Optimal Systematic Distributed Storage Codes with Fast Encoding

Preetum Nakkiran, K. V. Rashmi, Kannan Ramchandran, *Fellow, IEEE*

***Abstract*—We consider the problem of constructing explicit erasure codes for distributed storage with the following desirable properties motivated by system constraints: (i) Maximum-Distance-Separable (MDS), (ii)Optimal repair-bandwidth, (iii)Flexibility in repair (as will be described), (iv) Systematic Form, and (v) Fast encoding (enabled by a *sparse* generator matrix). Existing constructions in the literature satisfy only strict subsets of these desired properties. This paper presents the first explicit code construction which theoretically guarantees all the five desired properties *simultaneously*. We first present a construction that builds on Product-Matrix (PM) codes by enabling sparsity in its generator matrix. We then present a transformation for general classes of storage and repair optimal codes to enable fast encoding through sparsity. In practice, such sparse codes are roughly 7 times sparser than their standard counterparts, and result in encoding speedup by a factor of about 4 for typical parameters.**

I. Introduction

Erasure codes are being increasingly deployed in distributed-storage systems in place of replication, since they provide the same level of reliability with much less storage overhead. Large scale distributed-storage systems have many requirements that constrain and guide the design of underlying distributed-storage codes.

In large-scale systems, storage is a critical resource. For this reason, *Maximum-Distance-Separable (MDS)* codes such as Reed-Solomon codes, which require the minimal storage overhead to achieve a desired level of reliability, are a necessity [1]. An $[n, k]$ MDS code allows the data to be stored across n nodes such that the entire data can be recovered from the encoded data stored in any k (out of n) nodes. This is depicted in Figure 1a. Another critical resource is the network bandwidth. In large-scale systems, failures are the norm rather than the exception, and repair operations run continuously in the background [2], [3]. When nodes fail, they must be repaired by downloading some data from the remaining nodes. These nodes are termed *helper nodes*. Figure 1b depicts a repair operation where node 1 is being repaired with the help of nodes $\{2, \ldots, n-1\}$. Repair operations can consume a significant amount of network bandwidth [2]. Hence, it is important for storage codes to also minimize the amount of bandwidth consumed during repair.

Another important system consideration is that the failure of a single node should not force all the surviving $(n-1)$ nodes to be required in order to repair it. If d denotes the number of helper nodes required for repair, then this property requires $d < (n-1)$, as illustrated in Figure 1b. This property allows the system to send requests to more than d nodes during repair and make use of the *first* d nodes that respond. Such additional requests are called redundant requests, and these are helpful in reducing repair latency. Moreover, this property is critical to the performance of read requests called *degraded reads* [2], wherein a read request to data stored in a node that is busy or otherwise unavailable is served by performing a repair operation for that node's data.

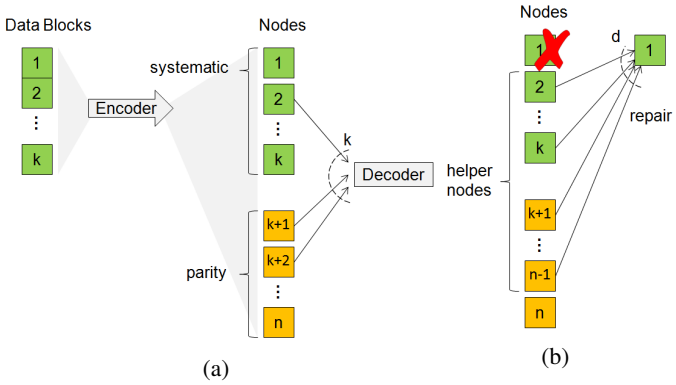


Fig. 1: (a) Encoding and decoding for an $[n, k]$ systematic MDS code, (b) Node repair: Connecting to $d = (n-2)$ helper nodes to repair failed node 1.

