Regenerating Codes and Locally Recoverable Codes for Distributed Storage Systems

Yongjune Kim and Yaoqing Yang

Abstract

We survey the recent results on applying error control coding to distributed storage systems. In such systems, errors often happen in a burst manner, or maybe only one storage node in the whole distributed system fails. Therefore, only a small amount of information, e.g., one data fragment, has to be recovered. Therefore, accessing the whole data is unnecessary. Especially, we focus on regenerating codes and locally recoverable codes, which reduce the communication cost during repairing of the corrupted data.

I. Introduction

In the distributed storage systems, redundancy must be introduced into the systems to improve reliability against node failures since each storage node is individually unreliable. The simplest way is replication of the data in multiple storage nodes. However, error control coding can improve the reliability for the same redundancy compared to replication [1]–[3].

Regenerating codes and locally recoverable codes are technologies to reconstruct a codeword from a partial corruption, such as a single node failure. They have great advantages in distributed storage systems, since in this kind of systems, code corruption often occurs in only a few symbols or in a single node. For example, in order to reconstruct each symbol, the whole codeword has to be transmitted if the conventional maximum distance separable (MDS) codes are used. However, if locally recoverable codes are utilized, the number of symbols that are needed to reconstruct a single symbol is less than the code length.

Y. Kim and Y. Yang are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA (e-mail: yongjunekim@cmu.edu, yyaoqing@andrew.cmu.edu).

Regenerating codes are important for distributed storage systems for the same reason, i.e., it might be practical to design codes regenerating the data in one node by communicating functions of the data in the surviving nodes with minimum communication cost. This serves as the main motivation for us to look at regenerating codes. However, instead of regenerating the data in one node that are exactly the same as the original data, one may generate another data fragment that is constituted by linear combinations of survival data fragments and preserves the ability to regenerate the original data. This property offers a huge flexibility, which yields a quite natural tradeoff between storage and communication cost to regenerate a data fragment.

We use an example to show how regenerating codes work. We divide a message (e.g., file) of size B into k fragments, encode them into n encoded fragments using an (n, k) MDS code, and store them at n nodes. Then, the message can be recovered from any set of k coded fragments. MDS codes such as Reed-Solomon (RS) codes are optimal in terms of the redundancy-reliability tradeoff because k fragments provide the minimum data for recovering the original message of size B [2].

However, the conventional MDS codes such as RS codes are not optimal in terms of repair bandwidth of the distributed storage systems. When an individual node fails, the distributed storage systems should regenerate (i.e., repair) a failed node. A straightforward method of regeneration is to connect to any k nodes, download the entire message, and extract the data that was stored in the failed node. However, downloading the entire message in order to recover the data stored in the failed node is wasteful, and raises the question as to whether there is a better solution. Such a solution is provided by the concept of *regenerating codes* [2]–[4].

II. Regenerating Codes

A. Regenerating Codes

Conventional RS codes treat each fragment stored in a single node as one symbol over the finite field $\mathbb{F}_q$. In contrast, regenerating codes treat each fragment as being comprised of α symbols over the finite field $\mathbb{F}_q$. Linear operations over $\mathbb{F}_q$ permit the transfer of a fraction of the data stored in a particular node [4]. In addition to this new parameter α, two other parameters d and β are associated with regenerating codes.

Under the definition of regenerating codes introduced in [2], a failed node connects to an arbitrary set of d remaining nodes while downloading $\beta \leq \alpha$ symbols from each node. This

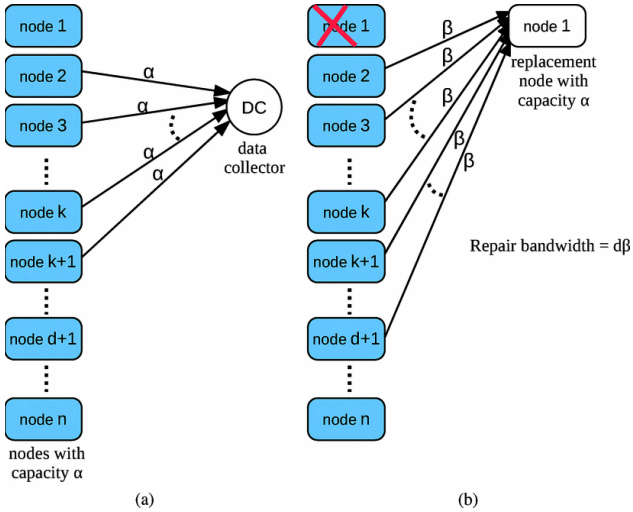

Fig. 1. Two operations of regenerating codes [5]. (a) Reconstruction. (b) Regeneration.

process is termed as *regeneration* and the total amount $\gamma = d\beta$ of data downloaded during regeneration as the *repair bandwidth*. Further, the set of d nodes are termed as *helper nodes*. By using regenerating codes, the repair bandwidth γ can be smaller than the size of the file B, i.e., $\gamma < B$ [2], [4].

The parameter set of regenerating codes over $\mathbb{F}_q$ is given by $\{n, k, d, \alpha, \beta, B\}$. The corresponding codes are called $[n, k, d]$ regenerating codes having a parameter set (α, β, B). Two important operations of $[n, k, d]$ regenerating codes are as follows.

