/P-404/18), Innovation and Technology Fund (ITS/315/18FX), the Quality Engineering Project of 2018 DGUT under Grant 201802016, the Quality Engineering Project of 2018 DGUT (No. 201802016), 2019 Guangdong Province Teaching Quality and Teaching Reform (Measurement and Control Technology Teaching Team No. 75).

Modern data centers often have hierarchical topologies by organizing nodes in racks, in which the cross-rack communication cost is much higher than the intra-rack communication cost [9]. While RC achieves the minimum repair bandwidth in flat networks, its *cross-rack repair bandwidth* (i.e., the total amount of symbols transferred across racks during a single-node repair) is sub-optimal in general. This motivates a number of studies that specifically address the repair problem for hierarchical data centers. For example, *Double Regenerating Codes (DRC)* [10], [11] minimize the cross-rack repair bandwidth under the condition that the minimum storage redundancy is achieved (as in RS and MSR codes); *Rack-aware Regenerating Codes (RRC)* [12] generalize DRC and achieve the optimal trade-off between storage redundancy and cross-rack repair bandwidth.

Specifically, RRC [12] partitions n nodes evenly into r racks with n/r nodes each, where n is a multiple of r. It encodes a data file of B symbols into $n\alpha$ symbols that are stored in n nodes with α symbols each. It satisfies the MDS property, i.e., any k out of n nodes can reconstruct the data file. Suppose that a node fails and its α symbols are lost. To repair the α lost symbols, RRC first identifies a new node in the same rack as the failed node (called the *host rack*). The new node first reads $(n/r-1)\alpha$ symbols stored in $n/r-1$ surviving nodes in the host rack, and downloads β symbols from each of the selected d ($d < r$) other racks. The cross-rack repair bandwidth is $d\beta$, the total amount of symbols downloaded from d other racks. The optimal trade-off between the cross-rack repair bandwidth $d\beta$ and the storage redundancy α is:

$$k\alpha + \sum_{\ell=1}^{m} \min\{(d-\ell+1)\beta - \alpha, 0\} \geq B, \tag{1}$$

where $m = \lfloor \frac{kr}{n} \rfloor$. There are two extreme points in the optimal trade-off in Eq. (1), namely the *minimum storage rack-aware regeneration (MSRR)* and *minimum bandwidth rack-aware regeneration (MBRR)* points, which correspond to the minimum storage and the minimum cross-rack repair bandwidth, respectively. The MSRR point corresponds to

$$\alpha = \frac{B}{k}, \beta = \frac{B}{k(d-m+1)}, \tag{2}$$

and the MBRR point corresponds to

$$\alpha = \frac{2Bd}{2kd - m(m-1)}, \beta = \frac{2B}{2kd - m(m-1)}.$$

Note that when $r = n$, MSRR codes and MBRR codes reduce to MSR codes and MBR codes, respectively.

Contributions. We present a general coding framework that can convert MSR codes into MSRR codes with the minimum cross-rack repair bandwidth. Prior studies [3]–[8] have proposed different MSR code constructions. Thus, we can obtain as many MSRR code constructions as the available MSR code constructions through our coding framework. To this end, we can obtain a family of MSRR code constructions for *all admissible parameters* by applying our coding framework to the existing MSR codes [7], [8] that support all admissible parameters. In particular, the MSRR code constructions obtained from our coding framework have a much lower sub-packetization α than existing MSRR codes [13].

The key idea of our coding framework is as follows. Let MSR(n, k, d) denote the MSR codes, such that any k out of n nodes can reconstruct all the data symbols and each node can be repaired by connecting d nodes. Thus, we can repair any one node of the MSR(r, m, d) codes (i.e., any m out of r nodes can reconstruct all data symbols) with the repair bandwidth $\frac{d\alpha}{d-m+1}$, which equals the cross-rack repair bandwidth of MSRR(n, k, r, d) codes. Thus, we can store the $r\alpha$ symbols of the MSR(r, m, d) code in r nodes, in which each node is chosen from each rack so as to maintain the minimum cross-rack repair bandwidth of MSRR(n, k, r, d) codes, while we ensure that the MDS property is satisfied by appropriately constructing the coded symbols.

