Repairable Block Failure Resilient Codes

Gokhan Calis and O. Ozan Koyluoglu
Department of Electrical and Computer Engineering
The University of Arizona
Email: {gcalis, ozan}@email.arizona.edu

***Abstract*—In large scale distributed storage systems (DSS) deployed in cloud computing, correlated failures resulting in simultaneous failure (or, unavailability) of blocks of nodes are common. In such scenarios, the stored data or a content of a failed node can only be reconstructed from the available live nodes belonging to available blocks. To analyze the resilience of the system against such block failures, this work introduces the framework of Block Failure Resilient (BFR) codes, wherein the data (e.g., file in DSS) can be decoded by reading out from a same number of codeword symbols (nodes) from each available blocks of the underlying codeword. Further, repairable BFR codes are introduced, wherein any codeword symbol in a failed block can be repaired by contacting to remaining blocks in the system. Motivated from regenerating codes, file size bounds for repairable BFR codes are derived, trade-off between per node storage and repair bandwidth is analyzed, and BFR-MSR and BFR-MBR points are derived. Explicit codes achieving these two operating points for a wide set of parameters are constructed by utilizing combinatorial designs, wherein the codewords of the underlying outer codes are distributed to BFR codeword symbols according to projective planes.**

I. Introduction

Increasing demand for storing and analyzing *big-data* as well as several applications of cloud computing systems require efficient cloud computing infrastructures. One inevitable nature of the storage systems is node failures. In order to provide resilience against failures, redundancy is introduced in the storage. Classical redundancy schemes range from *replication* to *erasure coding*. Erasure coding allows for better performance in terms of reliability and redundancy compared to replication, however repair bandwidth in reconstructing a failed node is higher. Regenerating codes are proposed to overcome this problem in the seminal work of Dimakis et al. [1]. In such a model of distributed storage systems (DSS), the file $\mathcal{M}$ is encoded to n nodes such that any $k \leq n$ nodes (each with α symbols) allow for reconstructing the file and any $d \geq k$ nodes (with $\beta \leq \alpha$ symbols from each) reconstruct a failed node with a repair bandwidth $\gamma = d\beta$. The trade-off between per node storage (α) and repair bandwidth (γ) is characterized and two ends of the trade-off are named as minimum storage regenerating (MSR) and minimum bandwidth regenerating (MBR) points [1]. Several explicit codes have been proposed to achieve these points recently [2]–[4]. Another metric for an efficient repair is repair degree d, and regenerating codes necessarily have $d \geq k$. Codes with locality and locally repairable codes with regeneration properties [5]–[10] allow for a small repair degree, wherein failed nodes are reconstructed via local connections. Instances of such codes are recently considered in DSS [11], [12].

In large-scale distributed storage systems (such as GFS [13]), *correlated failures* are unavoidable. As analyzed in [14], these simultaneous failures of multiple nodes affect the performance of computing systems severely. The analysis in [14] further shows that these correlated failures arise due to *failure domains*. For example, nodes connected to the same power source or nodes belonging to the same rack exhibit these failure bursts. The unavailability periods are transient, and largest failure bursts almost always have significant rack-correlation. In order to overcome from failures having such patterns, a different approach is needed.

In this paper, we develop a framework to analyze resilience against block failures in DSS with node repair efficiencies. We consider a DSS with a single failure domain, where nodes belonging to the same failure group constitute a block of the codeword. We introduce block failure resilient (BFR) codes, which allow for data collection from any $b_c = b - \rho$ blocks, where b is the number of blocks, and ρ is the resilience parameter of the code. Considering a load-balancing among blocks, a same number of nodes are contacted within these b_c blocks. (A total of $k = k_c b_c$ nodes and downloading α - i.e., all - symbols from each.) This constitutes data collection property of BFR codes. ($\rho = 0$ case can be considered as a special case of batch codes introduced in [15].) Then, we introduce repairability in BFR codes, where any node of a failed block can be reconstructed from any d_r of any remaining $b_r \leq b-1$ blocks. (A total of $d = d_r b_r$ nodes and downloading β symbols from each.) As introduced in [1], we utilize graph expansion of DSS employing these repairable codes, and derive file size bounds and characterize BFR-MBR and BFR-MSR points. (We note that the blocks in our model can be used to model racks in DSS. Such a model is related to the work [16] which differentiates between within-rack communication and cross-rack communication. Our focus here would correspond to the case where within rack communication is much higher than the cross-rack communication, as no nodes from the failed rack can be contacted to regenerate a node.) We construct explicit codes achieving these points for a wide set of parameters. For a system with $b = 2$ blocks case, we show that achieving both MSR and MBR properties simultaneously is asymptotically possible. (This is somewhat similar to the property of Twin codes [17], but here the data collection property is different.) Then, for a system with $b \geq 3$ blocks case, we consider utilizing multiple codewords, which are placed into DSS via a combinatorial design based codeword placement algorithm. We show this technique establishes optimal codes for a wide set of parameter ranges.