- (Data) Reconstruction: Reconstruct the message of size B symbols by downloading $B = k\alpha$ symbols from any k nodes
- Regeneration (repair): Repair a failed node of size α symbols by downloading $\gamma = d\beta$ symbols from any d nodes among $n - 1$ remaining nodes

These two operations are depicted in Fig. 1 of [5].

B. Tradeoff between Storage and Repair Bandwidth

In [2], the feasible storage-repair bandwidth points were derived by the cut-set bound. The parameters of a regenerating code must satisfy

$$B \leq \sum_{i=0}^{k-1} \{\alpha, (d-i)\beta\}. \tag{1}$$

The feasible points can be explicitly characterized by the following theorem.

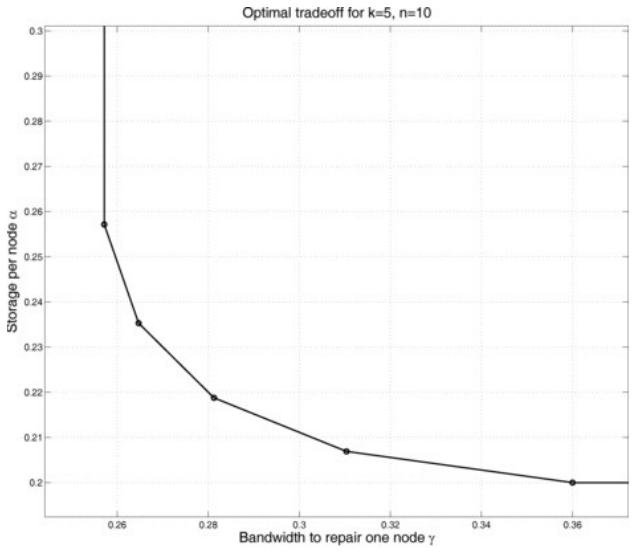

Fig. 2. Optimal tradeoff curve between storage α and repair bandwidth $\gamma = d\beta$ for the $[n = 10, k = 5, d = 9]$ regenerating code [2].

Theorem 1 ([2]): For any $\alpha \geq \alpha^*(n, k, d, \gamma)$, the points $(n, k, d, \alpha, \gamma)$ are feasible, and linear network codes suffice to achieve them. It is information theoretically impossible to achieve points with $\alpha < \alpha^*(n, k, d, \gamma)$. The threshold function $\alpha^*(n, k, d, \gamma)$ is the following:

$$\alpha^*(n,k,d,\gamma) = \begin{cases} \frac{B}{k}, & \gamma \in [f(0), +\infty) \\ \frac{B-g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)) \end{cases} \tag{2}$$

where

$$f(i) = \frac{2Bd}{(2k-i-1)i + 2k(d-k+1)}, \tag{3}$$

$$g(i) = \frac{(2d-2k+i+1)i}{2d}. \tag{4}$$

For the given $[n, k, d]$ regenerating code, it is desirable to minimize both α as well as β since minimizing α results in a minimum storage solution, while minimizing β (for given d) results in a minimum repair bandwidth solution. It is not possible to minimize both α and β simultaneously. Thus there is a tradeoff between α and β as shown in Fig. 2.

There are two extremal points on the optimal tradeoff curve, which correspond to the best storage efficiency and the minimum repair bandwidth, respectively. Theses two extremal points are called the minimum storage regeneration (MSR) and minimum bandwidth regeneration

(MBR) points. The parameters α and β for the MSR point on the tradeoff are given by

$$\begin{aligned} \alpha &= \frac{B}{k}, \\ \beta &= \frac{B}{k(d-k+1)}. \end{aligned} \tag{5}$$

The parameters α and β for the MBR point on the tradeoff are given by

$$\begin{aligned} \alpha &= \frac{2dB}{k(2d-k+1)} = d\beta, \\ \beta &= \frac{2B}{k(2d-k+1)}. \end{aligned} \tag{6}$$

All the other points becides MSR and MBR points are called interior points.

C. Striping of Data [4]

Given an optimal $[n, k, d]$ regenerating code with parameter set (α, β, B), a second optimal regenerating code with parameter set $(\alpha' = \delta\alpha, \beta' = \delta\beta, B' = \delta B)$ for any positive integer δ can be constructed, by dividing the δB message symbols into δ groups of B symbols each, and applying the (α, β, B) code to each group independently.

From (5) and (6), it is true that α and B of the MSR and MBR points are multiples of β. Thus, if we can construct an optimal $[n, k, d]$ regenerating code with $\beta = 1$, then we can construct an optimal $[n, k, d]$ regenerating codes of the MSR and MBR points for any positive integer β. If $\beta = 1$, the values of α and B for the MSR point are given

$$\begin{aligned} \alpha &= d - k + 1 \\ B &= k(d-k+1) = k\alpha. \end{aligned} \tag{7}$$

Also, for the MBR point, the values of α and B are given by

$$\begin{aligned} \alpha &= d \\ B &= kd - \binom{k}{2}. \end{aligned} \tag{8}$$

D. Functional Versus Exact Regeneration

The notion of functional regeneration was introduced in [2]. A failed node is replaced by a node that is functionally equivalent such that the resulting distributed storage system of n nodes must continue to possess the reconstruction and regeneration properties. In the functional