Another important system requirement of storage codes is of it being in *systematic* form. That is, the original data must exist in the system in an uncoded form. Figure 1a shows a systematic code wherein the first k nodes store the original data. This is essential when serving read requests since if the code is systematic, read requests can be served by simply reading the data in systematic nodes, without any decoding operation.

Finally, one of the most frequent operations performed in many distributed-storage systems is encoding the new data entering the system. Encoding cost is a non-issue when using replication, but can be significant when using erasure codes. Thus it is desirable for the code to support fast encoding operations. For linear codes, encoding the original data can be represented as multiplication between a generator matrix and the data vector. This encoding operation will be fast if the generator matrix is *sparse*, since this reduces the number of computations performed.

This forms the motivation for this paper: to construct storage codes that satisfy *all* the above system-driven constraints. That

The authors are with the department of EECS, University of California, Berkeley. Email: preetum@berkeley.edu, {rashmikv,kannanr}@eecs.berkeley.edu. This work was supported in part by NSF grant number 1409135, MURI CHASE grant number 556016, and a grant from Cisco.

is, storage codes having the following five properties: (i) MDS, (ii) Minimal repair bandwidth, (iii) Flexible repair parameters: $d < (n-1)$, (iv) Systematic form, and (v) Fast encoding, enabled by a sparse generator matrix.

There has been considerable interest in the recent past in constructing such erasure codes for distributed storage [4]–[8]. However, to the best of our knowledge, all existing constructions in the literature address only a *strict subset* of the above desired properties. This paper presents the first explicit codes which theoretically guarantee all the five desired properties simultaneously.

We first present a construction based on a powerful class of storage codes called Product-Matrix (PM) codes [5]. PM codes [1] are MDS and meet the lower bound on bandwidth consumption during repair presented in [9]. PM codes belong to a general class of codes known as regenerating codes [9], which meet this lower bound. Moreover, PM codes support a wide range of values for $d : (2k-2) \leq d \leq (n-1)$, and also possess special structure that makes their generator matrix sparse, leading to fast encoding [10]. Thus PM codes satisfy Properties (i)-(iii) and (v). However, they are not systematic. PM codes can be converted to systematic form using a generic transformation termed *systematic-remapping* [5], which essentially first precodes the data using the "decoding" operation such that after encoding, the first k nodes are in uncoded form. However, this remapping *often destroys the sparsity of PM codes.* Thus naively performing a remapping transform causes PM codes to be systematic, but at the expense of fast encoding. For example, an $[n, k, d = 2k-1]$ PM code requires a block-length of k^2 symbols. That is, each stored symbol can be, in general, a function of up to k^2 data symbols. However, due to the sparse structure of native PM codes, each stored symbol is a function of only $O(k)$ of these data symbols. This is no longer true after systematic remapping, and in general each parity symbol becomes a (dense) function of k^2 symbols. This results in significantly higher encoding time for systematic PM codes constructed in this manner [10], [11].

We then consider sparsity in general classes of storage and repair optimal codes, and present a transformation to enable sparsity leading to fast encoding.

A. Results

In this paper, first, we present an analytical framework for studying and understanding the interaction between the design of PM codes and the systematic-remapping transformation. Using this, we provide an explicit construction of PM codes which remains sparse after systematic-remapping, for $d = (2k-2)$. In particular, each parity symbol in this construction depends on only $d = O(k)$ data symbols. [2]

Second, we consider the sparsity of general distributed storage codes supporting a property called *repair-by-transfer.* A node assisting in a repair operation is said to perform repair-by-transfer if it does not perform any computation and merely transfers some of its stored symbols to the failed node. Storage codes which support repair-by-transfer are appealing, since they also minimize the amount of data read during repairs. We show that a particular type of repair-by-transfer property leads to sparsity in *any* MDS regenerating code. *This result provides a general way of constructing sparse MDS regenerating codes.*

Third, using the above result, we construct explicit sparse systematic PM codes for all $d \geq (2k-2)$. For example, the generator matrix of an $[n = 17, k = 8, d = 15]$ systematic-remapped PM code as in [5] is about 11% sparse, while our construction is about 77% sparse.