Related work. There exist several *exact-repair* constructions for MSRR codes [10], [11], [13] and MBRR codes [12] in the literature; by exact-repair, we mean that the lost symbols in the failed node are the same as those repaired in the new node. DRC [10], [11] can be viewed as a specific construction of MSRR codes with some special parameters, while Hou *et al.* [12] present the constructions for MBRR codes for all admissible parameters (n, k, r, d) and MSRR codes for all the parameters that satisfy $\alpha = 1$. Chen and Barg [13] propose a family of MSRR constructions that minimizes the cross-rack repair bandwidth for all admissible parameters, at the expense of incurring a very high level of sub-packetization (i.e., the size of α). Jin *et al.* [14] design a repair scheme for RS codes over some specific finite field such that it also minimizes the cross-rack repair bandwidth of MSRR codes, but the construction cannot support all admissible parameters. Note that when $\frac{kr}{n}$ is an integer, the cross-rack repair bandwidth of MSRR codes is the same as that of MSR codes; in other words, all existing MSR codes can be viewed as MSRR codes if $\frac{kr}{n}$ is an integer. Thus, in this paper, we focus on the exact-repair constructions of MSRR codes when $\frac{kr}{n}$ is *not* an integer.

Some studies focus on different settings on reducing the cross-rack repair bandwidth in rack-based data centers, such as a two-rack setting [15], [16] or a setting based on locally repairable codes [17]. Two closely related studies to RRC are by Sohn *et al.* [18] and Prakash *et al.* [19]. The repair models between RRC and that in [18] are different: RRC allows the encoding of symbols within the same rack during a repair, but not in [18]. For the model in [19], the MDS property is not satisfied; the reason is that repairing a data file needs to retrieve data from a certain number of racks, while the MDS property (i.e., the data file can be retrieved from any k out of n nodes) is not a necessary requirement.

II. Coding Framework

In this section, we present a coding framework that can convert any MSR code construction into the corresponding MSRR code construction.

A. Framework Design

Recall that MSRR codes contain B data symbols, where

$$B = k\alpha = k(d - m + 1)\beta,$$

according to Eq. (2). Also, there are three types of coded symbols: (i) a *global coded symbol*, which is generated by a linear combination of all B data symbols, (ii) a *local coded symbol*, which is generated by a linear combination of the symbols within a rack, and (iii) an *intermediate coded symbol*, which is generated in the middle of operations. Both the global and local coded symbols are finally stored, while the intermediate coded symbols are used for computations and will not be stored.

We first provide a brief overview of the coding framework.

- We compute $(n - r)\alpha$ global coded symbols by encoding all the B data symbols. We select $n/r - 1$ nodes from each of the r racks (i.e., $n - r$ nodes in total). We store the resulting $(n - r)\alpha$ global coded symbols in the selected $n - r$ nodes, each of which stores α global coded symbols.
- We choose $m\alpha$ out of B data symbols, and compute $r\alpha$ intermediate coded symbols by encoding the selected $m\alpha$ data symbols using an MSR(r, m, d) code. We divide the $r\alpha$ intermediate coded symbols into r groups with α symbols each. Each group corresponds to a distinct rack.
- For each group, we compute the α local coded symbols by encoding α intermediate coded symbols and all $(n/r-1)\alpha$ global coded symbols stored in $n/r - 1$ out of n/r nodes of the corresponding rack. The computed α local coded symbols are stored in the remaining node of corresponding rack.

We now elaborate the details of our coding framework. Let

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_B \end{bmatrix}$$

denote B data symbols. We first compute the $(n - r)\alpha$ global coded symbols, denoted by $c_1, c_2, \cdots, c_{(n-r)\alpha}$, as follows:

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_{(n-r)\alpha} \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & \cdots & s_B \end{bmatrix} \mathbf{Q},$$

where $\mathbf{Q}$ is a $B \times (n - r)\alpha$ ($B \geq (n - r)\alpha$) matrix of rank $(n - r)\alpha$. The resulting $(n - r)\alpha$ global coded symbols are then stored in $n - r$ nodes of r racks, among which we select $n/r - 1$ nodes from each of the r racks and each node stores