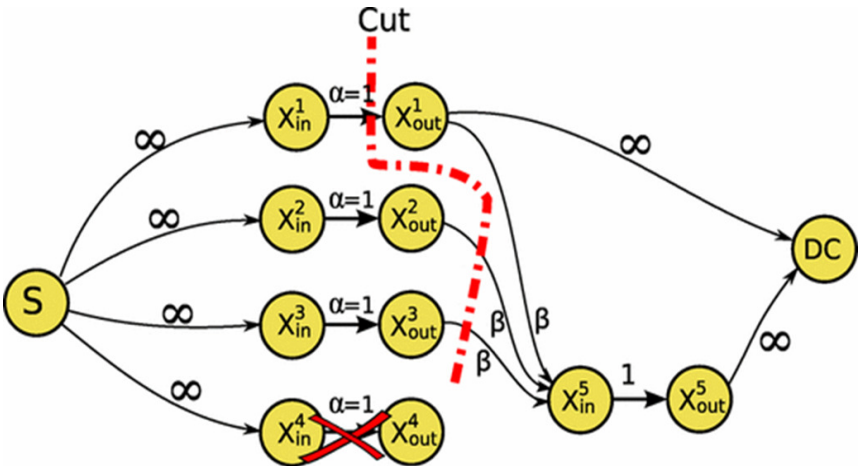

Fig. 3. Intuition on using cut-set bounds to obtain the optimum regenerating bandwidth [2].

regeneration, the data stored at the replacement node may be different from the data stored in the corresponding failed node. This difference may incur the additional communication to inform all nodes of the replacement. Moreover, the reconstruction and regeneration need to be retuned for the new set of coefficients.

On the other hand, by exact regeneration, a replaced node is required to store exactly the same data as was stored in the failed node. Hence, there is no change in the coefficients of a replace node under exact regeneration. This obviates additional communication overheads during the regeneration operation, and also avoids retuning of the reconstruction and regeneration operations. Another advantage of exact regeneration over functional regeneration is the ability to maintain the code in systematic form [4], [5].

Due to these advantages, the exact regeneration codes are of considerable interest. In the next section, we will describe the explicit constructions of exact regenerating codes at the MBR and MSR points, which were proposed in [4].

III. Information Flow Graph and Optimal Regenerating Codes [2]

In [2], the optimal tradeoff between storage and repair bandwidth was derived by the cut-set bound, which was explained in Theorem 1. This optimal tradeoff can be achieve by network coding techniques [2]. In what follows, we briefly review the proof techniques used in [2].

The basic idea can be illustrated using Fig. 3. In this figure, the distributed storage system is modeled as an information flow graph. The source is denoted by the node S and the file

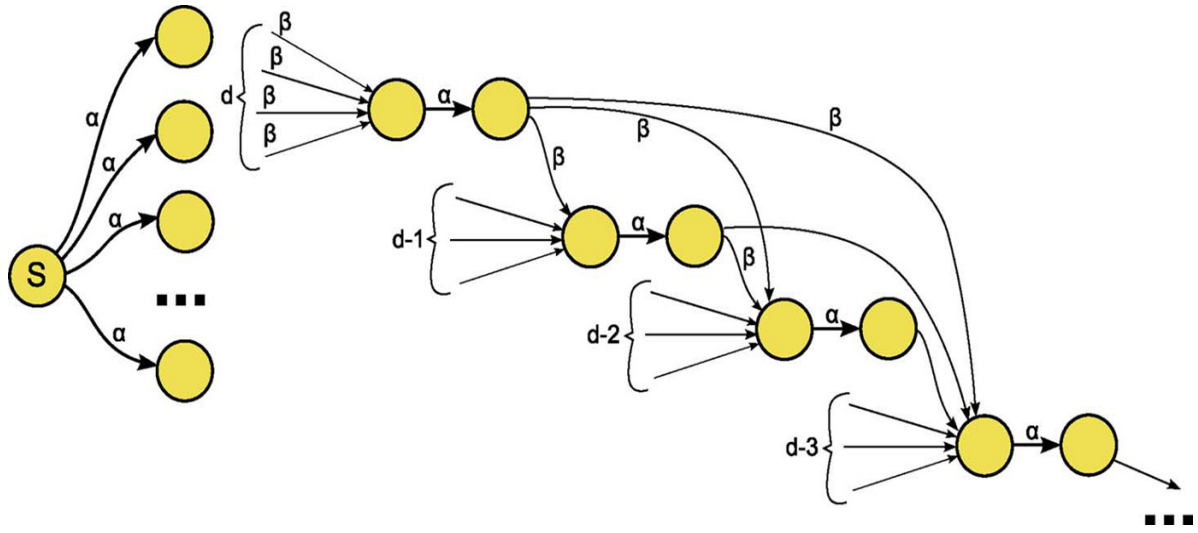

Fig. 4. Example that achieves the min-cut bound in (9) [2].

stored in it has size $B = 2Mb$. The source are connected to $n = 4$ distributed storage nodes and each node has a storage size $\alpha = 1Mb$. Each storage node is partitioned into an in-node and an out-node. The parameter $k = 2$, meaning that from any 2 distributed storage nodes, the original data can be retrieved. The parameter $d = 3$, which means that any node failure can be restored using only data from 3 existing nodes. From each existing node, the data size to be communicated is β. Suppose the fourth node is corrupted, and the fifth node is regenerated. We draw $d = 3$ links from existing nodes to the regenerated node. Each link has capacity β. Finally, we require that all data can be reconstructed in the data collector with infinite capacity links. From the cut-set bound, we know that on the red cut $\alpha+2\beta \geq B$, which means that $\beta \geq 0.5Mb$ and the total repair bandwidth $\gamma = d\beta \geq 1.5Mb$.