We note that the construction provided in this paper is similar to the codes considered in [12], wherein the authors present codes supporting repair-by-transfer for achieving savings in disk I/O. For $d = (2k-2)$, the construction provided in the present paper is also similar to the recent construction in [11] by Le Scouarnec, which presents a sparse PM code and computationally validates its properties for a fixed range of k. In fact, the results presented in this paper provide a theoretical proof of sparsity for the constructions in both works [11], [12].

The full version of this paper on arXiv [13] includes proofs, remarks, and examples that are omitted here due to space restrictions. The full version additionally includes results showing that the two constructions of sparse PM codes for $d = (2k-2)$ presented here are equivalent in a certain sense.

II. Background

A. Regenerating Codes

Let the message to be stored consist of B symbols from a finite field. An $[n, k, d](\alpha, \beta)$ regenerating code [9] allows the message to be stored across n nodes, each storing α encoded symbols. All the B symbols can be recovered from the data stored in any k nodes. Further, any node's data may be exactly recovered by connecting to any d other nodes and downloading $\beta \leq \alpha$ symbols from each. For linear regenerating codes, the symbols transferred from a helper node during node repair are a linear function of the data stored in it. In [9], the authors establish a fundamental tradeoff between storage overhead and network bandwidth used in node repair, and regenerating codes achieve optimal points on this tradeoff.

Minimum-Storage-Regenerating (MSR) codes are regenerating codes which are also MDS, and therefore satisfy $B = k\alpha$. It is shown in [9], that the parameters of any MSR code must necessarily satisfy $\alpha = \beta(d-k+1)$.

B. Product-Matrix Codes

Product-Matrix (PM) codes [5] are an explicit family of linear MDS codes which minimize bandwidth consumed in repair, and exist for all $[n, k, d \geq 2k-2]$. Further, they can be constructed with field size $O(nd)$. An $[n, k, d](\alpha)$ PM code is an $[n, k, d](\alpha, \beta = 1)$ linear regenerating code. In [5], the authors present explicit reconstruction and repair algorithms that demonstrate that PM codes are MDS and consume optimal network bandwidth. Here we only briefly describe some of the

[1] We use "PM codes" here to refer to the MDS version of Product-Matrix codes, termed PM-MSR in [5].

[2] Note that for a systematic $[n, k, d]$ MDS code, a sparsity of at least k symbols is necessary for each encoded symbol in the parity nodes.

notation and background of PM codes; further details can be found in the full version [13] and in [5].

In general, a PM code is described by an $(n \times d)$ *encoding matrix* Ψ and a $(d \times \alpha)$ *message matrix* M, yielding an $(n \times \alpha)$ *code matrix* C defined by

$$C := \Psi M. \tag{1}$$

Let c_i^T denote the i^{th} row of the code matrix C. Then the i^{th} node stores $c_i^T = \psi_i^T M$.

Here we review PM codes for $d = (2k-2)$. In this case, we have $\alpha = (d-k+1) = (k-1)$. For these parameters, the encoding matrix Ψ is of the form $\Psi = \begin{bmatrix}\Phi & \Lambda\Phi\end{bmatrix}$, where Φ is an $(n \times \alpha)$ matrix and Λ is an $(n \times n)$ diagonal matrix, chosen appropriately.

We will now describe the structure of the message matrix M. Recall for $d = (2k-2)$, we have $\alpha = (k-1)$, $d = 2\alpha$, and $B = k\alpha = \alpha(\alpha+1)$. The $(d \times \alpha)$ message matrix M is constructed as

$$M = \begin{bmatrix} S^a \\ S^b \end{bmatrix} \tag{2}$$

where S^a and S^b are $(\alpha \times \alpha)$ symmetric matrices. The matrices S^a and S^b together have precisely $\alpha(\alpha+1)$ distinct entries, which are now populated by the $B = \alpha(\alpha+1)$ message symbols.