The paper is organized as follows. Section II introduces model and preliminaries. Section III is devoted to the analysis of file size bounds. Code constructions are provided in Section IV. Section V includes extensions and concluding remarks.

II. Background and Preliminaries

A. Block failure resilient codes and repairability

Consider a code $\mathcal{C}$ which maps $\mathcal{M}$ symbols (over $\mathbb{F}_q$) in $\mathbf{f}$ (file) to length n codewords (nodes) $\mathbf{c} = (c_1, \cdots, c_n)$ with $c_i \in \mathbb{F}_q^{\alpha}$ for $i = 1, \cdots, n$. These codewords are distributed into b blocks each with block capacity $c = \frac{b}{n}$ nodes per block. We have the following definition.

Definition 1 (Block Failure Resilient (BFR) Codes)**.** *An $(n, b, \mathcal{M}, k, \rho, \alpha)$ block failure resilient (BFR) code encodes $\mathcal{M}$ elements in $\mathbb{F}_q$ ($\mathbf{f}$) to n codeword symbols (each in $\mathbb{F}_q^{\alpha}$) that are grouped into b blocks such that $\mathbf{f}$ can be decoded by accessing to any $\frac{k}{b-\rho}$ nodes of from each of the $b-\rho$ blocks.*

We remark that, in the above, ρ represents the resilience parameter of the BFR code, i.e., the code can tolerate ρ block erasures. Due to this data collection (file decoding) property of the code, we denote the number of blocks accessed as $b_c = b-\rho$ and number of nodes accessed per block as $k_c = \frac{k}{b_c}$. Noting that $k_c \leq c$ should be satisfied, we differentiate between *partial* block access, $k_c < c$, and *full* block access $k_c = c$. Throughout the paper, we assume $n|b$. i.e., c is integer, and $(b-\rho)|k$, i.e., k_c is integer.

Remarkably, any MDS array code [19] can be utilized as BFR codes for the full access case. In fact, such an approach will be optimal in terms of minimum distance, and therefore for resilience ρ. However, for $k_c < c$, MDS array codes may not result in an optimal code. Constructing optimal BFR codes in terms of the trade-off between resilience ρ and code rate $\frac{\mathcal{M}}{n\alpha}$ will be studied elsewhere. In this work, we focus on repairable BFR codes, as defined in the following.

Definition 2 (Block Failure Resilient Regenerating Codes (BFR-RC))**.** *An $(n, b, \mathcal{M}, k, \rho, \alpha, d, \sigma, \beta)$ block failure resilient regenerating code (BFR-RC) is an $(n, b, \mathcal{M}, k, \rho, \alpha)$ BFR code (data collection property) with the following repair property: Any node of a failed block can be reconstructed by accessing to any $d_r = \frac{d}{b-\sigma}$ nodes of any $b_r = b-\sigma$ blocks and downloading β symbols from each of these $d = b_r d_r$ nodes.*

We assume $(b-\rho)|d$, i.e., d_r is integer. (Note that d_r should necessarily satisfy $\frac{d}{b-\sigma} = d_r \leq c = \frac{n}{b}$ in our model.) We consider the trade-off between the *repair bandwidth* $\gamma = d\beta$ and *per node storage* α similar to the seminal work [1]. In particular, we define $\alpha_{\text{BFR-MSR}} = \frac{\mathcal{M}}{k}$ as the minimum per node storage and $\gamma_{\text{BFR-MBR}} = \alpha$ as the minimum repair bandwidth for an $(n, b, \mathcal{M}, k, \rho, \alpha, d, \sigma, \beta)$ BFR-RC. When deriving this trade-off, we focus on systems having $d_r = \frac{d}{b-\sigma} \geq k_c = \frac{k}{b-\rho}$, i.e., data collection process contacts to less number of nodes per block as compared to symbol regeneration. (We note that, similar to regenerating codes, without loss of generality, one should only consider systems that satisfy $d \geq k$, i.e., $d_r(b-\sigma) \geq k_c(b-\rho)$. Therefore, our $d_r \geq k_c$ assumption can be made without loss of generality for systems having $\rho \leq \sigma$.)