Then we discuss how the theorem 1 is proved. In fact, from any node failure patterns, a corresponding information flow graph can be constructed. Therefore, we would like to calculate the minimum sum-capacity of all min-cuts in all possible information flows. The main step of the proof is to show that the minimum min-cut value among all possible information flow graphs can be written as follows

$$C = \sum_{e \in S} c_e = \sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}. \tag{9}$$

There is an explicit example where the min-cut value is achieved, which is shown in Fig. 4. In this figure, the first regenerated node is connected to the last d distributed storage nodes. The

second regenerated node is connected to the last $d-1$ original nodes and the first regenerated nodes. In general, the i-th new node is connected to the last $d+1-i$ original nodes and the newly generated $i-1$ nodes. There is a cut, which either cuts between the i-th in-node and the i-th out-node, or cuts between the i-th in-node and the old nodes. The choice depends on which one is smaller. Clearly, this cut yields the expression in (9).

The remaining thing is to prove that for any cut on any feasible information flow graph, the sum-capacity is always greater all equal to the RHS of (9). It can be proved by defining a topological order of all the nodes on the right part of the examined cut. That is, the index i of one node is always smaller than the index j of another node, if there is a directed link from the out-node of node with index i to the in-node of the node with index j. Then, we examine the local cut that divides the topologically first node to the right part of the examined cut. This local cut either has value α or has value $d\beta$, because the d existing nodes that are connected to this topologically first node must all be on the left part of the examined cut, otherwise the topologically first node has a father-node on the right part. The same reasoning can be carried on, and the expression on the RHS of (9) can be obtained.

IV. Explicit Constructions of Exact Regenerating Codes [4]

A. *Product-Matrix Framework*

In [4], the product-matrix framework was proposed. Under this framework, each codeword in the distributed storage systems can be represented by an $(n \times \alpha)$ code matrix C whose ith row $\underline{c}_i^t$ contains the α symbols stored by the ith node. Note that t denotes the transpose. Each code matrix is the product

$$C = \Psi M \tag{10}$$

of an $(n \times d)$ encoding matrix Ψ and an $(d \times \alpha)$ message matrix M. The entries of the matrix Ψ are fixed a priori and independent of the message symbols. The message matrix M contains the B message symbols, with some symbols possibly repeated. We will refer to the ith row $\underline{\psi}_i^t$ of Ψ as the encoding vector of node i since it is used to encode the message into the fragment

$$\underline{c}_i^t = \underline{\psi}_i^t M \tag{11}$$

which is stored in the ith node.

Reconstruction amounts to recovering the message matrix M from the $k\alpha$ symbols obtained from an arbitrary set of k storage nodes. Let us denote the set of k nodes to which the data-collector (DC) connects as $\{i_1, \ldots, i_k\}$. The jth node in this set passes on the vector $\underline{\psi}_{i_j}^t M$ to the DC. Hence, the DC obtains the product matrix

$$\Psi_{\text{DC}} M \tag{12}$$

where Ψ_{DC} is the submatrix of Ψ consisting of the k rows $\left\{\underline{\psi}_{i_1}^t, \ldots, \underline{\psi}_{i_k}^t\right\}$. By using the properties of the matrices Ψ_{DC} and M, we can recover the message matrix M and obtain the B message symbols.

To regenerate a failed node f, the node replacing the failed node connects to an arbitrary subset $\{h_1, \ldots, h_d\}$ of d helper nodes. Each helper node passes on the inner product of the α symbols stored in it with a vector $\underline{\mu}_f$ of length α to the replacement node. Thus, the helper node h_j passes

$$\underline{\psi}_{h_j}^t M \underline{\mu}_f. \tag{13}$$

The replacement node obtains the product matrix

$$\Psi_{\text{repair}} M \underline{\mu}_f \tag{14}$$

where Ψ_{repair} is the submatrix of Ψ consisting of the d rows $\left\{\underline{\psi}_{h_1}^t, \ldots, \underline{\psi}_{h_d}^t\right\}$. By using the properties of the matrices Ψ_{repair} and M, we can repair (regenerate) the α symbols in the failed node.