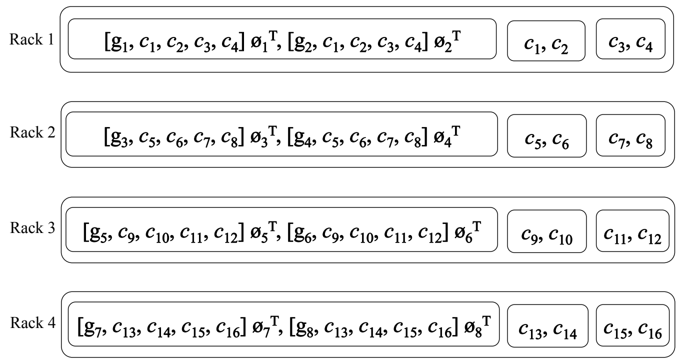
Rack 1
[g1, c1, c2, c3, c4] ø1T, [g2, c1, c2, c3, c4] ø2T
c1, c2
c3, c4
Rack 2
[g3, c5, c6, c7, c8] ø3T, [g4, c5, c6, c7, c8] ø4T
c5, c6
c7, c8
Rack 3
[g5, c9, c10, c11, c12] ø5T, [g6, c9, c10, c11, c12] ø6T
c9, c10
c11, c12
Rack 4
[g7, c13, c14, c15, c16] ø7T, [g8, c13, c14, c15, c16] ø8T
c13, c14
c15, c16

Fig. 1: Example of MSRR code with $(n, k, r, d) = (12, 8, 4, 3)$.

α symbols. Typically, we can choose $\mathbf{Q}$ as a Cauchy matrix or a Vandermonde matrix so that any k out of the $n-r$ nodes are sufficient to reconstruct B data symbols, if $n - r \geq k$.

Let $\mathbf{P}$ be a $m\alpha \times r\alpha$ coefficient matrix (of rank $m\alpha$) for an MSR(r, m, d) code, where the ℓ-th column is denoted by $\mathbf{p}_\ell^T$ for $\ell = 1, 2, \ldots, r\alpha$. We compute $r\alpha$ intermediate coded symbols, denoted by $g_1, g_2, \ldots, g_{r\alpha}$, through the MSR$(r, m, d)$ code as:

$$\begin{bmatrix} g_1 & g_2 & \cdots & g_{r\alpha} \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & \cdots & s_{m\alpha} \end{bmatrix} \mathbf{P}.$$

We divide the generated $r\alpha$ intermediate coded symbols into r groups with α symbols each. For $i = 1, 2, \ldots, r$, group i contains α symbols $g_{(i-1)\alpha+1}, g_{(i-1)\alpha+2}, \ldots, g_{i\alpha}$. We can reconstruct $m\alpha$ data symbols $s_1, s_2, \ldots, s_{m\alpha}$ from any m out of r groups due to the MDS property of the MSR(r, m, d) code. Also, we can reconstruct α intermediate coded symbols in any group by downloading β symbols from each of the d selected groups out of $r-1$ other groups due to the repair procedure of the MSR(r, m, d) code.

Now, let Φ be a $((n/r-1)\alpha+1) \times r\alpha$ matrix, with the i-th column denoted by

$$\phi_i^T = \begin{bmatrix} \phi_{1,i} & \phi_{2,i} & \cdots & \phi_{(n/r-1)\alpha+1,i} \end{bmatrix}^T$$

for $i = 1, 2, \ldots, r\alpha$. For $h = 1, 2, \ldots, r$, we compute α local coded symbols, each of which is a linear combination of the coded symbol $g_{(h-1)\alpha+i}$ and all $(n/r-1)\alpha$ symbols stored in $(n/r-1)$ nodes in rack h, i.e.,

$$\begin{bmatrix} g_{(h-1)\alpha+i} & c_{(h-1)(n/r-1)\alpha+1} & \cdots & c_{h(n/r-1)\alpha} \end{bmatrix} \cdot \phi_{(h-1)\alpha+i}^T$$

with $i = 1, 2, \ldots, \alpha$. The computed α local coded symbols are stored in the remaining node in rack h (recall that the other $n/r-1$ nodes are used to stored the global coded symbols).