C. Systematic Codes and Remapping

Throughout this paper we consider *systematic codes* in which the first k nodes (referred to as *systematic nodes*) store the uncoded message symbols.

Any linear MDS erasure code can be generically transformed into a systematic code, as follows. First, any linear code taking B message symbols to $n\alpha$ encoded symbols can be represented by an $(n\alpha \times B)$ *generator matrix* G, such that for a message-vector m of length B, the encoded $n\alpha$ symbols are given by Gm. A code can be made systematic through a transformation called *systematic remapping*: Let G_k be a $(B \times B)$ matrix consisting of the first B rows of G. To encode message m, first "remap" the message vector to $\overline{m} := G_k^{-1} m$, then encode as $G\overline{m}$. Now the first B encoded symbols are exactly the message symbols m, making the code systematic.

Observe that the systematic remapping operation applies G_k^{-1}, and hence can be thought of as decoding the message from the first k nodes under the original encoding with generator matrix G.

The above transform can be applied to non-systematic PM codes, to yield systematic PM codes. However, this often destroys the code's sparsity. Extended examples illustrating these issues of sparsity are included in the full version [13].

D. Notation

We will use the concept of an *inclusion map*, which is a map that injectively embeds one space into another. Inclusion maps will be denoted by hooked arrows ($\hookrightarrow$). We will often abuse notation by using the same symbols to denote a space as well as a vector in the space. For example, the systematic-remapping transformation of a message vector m, as in Section II-C, will be written as a function $f : m \to \overline{m}$. The $(i,j)^{th}$ entry of a matrix M is denoted $M_{i,j}$. We say a vector is d-*sparse* if it has at most d non-zero entries.

III. First Step towards Sparsity in PM Codes

In this section we analyze a family of encoding matrices Ψ which, after systematic remapping, results in codes with partial sparsity.

Consider a $d = (2k-2)$ PM code in which the first row of Φ is $e_1 = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}$. Then the encoding matrix will take the form

$$\Psi = \begin{bmatrix} e_1 & \lambda e_1 \\ \Phi' & \Lambda'\Phi' \end{bmatrix}. \tag{3}$$

We will now show that under such PM codes, the first symbol stored in every node is d-sparse after systematic remapping.

Let Ψ_k denote the first k rows of Ψ, that is, the encoding submatrix for the first k nodes. Then the first k nodes store

$$C_k := \Psi_k M. \tag{4}$$

Let $f_e : M \to C_k$ denote the above encoding function for the first k nodes. We represent systematic remapping as a linear transformation $f_S : M \to \overline{M}$ between the original matrix M and the resultant message matrix after transformation $\overline{M}$. After the remapping, the first k nodes become systematic (see Section II-C). Equivalently, for a systematic code, the entire encoding transform:

$$M \xrightarrow{f_S} \overline{M} \xrightarrow{f_e} C_k \tag{5}$$

must act as an inclusion map $M \hookrightarrow C_k$. This inclusion map "unwraps" the symmetric matrices in M into one matrix C_k with distinct message symbols.

To understand the interaction between the PM code and systematic remapping, we will define an explicit inclusion map f_ι, and decompose f_S into two stages. We first represent the matrix M as the matrix C_k using a carefully-designed inclusion map f_ι, and then "decode" C_k into $\overline{M}$ using the decoding function f_e^{-1}:

$$f_S : M \xhookrightarrow{f_\iota} C_k \xrightarrow{f_e^{-1}} \overline{M} \tag{6}$$

Note that f_e is invertible since it is an MDS encoding, wherein all message symbols can be decoded from any k nodes. The remapping transform thus becomes $f_S = f_e^{-1} \circ f_\iota$. Thus the entire encoding transformation for the first k nodes becomes

$$M \xhookrightarrow{f_\iota} C_k \xrightarrow{f_e^{-1}} \overline{M} \xrightarrow{f_e} C_k \tag{7}$$

Notice that this makes the entire encoding transform $M \to C_k$ an inclusion map (equal to f_ι, in fact), thus resulting in a systematic code as desired.