B. Product-Matrix MBR Code Construction

We can construct any $[n, k, d]$ regenerating codes for the MBR point (i.e., MBR codes) if all n, k, and d satisfy $k \leq d \leq n-1$. B of (8) can be rewritten in the following form.

$$B = \binom{k+1}{2} + k(d-k) \tag{15}$$

Then the parameter set of the $[n, k, d]$ MBR code is given by

$$\left(\alpha = d, \beta = 1, B = \binom{k+1}{2} + k(d-k)\right). \tag{16}$$

Let S be a $(k \times k)$ matrix constructed so that the $\binom{k+1}{2}$ entries in the upper-triangular half of the matrix are filled up by $\binom{k+1}{2}$ distinct message symbols drawn from the set $\{u_i\}_{i=1}^{B}$. The $\binom{k}{2}$

entries in the strictly lower-triangular portion of the matrix are then chosen so as to make the matrix S a symmetric matrix. The remaining $k(d-k)$ message symbols are used to fill up a second $(k \times (d-k))$ matrix T. The message matrix M is then defined as the $(d \times d)$ symmetric matrix given by

$$M = \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix} \tag{17}$$

The symmetry of the message matrix will be found to be instrumental during regeneration and reconstruction. Next, define the encoding matrix Ψ to be any $(n \times d)$ matrix of the form

$$\Psi = [\Phi \quad \Delta] \tag{18}$$

where Φ and Δ are $(n \times k)$ and $(n \times (d-k))$ matrices respectively, chosen in such a way that:

1) Any d rows of Ψ are linearly independent;
2) Any k rows of Φ are linearly independent.

The above requirements can be met, for example, by choosing Ψ to be either a Cauchy or a Vandermonde matrix. The only constraint on the field size comes from the above required properties of the encoding matrix Ψ. For instance, when Ψ is chosen as a Vandermonde matrix, any field of size n or higher suffices.

The following two theorems show that the code presented is an $[n, k, d]$ MBR code by establishing the exact-regeneration and data-reconstruction properties respectively.

Theorem 2 (MBR Exact-Regeneration [4]): In the code presented, exact-regeneration of any failed node can be achieved by downloading one symbol each from any d of the $(n-1)$ remaining nodes.

Proof: Let $\underline{\psi}_f^t$ be the row of Ψ corresponding to the failed node f. Thus the $\alpha(=d)$ symbols stored in the failed node correspond to the vector

$$\underline{\psi}_f^t M. \tag{19}$$

The replacement for the failed node f connects to an arbitrary set $\{h_j \mid j = 1, \ldots, d\}$ of d helper nodes. Upon being contacted by the replacement node, the helper node h_j computes the inner product

$$\underline{\psi}_{h_j}^t M \underline{\psi}_f \tag{20}$$

and passes on this value to the replacement node. Thus, in the present construction, the vector $\underline{\mu}_f$ in (13) equals $\underline{\psi}_f$. Thus, the replacement node obtains the d symbols $\Psi_{\text{repair}} M \underline{\psi}_f$ from the d helper nodes, where

$$\Psi_{\text{repair}} = \begin{bmatrix} \underline{\psi}_{h_1}^t \\ \vdots \\ \underline{\psi}_{h_d}^t \end{bmatrix}. \tag{21}$$

By construction, the $(d \times d)$ matrix Ψ_{repair} is invertible. Hence, the replacement node recovers $M\underline{\psi}_f$ through multiplication on the left by $\Psi_{\text{repair}}^{-1}$. Since M is symmetric, we can obtain

$$\left(M\underline{\psi}_f\right)^t = \underline{\psi}_f^t M \tag{22}$$

which is precisely the data previously stored in the failed node. ■

Theorem 3 (MBR Data-Reconstruction [4]): In the code presented, all the B message symbols can be recovered by connecting to any k nodes, i.e., the message symbols can be recovered through linear operations on the entries of any k rows of the matrix C.

Proof: Let

$$\Psi_{\text{DC}} = [\Phi_{\text{DC}} \quad \Delta_{\text{DC}}] \tag{23}$$

be the $(k \times d)$ submatrix of Ψ, corresponding to the k rows of Ψ to which the DC connects. Thus, the DC has access to the symbols

$$\Psi_{\text{DC}} M = \left[\Phi_{\text{DC}} S + \Delta_{\text{DC}} T^t \quad \Phi_{\text{DC}} S\right]. \tag{24}$$

By construction, Ψ_{DC} is invertible. Hence, by multiplying the matrix $\Psi_{\text{DC}} M$ on the left by Ψ_{DC}^{-1}, one can recover first T and subsequently, S. ■

C. Product-Matrix MSR Code Construction

We can construct any $[n, k, d \geq 2k-2]$ regenerating codes for the MSR point (i.e., MSR codes). We begin by constructing an MSR code for $d = 2k-2$ and can extend it to MSR codes such that $d > 2k-1$ [4].

At the MSR point with $d = 2k-2$, we have

$$\alpha = d - k + 1 = k - 1 \tag{25}$$

and hence

$$d = 2\alpha. \tag{26}$$

Also,

$$B = k\alpha = \alpha(\alpha + 1). \tag{27}$$

We define the $(d \times \alpha)$ message matrix M as

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \tag{28}$$

where S_1 and S_2 are $(\alpha \times \alpha)$ symmetric matrices constructed such that the $\binom{\alpha+1}{2}$ entries in the upper-triangular part of each of the two matrices are filled up by $\binom{\alpha+1}{2}$ distinct message symbols. Thus, all the $B = \alpha(\alpha+1)$ message symbols are contained in the two matrices S_1 and S_2. The entries in the strictly lower-triangular portion of the two matrices S_1 and S_2 are chosen so as to make these matrices symmetric.

Next, we define the encoding matrix Ψ to be the $(n \times d)$ matrix given by

$$\Psi = [\Phi \quad \Lambda\Phi] \tag{29}$$

where Φ is an $(n \times \alpha)$ matrix and Λ is an $(n \times n)$ diagonal matrix. The elements of Ψ are chosen such that the following conditions are satisfied:

1) Any d rows of Ψ are linearly independent;
2) Any α rows of Φ are linearly independent;
3) The n diagonal elements of Λ are distinct.