B. Information flow graph

The operation of a DSS employing such codes can be modeled by a multicasting scenario over an information flow graph [1], which has three types of nodes: 1) Source node (S): Contains original file $\mathbf{f}$. 2) Storage nodes, each represented as x_i with two sub-nodes($(x_i^{\text{in}}, x_i^{\text{out}})$), where x^{in} is the sub-node having the connections from the live nodes, and x^{out} is the storage sub-node, which stores the data and is contacted for node repair or data collection (edges between each x_i^{in} and x_i^{out}) has α-link capacity). 3) Data collector (DC) which contacts x^{out} sub-nodes of k live nodes (with edges each having ∞-link capacity). (As described above, for BFR codes these k nodes can be any $\frac{k}{b-\rho}$ nodes from each of the $b-\rho$ blocks.) Then, for a given graph $\mathcal{G}$ and DCs DC_i, the file size can be bounded using the max flow-min cut theorem for multicasting utilized in network coding [1], [20].

Lemma 3 (Max flow-min cut theorem for multicasting)**.**

$$\mathcal{M} \leq \min_{\mathcal{G}} \min_{\text{DC}_\text{i}} \text{maxflow}(\text{S} \rightarrow \text{DC}_\text{i}, \mathcal{G}),$$

where $\text{flow}(\text{S} \rightarrow \text{DC}_\text{i}, \mathcal{G})$ *represents the flow from the source node S to* DC_i *over the graph $\mathcal{G}$.*

Therefore, $\mathcal{M}$ symbol long file can be delivered to a DC, only if the min cut is at least $\mathcal{M}$. In the next section, similar to Dimakis et al., [1], we consider k successive node failures and evaluate the min-cut over possible graphs, and obtain a file size bound for a DSS operating with BFR-RC.

C. Block designs and projective planes

We first provide the definition of balanced incomplete block designs (BIBDs) [21].

Definition 4 (Balanced incomplete block design)**.** *A (v, κ, λ)-BIBD has v points distributed into blocks of size κ such that any pair of points are contained in λ blocks.*

Corollary 5. *For a (v, κ, λ)-BIBD,*

- *Every point occurs in $r = \frac{\lambda(v-1)}{\kappa-1}$ blocks.*
- *The design has exactly $b = \frac{vr}{\kappa} = \frac{\lambda(v^2-v)}{\kappa^2-\kappa}$ blocks.*

In the achievable schemes of this work, we utilize a special class of block designs that are called projective planes.

Definition 6. *A $(v = p^2+p+1, \kappa = p+1, \lambda = 1)$-BIBD with $p \geq 2$ is called a projective plane of order p.*

Projective planes have the property that every pair of blocks intersect at a unique point (as $\lambda = 1$). In addition, due to Corollary 5, in projective planes, every point occurs in $r = p+1$ blocks, and there are $b = v = p^2+p+1$ blocks.

III. File Size Bound for Repairable BFR Codes

Information flow graph analysis, similar to that of considered in [1], can be performed to obtain file size bounds for repairable BFR codes. In this paper, we focus on the case $\sigma = 1$, i.e., regeneration of a node in a failed block is performed by contacting to all remaining live blocks. In the following, we first analyze $\rho = 0$ case, i.e., data collector connects all the blocks to reconstruct the data.

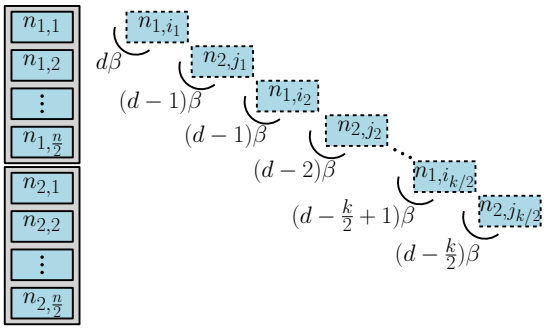

Fig. 1. Repair process for $b = 2$ (two blocks) case.