At a high level, the key ideas behind our approach for showing sparsity are as follows.

(1) We will design an inclusion map f_ι such that: In the systematic remapping $f_S : M \to \overline{M}$, the first column of $\overline{M}$ depends only on the first column of M.
(2) The first stored symbol in node i is the i^{th} row of Ψ times the first column of $\overline{M}$. This depends only on the

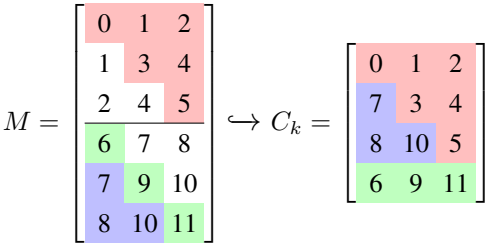
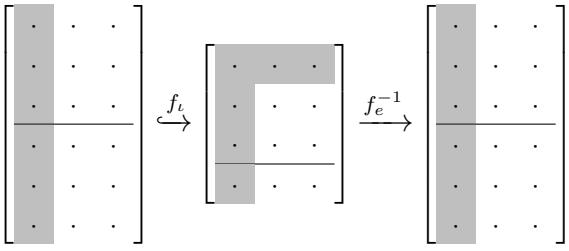


Fig. 2: An example of triangular inclusion map.



$$f_S : M \xhookrightarrow{f_\iota} C_k \xrightarrow{f_e^{-1}} \overline{M}$$

Fig. 3: Visualization of sparsity pattern during systematic remapping.

first column of $\overline{M}$, and therefore (through f_S) depends only on the first column of M.

A. The Triangular Inclusion Map

Here we will define the inclusion map f_ι, which we call the *triangular inclusion map*. Recall from Section II-B that the message matrix in $d = (2k-2)$ PM codes is of the form $M = \begin{bmatrix} S^a \\ S^b \end{bmatrix}$, where S^a and S^b are symmetric matrices of message symbols.

To map $M \hookrightarrow C_k$ by inclusion, place the upper-triangular half of S^a on the upper-triangular half of C_k, including the diagonal. Then place the lower-triangular half of S^b on the lower-triangular half of C_k, excluding the diagonal. Finally, place the diagonal of S^b on the last row of C_k. For example, consider a PM code with $k = 4, d = 6$, for which $\alpha = 3$ and number of message-symbols $B = 12$. The triangular inclusion map f_ι for this case is shown in Figure 2.

B. The Inverse Map

Lemma 1. *Let $f_e : M \to C_k$ denote the encoding function of a $d = (2k-2)$ PM code for the first k nodes. Then, in the inverse transform $f_e^{-1} : C_k \to M$, the first column of M depends only on the first row and first column of C_k.*

C. Sparsity

Here we combine the above maps and show that the entire encoding transform has a certain sparsity.

Lemma 2. *Let $f_e : M \to C_k$ denote the encoding function of a $d = (2k-2)$ PM code for the first k nodes. Let f_S denote the systematic-remapping transform defined by $f_S = f_e^{-1} \circ f_\iota$, where f_ι is the* triangular inclusion map *defined in Section III-A. Then, in the systematic-remapping $f_S : M \to \overline{M}$, a symbol in the first column of $\overline{M}$ only depends on symbols in the first column of M.*

Proof (sketch).

(1) The "decoding", $f_e^{-1} : C_k \to \overline{M}$ is such that the first column of $\overline{M}$ depends only on the first row and first column of C_k (from Lemma 1).
(2) Our design of the triangular inclusion map $f_\iota : M \hookrightarrow C_k$ is such that the symbols in the first row/column of C_k correspond exactly to the first column of M (Section III-A).