Fig. 1 shows an example of $(n, k, r, d) = (12, 8, 4, 3)$ and $\beta = 1$. This implies that $m = \lfloor \frac{kr}{n} \rfloor = 2$, $\alpha = (d-m+1)\beta = 2$, and $B = 16$. The 16 data symbols are $s_1, s_2, \ldots, s_{16}$. First, we compute $(n-r)\alpha = 16$ global coded symbols $c_1, c_2, \ldots, c_{16}$ by multiplying the 16 data symbols and a 16×16 Cauchy matrix, and the global coded symbols will be stored in two nodes for each rack. Then we create an MSR$(r = 4, m = 2, d = 3)$ code by encoding $m\alpha = 4$ data symbols. Here, we use the MSR code in [20], such that we compute the $r\alpha = 8$ intermediate coded symbols as:

$$\begin{bmatrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 & g_8 \end{bmatrix}$$
$$= \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Finally, we compute and store the following $\alpha = 2$ local coded symbols

$$\begin{bmatrix} g_{2h-1} & c_{4(h-1)+1} & c_{4(h-1)+2} & c_{4(h-1)+3} & c_{4h} \end{bmatrix} \cdot \phi_{2h-1}^T,$$
$$\begin{bmatrix} g_{2h} & c_{4(h-1)+1} & c_{4(h-1)+2} & c_{4(h-1)+3} & c_{4h} \end{bmatrix} \cdot \phi_{2h}^T,$$

in the remaining node in rack h with $h = 1, 2, 3, 4$.

B. Repair Method

We now show that we can recover α symbols stored in any node by downloading β symbols from each of other d racks and accessing all other $(n/r-1)\alpha$ symbols in the host rack.

Suppose that a node in rack f fails, where $f \in \{1, 2, \ldots, r\}$. The new node connects to any d helper racks h_i for $i = 1, 2, \ldots, d$. Recall that rack h_i stores the following $\alpha n/r$ symbols:

$$\begin{bmatrix} g_{(h_i-1)\alpha+1} & c_{(h_i-1)(\frac{n}{r}-1)\alpha+1} & \cdots & c_{h_i(\frac{n}{r}-1)\alpha} \end{bmatrix} \cdot \phi_{(h_i-1)\alpha+1}^T,$$
$$\ldots,$$
$$\begin{bmatrix} g_{(h_i-1)\alpha+\alpha} & c_{(h_i-1)(\frac{n}{r}-1)\alpha+1} & \cdots & c_{h_i(\frac{n}{r}-1)\alpha} \end{bmatrix} \cdot \phi_{(h_i-1)\alpha+\alpha}^T,$$
$$\begin{bmatrix} c_{(h_i-1)(\frac{n}{r}-1)\alpha+1} & c_{(h_i-1)(\frac{n}{r}-1)\alpha+2} & \cdots & c_{h_i(\frac{n}{r}-1)\alpha} \end{bmatrix}.$$

Thus, we can access the above $\alpha n/r$ symbols in rack h_i and retrieve the following α symbols

$$g_{(h_i-1)\alpha+1}, g_{(h_i-1)\alpha+2}, \ldots, g_{(h_i-1)\alpha+\alpha}.$$

Recall that

$$\begin{bmatrix} g_1 & g_2 & \cdots & g_{r\alpha} \end{bmatrix}$$

are the codeword of the MSR(r, m, d) code. We can repair the α symbols

$$g_{(f-1)\alpha+1}, g_{(f-1)\alpha+2}, \ldots, g_{(f-1)\alpha+\alpha} \tag{3}$$

by downloading β symbols from each of the d racks $h_1, h_2, \ldots, h_d$ according to the repair method of the MSR(r, m, d) code. Once the α symbols in Eq. (3) are obtained, we recover the α lost symbols in rack f by accessing the other $(n/r-1)\alpha$ symbols stored in rack f and the symbols in Eq. (3). Thus, we can repair any single failed node by downloading β symbols from each of d other racks to the new node, and the cross-rack repair bandwidth is optimal.

C. MDS Property

We first review the Schwartz-Zippel Lemma before showing the MDS property (i.e., the data file can be reconstructed from any k out of n nodes).

Lemma 1. *(Schwartz-Zippel [21]) Let $Q(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a non-zero multivariate polynomial of total* *degree d. Let $r_1, \dots, r_n$ be chosen independently and uniformly at random from a subset $\mathbb{S}$ of $\mathbb{F}_q$. Then*

$$Pr[Q(r_1, \dots, r_n) = 0] \leq \frac{d}{|\mathbb{S}|}. \tag{4}$$

The next theorem shows that the MDS property can be satisfied if the field size is sufficiently large.