A. $\rho = 0$, $b = 2$ *case*

Consider $b = 2$-block case as in Fig. 1 and assume $2|k$. From Lemma 3, the file size $\mathcal{M}$ can be upper bounded with the repair procedure shown in Fig. 1, which displays one of the "minimum-cut" scenarios, wherein any two consecutive node failures belong to different blocks. Assuming k is even and $d \geq \frac{k}{2}$,

$$\mathcal{M} \leq \sum_{i=0}^{\frac{k}{2}-1} \min(\alpha, (d-i)\beta) + \sum_{i=1}^{\frac{k}{2}} \min(\alpha, (d-i)\beta). \quad (1)$$

Achieving this upper bound (1) with equality would yield maximum possible file size. One particular instance is shown in Fig. 1, and we note that the order of failed nodes does not matter as the sum of the cut would be the same with different order of failures as long as we consider connection from data collector to $\frac{k}{2}$ repaired nodes from each block.

For MSR point, $\alpha = \alpha_{\text{BFR-MSR}} = \frac{\mathcal{M}}{k}$. In the bound (1), we then have $\alpha_{\text{BFR-MSR}} \leq (d - \frac{k}{2})\beta_{\text{BFR-MSR}}$. Achieving equality would give the minimum repair bandwidth for the MSR case. Hence, BFR-MSR point is given by

$$(\alpha_{\text{BFR-MSR}}, \gamma_{\text{BFR-MSR}}) = (\frac{\mathcal{M}}{k}, \frac{2\mathcal{M}d}{2kd-k^2}). \quad (2)$$

BFR-MBR codes, on the other hand, have the property that $d\beta = \alpha$ with minimum possible $d\beta$ while achieving the equality in (1). Inserting $d\beta = \alpha$ in (1), we obtain that

$$(\alpha_{\text{BFR-MBR}}, \gamma_{\text{BFR-MBR}}) = (\frac{4\mathcal{M}d}{4dk-k^2}, \frac{4\mathcal{M}d}{4dk-k^2}). \quad (3)$$

Same analysis can be done for odd values of k as well,

$$(\alpha_{\text{BFR-MSR}}, \gamma_{\text{BFR-MSR}}) = \begin{cases} (\frac{M}{k}, \frac{2Md}{2kd-k^2-k}), & \text{if } k \text{ is odd} \\ (\frac{M}{k}, \frac{2Md}{2kd-k^2}), & \text{o.w.} \end{cases} \quad (4)$$

$$(\alpha_{\text{BFR-MBR}}, \gamma_{\text{BFR-MBR}}) = \begin{cases} (\frac{4Md}{4dk-k^2+1}, \frac{4Md}{4dk-k^2+1}), & \text{if } k \text{ is odd} \\ (\frac{4Md}{4dk-k^2}, \frac{4Md}{4dk-k^2}), & \text{o.w.} \end{cases} \quad (5)$$

Here, we compare $\gamma_{\text{BFR-MSR}}$ and γ_{MBR}. We have $\gamma^{\text{k-odd}}_{\text{BFR-MSR}} \geq \gamma^{\text{k-even}}_{\text{BFR-MSR}} \geq \gamma_{\text{MBR}} = \frac{2\mathcal{M}d}{k(2d-k+1)}$, and, if we have $2d - k \gg 1$, then $\gamma^{\text{k-odd}}_{\text{BFR-MSR}} \approx \gamma^{\text{k-even}}_{\text{BFR-MSR}} \approx \gamma_{\text{MBR}}$. This implies that BFR-MSR codes with $b = 2$ achieves repair bandwidth of MBR and per-node storage of MSR codes simultaneously for systems with $d \gg 1$. We provide the generalization of these bounds to $b \geq 2$ case in the following.

B. $\rho = 0$, $b \geq 2$ *case*

The same steps described above can be used to derive the file size bound for b-blocks.

Lemma 7. *The optimal file size is given by*

$$\begin{aligned} \mathcal{M} = &\sum_{i=0}^{\frac{k}{b}-1} \min(\alpha, (d-(b-1)i)\beta) \\ &+ \sum_{i=0}^{\frac{k}{b}-1} \min(\alpha, (d-1-(b-1)i)\beta) + \dots \\ &+ \sum_{i=0}^{\frac{k}{b}-1} \min(\alpha, (d-(b-1)-(b-1)i)\beta). \end{aligned} \quad (6)$$