The above requirements can be met, for example, by choosing Ψ to be a Vandermonde matrix with elements chosen carefully to satisfy the third condition.

Theorem 4 (MSR Exact-Regeneration [4]): In the code presented, exact-regeneration of any failed node can be achieved by downloading one symbol each from any $d = 2k-2$ of the $(n-1)$ remaining nodes.

Proof: Let $\left[\underline{\phi}_f^t \quad \lambda_f \underline{\phi}_f^t\right]$ be the row of Ψ corresponding to the failed node f. The α symbols stored in the failed node were

$$\left[\underline{\phi}_f^t \quad \lambda_f \underline{\phi}_f^t\right] M = \underline{\phi}_f^t S_1 + \lambda_f \underline{\phi}_f^t S_2. \tag{30}$$

The replacement for the failed node f connects to an arbitrary set $\{h_j \mid j = 1, \ldots, d\}$ of d helper nodes. Upon being contacted by the replacement node, the helper node h_j computes the inner product $\underline{\psi}_{h_j}^t M \underline{\phi}_f$ and passes on this value to the replacement node. Note that the vector $\underline{\mu}_f$ in (13) equals $\underline{\phi}_f$. The replacement node obtains the d symbols $\Psi_{\text{repair}} M \underline{\phi}_f$ from the d helper nodes, where

$$\Psi_{\text{repair}} = \begin{bmatrix} \underline{\psi}_{h_1}^t \\ \vdots \\ \underline{\psi}_{h_d}^t \end{bmatrix}. \tag{31}$$

By construction, the $(d \times d)$ matrix Ψ_{repair} is invertible. Thus, the replacement node knows

$$M\underline{\phi}_f = \begin{bmatrix} S_1 \underline{\phi}_f \\ S_2 \underline{\phi}_f \end{bmatrix}. \tag{32}$$

As S_1 and S_2 are symmetric matrices, the replacement node has thus acquired through transposition, both $\underline{\phi}_f^t S_1$ and $\underline{\phi}_f^t S_2$. Using these, it can obtain (30) which is precisely the data previously stored in the failed node. ■

Theorem 5 (MSR Data-Reconstruction [4]): In the code presented, all the B message symbols can be recovered by connecting to any k nodes, i.e., the message symbols can be recovered through linear operations on the entries of any k rows of the code matrix C.

Proof: Let

$$\Psi_{\text{DC}} = [\Phi_{\text{DC}} \quad \Lambda_{\text{DC}} \Phi_{\text{DC}}] \tag{33}$$

be the $(k \times d)$ submatrix of Ψ, containing the k rows of Ψ which correspond to the k nodes to which the DC connects. Hence, the DC obtains the symbols

$$\begin{aligned} \Psi_{\text{DC}} M &= [\Phi_{\text{DC}} \quad \Lambda_{\text{DC}} \Phi_{\text{DC}}] \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \\ &= [\Phi_{\text{DC}} S_1 + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2]. \end{aligned} \tag{34}$$

The DC can post-multiply this term with Φ_{DC}^t to obtain

$$[\Phi_{\text{DC}} S_1 + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2] \, \Phi_{\text{DC}}^t = \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^t + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^t. \tag{35}$$

Next, let the symmetric matrices P and Q be defined as

$$P = \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^t \tag{36}$$

$$Q = \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^t. \tag{37}$$

In terms of P and Q, the DC has access to the symbols of the matrix $P + \Lambda_{\text{DC}} Q$.

The (i, j) element of this matrix (for $1 \leq i, j \leq k$) is

$$P_{i,j} + \lambda_i Q_{i,j}, \tag{38}$$

while the (j, i) element is given by

$$P_{j,i} + \lambda_j Q_{j,k} = P_{i,j} + \lambda_j Q_{i,j}, \tag{39}$$

since P and Q are symmetric. Note that $\lambda_i \neq \lambda_j$ by construction of the encoding matrix Ψ. Thus, the DC can solve for the values of $P_{i,j}$ and $Q_{i,j}$ for all $i \neq j$.

Consider the matrix P. Let Φ_{DC} be given by

$$\Phi_{\text{DC}} = \begin{bmatrix} \underline{\phi}_1^t \\ \vdots \\ \underline{\phi}_{\alpha+1}^t \end{bmatrix}. \tag{40}$$

All the nondiagonal elements of P are known. The elements in the ith row (excluding the diagonal element) are given by

$$\underline{\phi}_i^t S_1 \left[\underline{\phi}_1 \ \cdots \ \underline{\phi}_{i-1} \ \underline{\phi}_{i+1} \ \cdots \ \underline{\phi}_{\alpha+1} \right]. \tag{41}$$

Since the matrix to the right is invertible by construction, the DC can obtain

$$\left\{ \underline{\phi}_i^t S_1 \mid 1 \leq i \leq \alpha + 1 \right\}. \tag{42}$$

Selecting the first α of these, the DC has access to

$$\begin{bmatrix} \underline{\phi}_1^t \\ \vdots \\ \underline{\phi}_{\alpha}^t \end{bmatrix} S_1. \tag{43}$$

The matrix on the left is also invertible by construction and hence the DC can recover S_1. Similarly, using the values of the nondiagonal elements of Q, the DC can recover S_2. ■

V. Locally Recoverable Codes

Suppose the message vector is $\mathbf{m}$ and the corresponding codeword is $\mathbf{x}$. When the code is linear, it holds that $\mathbf{x}^T = \mathbf{m}^T \cdot \mathbf{G}$, where $\mathbf{G}$ is the generator matrix. We follow the notations of [6] to define a locally recoverable code.