Theorem 2. *If the field size is larger than*

$$B \sum_{i=1}^{\min\{k,r\}} \binom{n-r}{k-i}\binom{r}{i}, \tag{5}$$

then any k nodes can recover the B data symbols.

Proof. We need to show that any k out of n nodes can reconstruct B data symbols. Suppose that a data collector connects to k nodes. If each of the chosen k nodes stores global coded symbols, then we can retrieve the B data symbols as any square sub-matrix of a Cauchy matrix is non-singular. Consider that a data collector connects to $k-\ell$ nodes that store global coded symbols and ℓ nodes that store local coded symbols, where $\ell = 1, 2, \dots, \min(k, r)$. The received $B = k\alpha$ symbols can be represented by the $B \times B$ encoding matrix. If we view each entry of $\mathbf{P}$ and Φ as a non-zero variable, we can check that the determinant of the $B \times B$ matrix is a non-zero polynomial with total degree at most B. There are in total

$$\sum_{i=1}^{\min\{k,r\}} \binom{n-r}{k-i}\binom{r}{i} \text{ choices.}$$

The multiplication of all the determinants is a polynomial with total degree at most (5). Thus, we can decode the B data symbols from any k nodes if the field size is larger than the value in Eq. (5) according to the Schwartz-Zippel Lemma. □

Although the upper bound of field size in Theorem 2 is exponential in k, we may directly check by computer search whether any k nodes can reconstruct the B data symbols over a small field. For example, we have checked by computer search that we can always find the matrices $\mathbf{P}$ and Φ such that any k nodes can reconstruct the B data symbols for the case with $(n, k, r, d) = (12, 8, 4, 3)$ when the field size is 41.

D. Example

Continue the example in Fig. 1. We can recover the two symbols in any node by downloading three symbols from other three racks. Suppose that the node that stores local coded symbols in the first rack fails. We need to recover the following two symbols

$$\begin{bmatrix} s_1 & c_1 & c_2 & c_3 & c_4 \end{bmatrix} \cdot \phi_1^T,$$
$$\begin{bmatrix} s_2 & c_1 & c_2 & c_3 & c_4 \end{bmatrix} \cdot \phi_2^T.$$

Recall that the second rack stores the following six symbols

$$g_3\phi_{1,3} + c_5\phi_{2,3} + c_6\phi_{3,3} + c_7\phi_{4,3} + c_8\phi_{5,3},$$
$$g_4\phi_{1,4} + c_5\phi_{2,4} + c_6\phi_{3,4} + c_7\phi_{4,4} + c_8\phi_{5,4},$$
$$c_5, c_6, c_7, c_8.$$

The second rack can first compute

$$\begin{aligned} g_3\phi_{1,3} =& (g_3\phi_{1,3} + c_5\phi_{2,3} + c_6\phi_{3,3} + c_7\phi_{4,3} + c_8\phi_{5,3}) \\ & - (c_5\phi_{2,3} + c_6\phi_{3,3} + c_7\phi_{4,3} + c_8\phi_{5,3}), \\ g_4\phi_{1,4} =& (g_4\phi_{1,4} + c_5\phi_{2,4} + c_6\phi_{3,4} + c_7\phi_{4,4} + c_8\phi_{5,4}) \\ & - (c_5\phi_{2,4} + c_6\phi_{3,4} + c_7\phi_{4,4} + c_8\phi_{5,4}), \end{aligned}$$

and then obtain g_3 and g_4, as $\phi_{1,3} \neq 0$ and $\phi_{1,4} \neq 0$. By the same procedure for rack 3 and rack 4, rack 3 can compute g_5, g_6 and rack 4 can compute g_7, g_8. Finally, rack 2 sends one symbol

$$g_3 + g_4 = s_3 + s_4$$

to the new node, rack 3 sends one symbol

$$g_5 + g_6 = (s_1 + s_3) + (2s_2 + s_4)$$

to the new node, and rack 4 sends one symbol

$$g_7 + g_8 = (2s_1 + s_3) + (2s_2 + s_4)$$

to the new node. We can cancel out the term $s_3 + s_4$ from $(s_1+s_3)+(2s_2+s_4)$ and $(2s_1+s_3)+(2s_2+s_4)$, and then solve for $g_1 = s_1$ and $g_2 = s_2$ from the two linearly independent equations. The new node thus can recover the two symbols