Proposition 8. *BFR-MSR and BFR-MBR points are as follows,*

$$(\alpha_{\textit{BFR-MSR}}, \gamma_{\textit{BFR-MSR}}) = \left(\frac{\mathcal{M}}{k}, \frac{\mathcal{M}d}{kd - \frac{k^2(b-1)}{b}}\right) \quad (7)$$

$$(\alpha_{\textit{BFR-MBR}}, \gamma_{\textit{BFR-MBR}}) = \left(\frac{\mathcal{M}d}{kd - \frac{k^2(b-1)}{2b}}, \frac{\mathcal{M}d}{kd - \frac{k^2(b-1)}{2b}}\right) \quad (8)$$

We observe that $\gamma_{\text{BFR-MSR}} \leq \gamma_{\text{MSR}} = \frac{\mathcal{M}d}{k(d-k+1)}$ for $b \leq k$, which is the case here as $b \mid k$. Also, we have $\frac{\gamma_{\text{BFR-MSR}}}{\gamma_{\text{MBR}}} = \frac{d-\frac{k-1}{2}}{d-k\frac{b-1}{b}} \geq 1$ when $b \geq \frac{2k}{k+1}$ which is always true. Hence, $\gamma_{\text{BFR-MSR}}$ is between γ_{MSR} and γ_{MBR}.

C. $\rho > 0$ *case*

If we restrict data collector to connect $b_c < b$ blocks (i.e., $\rho > 0$), but keep the repair process same as before, the above analysis follows and corresponding MSR and MBR points are given by replacing b in (7) and (8) with $b_c = b - \rho$ - for systems satisfying $d_r \geq k_c$. (This follows as the repair from these ρ blocks will not contribute to the cut between the source S and DC.)

IV. BFR-MSR and BFR-MBR Code Constructions

A. *Transpose code for b=2 case*

One instance of BFR codes is given in the Fig. 2. We set $\alpha = d = \frac{n}{2}$, and store the transpose of the first block's symbols in the second block. The repair of a failed node i in the first block can be performed by connecting all the nodes in the second block and downloading only 1 symbol from each node. That is, $d\beta = \alpha$. Further, we set $\mathcal{M} = kd - (\frac{k}{2})^2$, and use an $[N = \alpha^2, K = \mathcal{M}]$ MDS code to encode file $\mathbf{f}$ into symbols denoted with $x_{i,j}$, $i, j = 1, ..., \alpha$. BFR data collection property allows for reconstructing the file, as connecting any $\frac{k}{2}$ nodes from each block assures at least K distinct symbols. This code is a BFR-MBR code for $\beta = 1$ (scalar code), as the optimal file size in (3), i.e., $\mathcal{M} = kd - (\frac{k}{2})^2$, is achieved with $d\beta = \alpha$. A similar code to this construction is Twin codes introduced in [17], where the nodes are split into two types and a failed node of a a given type is regenerated by connecting to nodes only in the other type. However, Twin codes, as opposed to our model, do not have balanced node connection for data

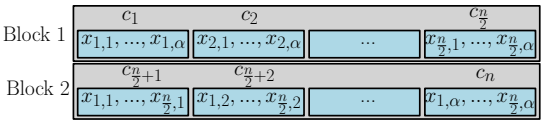

Fig. 2. Transpose code is a two-block BFR-MBR code.

collection. In particular, DC connects to only (a subset of k nodes from) a single type. On the other hand, BFR codes, for $b = 2$ case, connects to $\frac{k}{2}$ nodes from each block.

B. Block design based regenerating code symbol placement

Consider that the file $\mathcal{F}$ of size $\mathcal{M}$ contains 3 sub-files $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ each of size $\tilde{\mathcal{M}}$. We encode these sub-files with $[\tilde{n} = 10, \tilde{k} = 4, \tilde{d} = 5, \tilde{\alpha}, \tilde{\beta}]$ regenerating code $\tilde{\mathcal{C}}$, represent the resulting symbols with $\mathcal{P}_1 = p_{1,1:\tilde{n}}$ for $\mathcal{F}_1$, $\mathcal{P}_2 = p_{2,1:\tilde{n}}$ for $\mathcal{F}_2$, and $\mathcal{P}_3 = p_{3,1:\tilde{n}}$ for $\mathcal{F}_3$. These symbols are grouped in a specific way placed into nodes within blocks as represented in Fig. 3, where each node contains two symbols each coming from two of the different sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$. We set the sub-code $\tilde{\mathcal{C}}$ parameters as $[\mathcal{M} = 3\tilde{\mathcal{M}}, k = \frac{3}{2}\tilde{k}, d = 2\tilde{d}, \alpha = 2\tilde{\alpha}, \beta = \tilde{\beta}]$.