Definition 6: An (n, r, d, M, α)-LRC (locally recoverable code) is a code that satisfies the following conditions:

- The message to be encoded has size M bits. The code length is n and each code symbol has α bits;
- Any coded symbol can be reconstructed by accessing at most r symbols.
- The code distance is d.

The second property is often called the *locality* of the code. Clearly, the lower the locality of the code is, the easier it is for the distributed storage system to reconstruct from single data corruption.

There is another parameter called information locality to characterize the local recovering ability of a code, which is quite similar to the locality. Suppose the generator matrix of the code can be written as

$$\mathbf{G} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n], \tag{44}$$

where each $\mathbf{c}_i$ is a column vector. Each $\mathbf{c}_i$ has a corresponding code symbol because the code symbol x_i When a code has information locality r, there exist a sub-matrix $\mathbf{G}_I$ of $\mathbf{G}$ such that $\mathbf{G}_I$ is full rank and the corresponding codeword symbols all have locality r. If a code has information locality r, all code symbols can be divided into two sets S_1 and S_2, $S_1 \cup S_2 = \{1, 2, \ldots, n\}$, such that each code symbol in S_1 can be reconstructed from r symbols and each code symbol in S_2 can be reconstructed from symbols in S_1. The information locality is especially useful when the code is a systematic one and information symbols are more frequently updated than parity symbols.

A. Tradeoff between information locality and minimum distance

A breakthrough in the local decoding problem comes from [7], where a special class of linear codes called Pyramid codes is invented. A Pyramid code is obtained in the following way. First

pick an arbitrary linear $(k+d-1,k,d)$ MDS code $\mathcal{C}_0$, i.e., a code that achieves the Singleton bound. Suppose the encoding map can be written as

$$\mathcal{E}_0(\mathbf{x}) = \left(\mathbf{x}^T, \mathbf{x}^T\mathbf{p}_0, \mathbf{x}^T\mathbf{p}_1, \ldots, \mathbf{x}^T\mathbf{p}_{d-2}\right), \tag{45}$$

where the generator matrix

$$\mathbf{G}_0 = [\mathbf{I}, \mathbf{P}_0] = [\mathbf{I}, \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{d-2}]. \tag{46}$$

Then, we partition the the message bit index set $\{1,2,\ldots,k\}$ into $t = \left\lceil \frac{k}{r} \right\rceil$ disjoint subsets $\{1,2,\ldots,k\} = \bigcup_{i=1}^{t} S_i$, with each set S_i containing at most r elements. By $\mathbf{x}|_S$ we denote the $|S|$ dimensional restriction of $\mathbf{x}$ to coordinates in the set S which is a subset of $\{1,2,\ldots,k\}$. The Pyramid code is finally defined to be the code that has the encoding map

$$\mathcal{E}(\mathbf{x}) = \left(\mathbf{x}^T, \mathbf{x}^T|_{S_1} \cdot \mathbf{p}_0|_{S_1}, \mathbf{x}^T|_{S_2} \cdot \mathbf{p}_0|_{S_2}, \ldots \mathbf{x}^T|_{S_t} \cdot \mathbf{p}_0|_{S_t}, \mathbf{x}^T\mathbf{p}_1, \ldots, \mathbf{x}^T\mathbf{p}_{d-2}\right). \tag{47}$$

Note that an MDS code has the property that the vector $\mathbf{p}_0$ has full Hamming weight. Therefore, the restriction of $\mathbf{p}_0$ to each subset S_i makes the corresponding coordinates in S_i able to be locally decoded with locality r. Since this construction only captures the locality of k information symbols, we conclude that Pyramid codes with $t = \left\lceil \frac{k}{r} \right\rceil$ have information locality r. Another way to look at the locality of Pyramid code is from the parity check matrix. Since the original MDS code $\mathcal{C}_0$ has the generator matrix structure shown in (46), the generator matrix $\mathbf{G}$ of the constructed Pyramid code can be written as

$$\mathbf{G} = [\mathbf{I}, \mathbf{p}_0 \circ \mathbf{e}_{S_1}, \mathbf{p}_0 \circ \mathbf{e}_{S_2} \ldots, \mathbf{p}_0 \circ \mathbf{e}_{S_t}, \mathbf{p}_1, \ldots, \mathbf{p}_{d-2}], \tag{48}$$

where $\circ$ denotes the Hadamard product and $\mathbf{e}_{S_i}$ denote the k-bit vector that has support S_i. Since we know that $\mathbf{G} = [\mathbf{I}, \mathbf{P}]$ has parity check matrix $\mathbf{H} = [\mathbf{P}^T, \mathbf{I}]$, the corresponding parity check matrix can be determined from (46). The information locality of the Pyramid code can also be examined directly from the parity check matrix.