$$g_1\phi_{1,1} + c_1\phi_{2,1} + c_2\phi_{3,1} + c_3\phi_{4,1} + c_4\phi_{5,1}$$
$$g_2\phi_{1,2} + c_1\phi_{2,2} + c_2\phi_{3,2} + c_3\phi_{4,2} + c_4\phi_{5,2}$$

in the failed node from symbols $g_1, g_2, c_1, c_2, c_3, c_4$. The other node can be recovered similarly by downloading three symbols from other three racks.

E. Discussion

Low computational complexity is another practical concern in data centers. One method of reducing computational complexity is replacing the field operations by XOR operations, and the corresponding codes are called *binary MDS array codes*. Although we present the coding framework for MSR codes over a finite field, we can substitute the MSR codes over a finite field by a binary MDS array code with the minimum repair bandwidth. The idea is to replace all the field operations by cyclic shifts and XOR operations to obtain a coding framework that can convert binary MDS array codes with the minimum repair bandwidth into MSRR codes with lower computational complexity. Some existing constructions of binary MDS array codes with the minimum or asymptotically minimum repair bandwidth can be found in previous work [22]–[25].

When the B data symbols are required to be retrieved, we can connect the k nodes from $m+1$ racks. Among the $m+1$ racks, all mn/r nodes in any m racks and $k - mn/r$ nodes which store global coded symbols in the remaining rack are connected. With the above k nodes, the existing efficient decoding method can be employed in the decoding procedure. For example, suppose that the connected k nodes are mn/r nodes from the first racks plus $k - mn/r$ nodes that store global coded symbols from rack $m+1$. For $h = 1, 2, \dots, m$, rack h can access all $\alpha n/r$ symbols and obtain the following $\alpha n/r$ symbols.

$$g_{(h-1)\alpha+1}, g_{(h-1)\alpha+2}, \cdots, g_{(h-1)\alpha+\alpha},$$
$$c_{(h-1)(\frac{n}{r}-1)\alpha+1}, c_{(h-1)(\frac{n}{r}-1)\alpha+2}, \cdots, c_{h(\frac{n}{r}-1)\alpha}.$$

Recall that

$$\begin{bmatrix} g_1 & g_2 & \cdots & g_{r\alpha} \end{bmatrix}$$

are the codeword of the MSR(r, m, d) code. Therefore, we can retrieve all $m\alpha$ data symbols from the symbols

$$\begin{bmatrix} g_1 & g_2 & \cdots & g_{m\alpha} \end{bmatrix},$$

and of course all $r\alpha$ symbols of the MSR(r, m, d) code. In rack $m + 1$, the $(k - mn/r)\alpha$ global coded symbols stored in $k - mn/r$ nodes are

$$c_{((2m+1)(\frac{n}{r})+1-k)\alpha+1}, c_{((2m+1)(\frac{n}{r})+1-k)\alpha+2}, \cdots, c_{(m+1)(\frac{n}{r})\alpha}.$$

Together with $m\alpha(\frac{n}{r} - 1)$ global coded symbols

$$c_1, c_2, \ldots, c_{m\alpha(\frac{n}{r}-1)}$$

stored in the first m racks and the decoded $m\alpha$ data symbols, we can retrieve all other $B - m\alpha$ data symbols by solving the $(B - m\alpha) \times (B - m\alpha)$ linear system, whose encoding matrix is either a Cauchy matrix or a Vandermonde matrix. The existing decoding method for solving the Cauchy linear system [26] or the Vandermonde linear system [27] can be applied to reduce the decoding complexity.

By applying the proposed coding framework, we can convert the existing MSR codes into the corresponding MSRR codes that have the minimum cross-rack repair bandwidth for any node. It is worth mentioning that the low coding rate ($k/n \leq 0.5$) MSR codes, such as product-matrix construction [4] with $n = 2k-1, 2k$, can be converted into high coding rate ($k/n > 0.5$) MSRR codes, as $\frac{kr}{n}$ is not an integer. For example, we convert the MSR($n = 4, k = 2$) code (low coding rate) into an MSRR code with $(n, k, r, d) = (12, 8, 4, 3)$ (high coding rate) in the example in Fig. 1.