Assume Block 1 is unavailable and its first node, which contains codeword c_1, has to be reconstructed. Due to underlying regenerating code, contacting 5 nodes of Block 2 and accessing to $p_{1,6:10}$ repairs $p_{1,1}$. Similarly, $p_{2,1}$ can be reconstructed from Block 3. Any node failures can be handled similarly, by connecting to remaining 2 blocks and repairing each symbol of lost node by connecting $\tilde{d}$ nodes in a block. As we have $k = 6$, DC, connecting to 2 nodes from each block, obtains 12 symbols which has 4 different symbols from each of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. As the embedded regenerating code has $\tilde{k} = 4$, all 3 sub-files can be recovered.

We generalize the BFR-RC construction above utilizing projective planes. First, the file $\mathbf{f}$ of size $\mathcal{M}$ is partitioned into v parts, $\mathcal{M}_1, \mathcal{M}_2,...,\mathcal{M}_v$. Each part, of size $\tilde{\mathcal{M}}$, then encoded using $[\tilde{n}, \tilde{k}, \tilde{d}, \tilde{\alpha}, \tilde{\beta}]$ regenerating code $\tilde{\mathcal{C}}$. We represent the resulting symbols with $\mathcal{P}_i = p_{i,1:\tilde{n}}$ for $i = 1, \cdots, v$. We then consider index of each part as a point in a $(v = p^2+p+1, \kappa = p+1, \lambda = 1)$ projective plane. (Indices of symbol sets $\mathcal{P}_\mathcal{J}$ and points $\mathcal{J}$ of projective plane are used interchangeably in the following.) We perform the placement of each point in the system using this projective plane mapping. (The setup in Fig. 3 can be considered as a toy model. Although the combinatorial design with blocks given by $\{p_1, p_2\}, \{p_1, p_3\}, \{p_2, p_3\}$ has projective plane properties, it is not considered as an instance of a projective plane.) In this placement, total of $\tilde{n}$ nodes from each partition $\mathcal{P}_i$ are distributed to r blocks evenly, each block contains $\frac{\tilde{n}}{r}$ nodes where each node stores $\alpha = \kappa\tilde{\alpha}$ symbols. Note that blocks of projective plane give the indices of parts $\mathcal{P}_i$ stored in the nodes of the corresponding block in DSS. That is, all nodes in a block stores symbols from unique subset of $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_v\}$ of size κ. Overall, the system can store a file of size $\mathcal{M} = v\tilde{\mathcal{M}}$ with $b = v$ blocks. We set the sub-code $\tilde{\mathcal{C}}$ parameters as

$$M = v\tilde{M}, k = \frac{b}{r}\tilde{k}, d = \kappa\tilde{d}, \alpha = \kappa\tilde{\alpha}, \beta = \tilde{\beta} \tag{9}$$

where we choose parameters to satisfy $r - 1 \mid \tilde{d}$, $r \mid \tilde{n}$ and $r \mid \tilde{k}$.

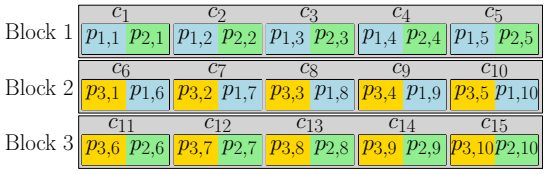

Fig. 3. Three-block BFR-RC via projective plane symbol placement.

Node Repair: Consider that one of the nodes in a block is to be repaired. Note that the failed node contains κ symbols, each coming from a distinct subfile's regenerating codeword. Using projective planes' property that any 2 blocks has only 1 point in common, any remaining block can help for in the regeneration of 1 symbol of the failed node. Furthermore, as any point has a repetition degree of r, one can connect to $r-1$ blocks, $\frac{\tilde{d}}{r-1}$ nodes per block, to repair one symbol of a failed node. Combining these two, node regeneration is performed by connecting $(r-1)\kappa$ blocks. Substituting $\kappa = p+1$ and $r = p+1$, connecting to $p^2 + p = b - 1$ blocks allows for reconstructing any node of a failed block.