The information locality r and the minimum distance d of an (n,k) Pyramid code satisfies the following constraint:

$$n - k = \left\lceil \frac{k}{r} \right\rceil + d - 2. \tag{49}$$

It is proved later in [8] that the best obtainable information locality-minimum distance tradeoff for a (n,k,d) linear code is

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{50}$$

Therefore, the Pyramid code obtains the best information locality-minimum distance tradeoff. The proof of (49) in [8] is based on the following fact: for a given linear code with generator matrix $\mathcal{G}$, for any range space spanned by columns of $\mathcal{G}$ with rank smaller or equal to $k-1$, the number of columns used to span this range space must be smaller or equal to $n-d$. Based on this column, one can upper bound the number of selected columns in $\mathcal{G}$ with $n-d$, while a specific selection procedure results in a lower bound of the number of selected columns, which is a function of the information locality. Using this technique, one can obtain the upper bound of d for a given information locality r.

In Theorem 7 of [8], an even stronger result is given for the case that $r|k$. It is proved that under such condition, the code that attains the maximum minimum distance in (49) must satisfy the property that the k information symbols in the systematic code (note that information locality is always defined for a linear code with the first k code symbols being able to be locally decoded) are divided into several disjoint subsets, where each set corresponds one linear dependence of the information symbols in it and some other parity check symbols. That is to say, at least for the case when $r|k$, the Pyramid code, although it seems simple, is actually the only way to achieve the best distance under a locality constraint.

B. Tradeoff between locality and code rate

People have obtained the tradeoff between locality and code rate under extreme rates that are close to the Shannon's limit. For example, a code construction based on grouping code symbols is given in [9] to get a good locality-rate tradeoff. First, similar to the construction of Pyramid codes, a powerful code $\mathcal{C}_0$ is chosen. The code $\mathcal{C}_0$ is designed to asymptotically achieve the random coding bound, i.e., it is a linear code with length m and rate $1-H(p)-\epsilon$ that has probability of incorrect decoding at most $2^{-E(p,\epsilon)m}$. The best error exponent is usually called the random coding exponent and is detailed in many text books, such as [10]. Denote by $\mathbf{G}_0$ the generator matrix of the code $\mathcal{C}_0$. Then, one can construct a large code with length n satisfying

$$m = (1+\alpha)/E(p,\epsilon)\log n, \tag{51}$$

where α is a constant. We simply assume that n/m is an integer, then the generator matrix of the constructed code is

$$\mathbf{G} = \mathbf{I}_{n/m} \otimes \mathbf{G}_0. \tag{52}$$

Clearly, this code is encoded by grouping the code bits into groups of $\log n$ and encoding each group with the powerful linear code $\mathcal{C}_0$. This construction preserves the good code rate $1 - H(p) - \epsilon$ and yields a code locality in the order of $\Theta(\log n)$. But the bonus point is that by using the bounded error probability $2^{-E(p,\epsilon)m}$ of each group and the union bound, one can show that the total error probability of this code approximately in the order of $O(\frac{1}{n^{\alpha}})$, which means that this can even achieve acceptable error correction performance when the rate is approaching the channel capacity.

An impossibility result is also given in [9], stating that for any code $\mathcal{C}$ with code length n and rate $1 - p - \epsilon$ that achieves probability of error bounded away from 1, when used over BEC(p) channel, the locality of $\mathcal{C}$ is at least $c \log 1/\epsilon$, where c is a constant. This result shows that the LDGM code construction in [9] attains the best locality when being transmitted at a rate that approaches the channel capacity.

VI. Open Problems

In regenerating Codes, for the two extremal points such as MSR and MBR points, the explicit code constructions of exact regeneration were proposed in [4]. Thus, we can claim that the optimal tradeoff of functional regeneration of MSR and MBR points is identical to that of exact regeneration. However, we do not know the optimal tradeoff of exact regeneration codes for interior points.

Recently, it was shown that there exists a non-vanishing gap between the optimal tradeoff of the functional regeneration codes and that of the exact regeneration codes [11]. The optimal tradeoff between storage and repair bandwidth of the interior points of exact regeneration is an important open problem.

Also, the code construction for MSR point was proposed for $d \geq 2k - 2$ [4]. Since $2k - 2 \leq d \leq n - 1$,

$$\frac{k}{n} \leq \frac{k}{2k-1} \approx \frac{1}{2}. \tag{53}$$

The explicit code construction of MSR code for high rate is not known, which is a practically important open problem. In [12], it was shown that we can not achieve the high rate MSR code if $\beta = 1$.

REFERENCES

[1] J. S. Plank, "Erasure codes for storage systems: A brief primer," *Usenix Magazine*, vol. 38, no. 6, 2013.

[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[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. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.

[5] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, 2012.

[6] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2771–2775.

[7] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. Sixth IEEE Int. Symp. Network Computing and Applications (NCA)*, Cambridge, MA, 2007, pp. 79–86.

[8] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[9] A. Mazumdar, V. Chandar, and G. W. Wornell, "Update-efficiency and local repairability limits for capacity approaching codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 976–988, May 2014.

[10] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[11] C. Tian, "Characterizing the Rate Region of the (4,3,3) Exact-Repair Regenerating Codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 967–975, 2014.

[12] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Cairo, Egypt, 2010, pp. 1–5.