III. Comparison

In this section, we compare the sub-packetization and the supported parameters of the proposed MSRR codes by applying the coding framework in Section II with the existing construction of MSRR codes in [10]–[14].

Lemma 3. *The sub-packetization of our MSRR codes by applying the coding framework given in Section II for MSR codes in [4], [6] and [8] is* $d-m+1$, $(d-m+1)^{\frac{r}{d-m+1}}$ *and* $(d-m+1)^{\left\lceil\frac{r}{(d-m+1)\lfloor\frac{r-m-1}{d-m}\rfloor}\right\rceil}$*, respectively. As the d helper nodes in repairing a failed node in [8] are specific, the d helper nodes in the MSRR codes by applying the coding framework for the codes in [8] are also specific.*

Proof. According to the coding framework given in Section II, the sub-packetization of our MSRR codes is the same as that of the employed MSR(r, m, d) codes. Recall that the sub-packetization of MSR(n, k, d) in [4], [6] and [8] is $d -$

TABLE I: Comparison with related work.

	Parameters	Sub-packetization
DRC [10], [11]	$\frac{n}{n-k}$ is integer or $r = 3$	$\alpha = 1$
Codes in [12]	$d = m$	$\alpha = 1$
Codes in [14]	n: prime power	$\alpha = 1$
Codes in [13]	all parameters	$(d - m + 1)^r$
Our codes with [4]	$2m - 2 \leq d$	$d - m + 1$
Our codes with [6]	all parameters	$(d - m + 1)^{\lceil\frac{r}{d-m+1}\rceil}$
Our codes with [8]	all parameters	$(d - m + 1)^{\left\lceil\frac{r}{(d-m+1)\lfloor\frac{r-m-1}{d-m}\rfloor}\right\rceil}$

TABLE II: Sub-packetization of our codes by applying [4] and the codes in [13].

Parameters (n, k, r, d)	(12,8,4,3)	(20,15,5,4)	(35,24,7,6)
α in [13]	16	32	$3^7 = 2187$
α of our codes with [4]	2	2	3

$k + 1$, $(d - k + 1)^{\frac{n}{d-k+1}}$ and $(d - k + 1)^{\left\lceil\frac{n}{(d-k+1)\lfloor\frac{n-k-1}{d-k}\rfloor}\right\rceil}$, respectively. Therefore, The sub-packetization of our MSRR codes by applying the coding framework for MSR codes in [4], [6] and [8] is $d-m+1$, $(d-m+1)^{\frac{r}{d-m+1}}$ and $(d-m+1)^{\left\lceil\frac{r}{(d-m+1)\lfloor\frac{r-m-1}{d-m}\rfloor}\right\rceil}$, respectively. □

Table I shows the comparison with the existing constructions of MSRR codes in terms of supported parameters and sub-packetization. Among the existing constructions, only the codes in [13] can support all the admissible parameters and other constructions only focus on some parameters. The sub-packetization of MSRR codes in [13] is $(d-m+1)^r$, which is much larger than that our MSRR codes by applying the coding framework for MSR codes in [4], [6] and [8]. Table II shows the sub-packetization of the proposed codes and the codes in [13] for some parameters.

IV. Conclusions and Future Work

In this paper, we present a coding framework that can convert any MSR code into MSRR code that has optimal cross-rack repair bandwidth for any single node. By applying the proposed coding framework for the existing constructions of MSR codes, we can obtain many constructions of MSRR codes. Moreover, the proposed MSRR codes have much less sub-packetization compared to the existing MSRR codes in general parameters. In addition to cross-rack repair bandwidth, another important metric in the repair procedure is repair access, defined as the total number of symbols accessed in repairing one single failure node. How to design MSRR codes with small sub-packetization and less repair access is our future work. Another future work is how to fix the tight lower bound of the sub-packetization for MSRR codes. Note that with the coding framework, we can prove the existence of the MSRR code construction, but we do not provide the explicit construction. How to obtain an explicit construction of MSRR codes for all admissible parameters with small sub-packetization and small finite field is also an interesting future work.