Data Collection: DC, connects $\frac{\tilde{k}}{r}$ nodes per block from all $b_c = b$ blocks, i.e., a total of $k = \frac{b}{r}\tilde{k}$ nodes each having encoded symbols of κ subfiles. These total of $v\tilde{k}$ symbols include $\tilde{k}$ symbols from each subfile, from which all subfiles, hence the file $\mathbf{f}$, can be decoded.

1) BFR-MSR: To construct a BFR-MSR code, we set each subcode $\tilde{\mathcal{C}}$ as an MSR code, which has

$$\tilde{\alpha} = \frac{\tilde{\mathcal{M}}}{\tilde{k}}, \tilde{d}\tilde{\beta} = \frac{\tilde{\mathcal{M}}\tilde{d}}{\tilde{k}(\tilde{d} - \tilde{k} + 1)}. \tag{10}$$

This, together with (9), results in the following parameters of our BFR-MSR construction

$$\alpha = \tilde{\alpha}\kappa = \frac{\mathcal{M}}{k}, d\beta = \kappa\tilde{d}\tilde{\beta} = \frac{\mathcal{M}d}{k(d - \frac{k(p+1)^2}{p^2+p+1} + p + 1)}. \tag{11}$$

We remark that if we utilize ZigZag codes [2] as the subcode $\tilde{\mathcal{C}}$ above, we have $[\tilde{n}, \tilde{k}, \tilde{d} = \tilde{n} - 1, \tilde{\alpha} = \tilde{r}^{\tilde{k}-1}, \tilde{\beta} = \tilde{r}^{\tilde{k}-2}, \tilde{r} = \tilde{n} - \tilde{k}]$, and having $\tilde{d} = \tilde{n} - 1$ requires connecting to 1 node per block for repairs in our block model. On the other hand, product matrix MSR codes [3] can be used as the subcode $\tilde{\mathcal{C}}$ for any $\tilde{d} \geq 2\tilde{k} - 2$, for which we do not necessarily have $\frac{\tilde{d}}{r-1} = 1$. We observe from (7) and (11) that MSR point is achieved for $\tilde{k} = p + 1$, meaning $k = b$.

2) BFR-MBR: To construct a BFR-MBR code, we set each subcode $\tilde{\mathcal{C}}$ as a product matrix MBR code [3], which has

$$\tilde{\alpha} = \tilde{d}\tilde{\beta} = \frac{2\tilde{\mathcal{M}}\tilde{d}}{\tilde{k}(2\tilde{d} - \tilde{k} + 1)}. \tag{12}$$

This, together with (9), results in the following parameters of our BFR-MSR construction

$$\alpha = d\beta = \frac{2\mathcal{M}d}{k(2d - \frac{k(p+1)^2}{p^2+p+1} + p + 1)}. \tag{13}$$

From (8) and (13), MBR point is achieved for $\tilde{k} = p + 1$.

V. Extensions and concluding remarks

A. $\rho > 0$ case

In the above, we considered the cases where DC connects all b blocks in file reconstruction. In order to support $b_c < b$, we consider employing Gabidulin codes [18] as an outer code similar to the constructions provided in [8], [10]. We briefly discuss our approach here. Detailed results will be provided elsewhere. $[N, K, D = N - K + 1]_{q^m}$ Gabidulin code C^{Gab}, $m \geq N$, has a codeword $(f(g_1), f(g_2), ..., f(g_N)) \in \mathbb{F}_{q^m}^N$, where $f(x)$ is a linearized polynomial over $\mathbb{F}_{q^m}$ of q-degree $K-1$ with K message symbols as its coefficients and $g_1, g_2, ..., g_N \in \mathbb{F}_{q^m}$ are linearly independent over F_q [18].

Remark 9. *Given evaluations of $f(\cdot)$ at any K linearly independent (over $\mathbb{F}_q$) points in $\mathbb{F}_{q^m}$, one can reconstruct the message vector.*

Here, before partitioning the message into v parts, we encode the file with a Gabidulin code first, then partition the resulting codeword into v parts and follow remaining steps as before. With this approach, decoding the message at DC follows by obtaining at least K independent evaluations from k nodes, $k_c = \frac{k}{b_c}$ nodes per block from a total of $b_c = b - \rho$ blocks. As considered in [8], [10], the number of such evaluations can be derived from the rank accumulation profile of the inherent MSR/MBR codes $\tilde{\mathcal{C}}$ as in the following

$$\tilde{a_j} = \begin{cases} \tilde{\alpha}, & \text{if } \tilde{\mathcal{C}} \text{ is MSR and } 1 \leq j \leq \tilde{k} \\ \tilde{\alpha} - (j-1)\tilde{\beta}, & \text{if } \tilde{\mathcal{C}} \text{ is MBR and } 1 \leq j \leq \tilde{k} \\ 0, & \text{if } \tilde{\mathcal{C}} \text{ is MSR/MBR and } \tilde{k}+1 \leq j \leq \tilde{n} \end{cases} \tag{14}$$