References

[1] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[2] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[3] I. Tamo, Z. Wang, and J. Bruck, "Zigzag Codes: MDS Array Codes with Optimal Rebuilding," *IEEE Trans. Information Theory*, vol. 59, no. 3, pp. 1597–1616, 2012.

[4] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction," *IEEE Trans. Information Theory*, vol. 57, no. 8, pp. 5227–5239, August 2011.

[5] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC Codes: Low-Complexity Regenerating Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.

[6] J. Li, X. Tang, and C. Tian, "A Generic Transformation for Optimal Repair Bandwidth and Rebuilding Access in MDS Codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 1623–1627.

[7] M. Ye and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes with Optimal Repair Bandwidth," *IEEE Trans. Information Theory*, vol. 63, no. 4, pp. 2001–2014, 2017.

[8] H. Hou, P. P. Lee, and Y. S. Han, "Multi-Layer Transformed MDS Codes with Optimal Repair Access and Low Sub-Packetization," *arXiv preprint arXiv:1907.08938*, 2019.

[9] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. of ACM SOSP*, 2009.

[10] Y. Hu, P. P. C. Lee, and X. Zhang, "Double Regenerating Codes for Hierarchical Data Centers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 245–249.

[11] Y. Hu, X. Li, M. Zhang, P. P. C. Lee, X. Zhang, P. Zhou, and D. Feng, "Optimal Repair Layering for Erasure-Coded Data Centers: From Theory to Practice," *ACM Transactions on Storage*, vol. 13, no. 4, pp. 33–56, 2017.

[12] H. Hou, P. P. Lee, K. W. Shum, and Y. Hu, "Rack-Aware Regenerating Codes for Data Centers," *IEEE Trans. Information Theory*, vol. 65, no. 8, pp. 4730–4745, Aug. 2019.

[13] Z. Chen and A. Barg, "Explicit Constructions of MSR Codes for Clustered Distributed Storage: The Rack-Aware Storage Model," *Accepted in IEEE Trans. Information Theory*, 2019.

[14] L. Jin, G. Luo, and C. Xing, "Optimal Repairing Schemes for Reed-Solomon Codes with Alphabet Sizes Linear in Lengths under the Rack-Aware Model," *arXiv preprint arXiv:1911.08016*, 2019.

[15] B. Gastón, J. Pujol, and M. Villanueva, "A Realistic Distributed Storage System That Minimizes Data Storage And Repair Bandwidth," in *Proc. IEEE Data Compression Conference*, Snowbird, March 2013.

[16] J. Pernas, C. Yuen, B. Gastón, and J. Pujol, "Non-Homogeneous Two-Rack Model for Distributed Storage Systems," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 1237–1241.

[17] M. A. Tebbi, T. H. Chan, and C. W. Sung, "A Code Design Framework for Multi-Rack Distributed Storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2014, pp. 55–59.

[18] J.-y. Sohn, B. Choi, S. W. Yoon, and J. Moon, "Capacity of Clustered Distributed Storage," *IEEE Trans. Information Theory*, vol. 65, no. 1, pp. 81–107, 2019.

[19] N. Prakash, V. Abdrashitov, and M. Médard, "The Storage versus Repair-Bandwidth Trade-off for Clustered Storage Systems," *IEEE Trans. Information Theory*, vol. 64, no. 8, pp. 5783–5805, August 2018.

[20] Y. Wu and A. G. Dimakis, "Reducing Repair Traffic for Erasure Coding-Based Storage via Interference Alignment," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, July 2009, pp. 2276–2280.

[21] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[22] H. Hou and P. P. Lee, "Binary MDS Array Codes with Optimal Repair," *IEEE Trans. Information Theory*, vol. 66, no. 3, pp. 1405–1422, Mar. 2020.

[23] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, June 2017.

[24] H. Hou, Y. S. Han, P. P. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *IEEE Trans. Information Theory*, vol. 65, no. 11, pp. 7095–7113, 2019.

[25] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-Optimal MDS Array Codes over GF(2)," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 887–891.

[26] H. Hou and Y. S. Han, "A New Construction and an Efficient Decoding Method for Rabin-Like Codes," *IEEE Trans. Communications*, vol. 66, no. 2, pp. 521–533, 2018.

[27] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. Communications*, vol. 66, no. 11, pp. 5053–5066, 2018.