Note that because of projective plane property, connecting $b-1$ blocks would result in getting $k_c r$ evaluations for $v - \kappa$ points and $k_c(r-1)$ evaluations for κ points. Hence DC can decode the message by using an outer Gabidulin code if

$$\sum_{t=1}^{v-\kappa} \sum_{j=1}^{k_c r} \tilde{a}_j + \sum_{t=1}^{\kappa} \sum_{j=1}^{k_c(r-1)} \tilde{a}_j \geq K. \tag{15}$$

Similarly, for $b_c = b - 2$, decoding at DC is possible if

$$\sum_{t=1}^{v-(2\kappa-1)} \sum_{j=1}^{k_c r} \tilde{a}_j + \sum_{t=1}^{2\kappa-2} \sum_{j=1}^{k_c(r-1)} \tilde{a}_j + \sum_{j=1}^{k_c(r-2)} \tilde{a}_j \geq K. \tag{16}$$

With such an approach, for $b_c \leq b - 3$ there are multiple collection possibilities for DC. For example, by connecting $b-3$ blocks DC can observe either a) $k_c r$ evaluations for $v - (3\kappa - 2)$, $k_c(r-1)$ evaluations for $3(\kappa - 1)$ points and $k_c(r-3)$ evaluations for 1 point, or b) $k_c r$ evaluations for $v - (3\kappa - 3)$, $k_c(r-1)$ evaluations for $3(\kappa - 2)$ points and $k_c(r-2)$ evaluations for 3 points. Therefore, we need to ensure that minimum rank accumulations of all cases is at least K.

B. Concluding remarks

We introduced the framework of block failure resilient (BFR) codes that can recover data stored in the system from a subset of available blocks with a load balancing property. Repairability is studied, file size bounds are derived, BFR-MSR and BFR-MBR points are characterized, explicit code constructions for a wide set of parameters are provided.

References

[1] 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, Sep. 2010.

[2] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[3] 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, Aug. 2011.

[4] 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, Mar. 2011.

[5] 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, Nov. 2012.

[6] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. 2012 IEEE International Symposium on Information Theory (ISIT 2012)*, Boston, MA, Jul. 2012.

[7] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. 2011 IEEE INFOCOM*, Shanghai, China, Apr. 2011.

[8] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, Jan. 2014.

[9] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration," *CoRR*, vol. abs/1211.1932, Nov. 2012.

[10] G. M. Kamath, N. Silberstein, N. Prakash, A. S. Rawat, V. Lalitha, O. O. Koyluoglu, P. V. Kumar, and S. Vishwanath, "Explicit MBR all-symbol locality codes," in *Proc. 2013 IEEE International Symposium on Information Theory (ISIT 2013)*, Istanbul, Turkey, Jul. 2013.

[11] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endow.*, vol. 6, no. 5, pp. 325–336, Mar. 2013.

[12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows azure storage," in *Proc. USENIX Annual Technical Conference*, Boston, MA, Jun. 2012.

[13] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. Nineteenth ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, Oct. 2003.

[14] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Symposium on Operating Systems Design and Implementation*, Vancouver, BC, Oct. 2010.

[15] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proc. Thirty-sixth Annual ACM Symposium on Theory of Computing*, Chicago, IL, Jun. 2004.

[16] B. Gaston, J. Pujol, and M. Villanueva, "A realistic distributed storage system: The rack model," *CoRR*, vol. abs/1302.5657, Feb. 2013.

[17] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Enabling node repair in any erasure code for distributed storage," in *Proc. 2011 IEEE International Symposium on Information Theory (ISIT 2011)*, Saint Petersburg, Russia, Jul. 2011.

[18] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problemy Peredachi Informatsii*, vol. 21, no. 1, pp. 3–16, 1985.

[19] F. J. McWilliams and N. J. A. Sloane, *The theory for error-correcting codes*. North-Holland, 1977.

[20] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[21] D. R. Stinson, *Combinatorial designs: construction and analysis*. Springer, 2004.