□

The resulting sparsity pattern of systematic remapping (Lemma 2) is visualized in Figure 3.

Using the above two Lemmas, we can prove partial sparsity in the entire encoding process as stated below.

Theorem 1. *Consider a $d = (2k-2)$ PM code in which the first row of Φ is $e_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$. When this code is made systematic, the first symbol stored in every node is d-sparse.*

IV. Explicit Sparse, Systematic PM Codes for $d = (2k-2)$

Here we extend the techniques of the previous section to present sparse systematic PM codes for $d = (2k-2)$. In this version, we only present the main ideas and results; all proofs and a more in-depth analysis are in the full version [13].

Consider a $d = (2k-2)$ PM code in which the first α rows of Φ form an Identity matrix. In this case, the encoding matrix for the first k nodes will take the form:

$$\Psi_k = \begin{bmatrix} I & \Lambda \\ r^T & \lambda r^T \end{bmatrix} \tag{8}$$

where r is an α-length vector. Then, analogous to Lemma 2, we can show that the systematic-remapping transform has a certain sparsity.

Lemma 3. *Let $f_e : M \to C_k$ denote the encoding function of a $d = (2k-2)$ PM code for the first k nodes. Let f_S denote the systematic-remapping transform defined by $f_S = f_e^{-1} \circ f_\iota$, where f_ι is the* triangular inclusion map *defined in Section III-A. Then, in the systematic-remapping transform $f_S : M \to \overline{M}$, the symbol $\overline{M}_{i,j}$ only depends on symbols in column j of M.*

This allows us to prove sparsity in the entire encoding process as stated below.

Theorem 2. *Consider a $d = (2k-2)$ PM code in which the first α rows of Φ form an Identity matrix. When this code is made systematic, each encoded symbol is d-sparse.*

We construct encoding matrices Ψ which conform to the structure of Theorem 2 as follows. This yields explicit sparse PM codes for $d = (2k-2)$.

Theorem 3. *Let $\Psi = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix}$ be the encoding matrix for a $d = (2k-2)$ PM code, satisfying all the necessary properties*

(e.g., a Vandermonde matrix as in [5]). Let the $(\alpha \times \alpha)$ matrix Φ_α denote the first α rows of Φ. Then the following encoding matrix

$$\Psi' = \begin{bmatrix} \Phi\Phi_\alpha^{-1} & \Lambda\Phi\Phi_\alpha^{-1} \end{bmatrix} := \begin{bmatrix} \Phi' & \Lambda\Phi' \end{bmatrix} \quad (9)$$

defines a $d = (2k-2)$ PM code in which, after systematic remapping, each encoded symbol is d-sparse.

In other words, if we represent the encoding procedure for this systematic code as a $(n\alpha \times B)$ generator matrix G mapping B message symbols to $n\alpha$ encoded symbols (α per node), then each row of G will be d-sparse.

V. Sparsity in Systematic MSR Codes from Repair-By-Transfer

Sections III and IV dealt with constructing sparse systematic PM codes. In this section, we present general results on the sparsity of systematic regenerating codes.

A. Background: Repair-by-Transfer

During a node-repair operation, a helper node is said to perform *repair-by-transfer (RBT)* if it does not perform any computation and merely transfers one of its α stored symbols to the failed node. We say a linear $[n, k, d](\alpha, \beta = 1)$ MSR code supports RBT with the *RBT-SYS pattern* if every node can help the first α nodes via RBT.

B. Sparsity from Repair-by-Transfer

We now present a general connection between sparsity and repair-by-transfer, by showing that an MSR code with a certain RBT property must necessarily be sparse.

Theorem 4. *Let $\mathcal{C}$ be a linear systematic MSR $[n, k, d](\alpha, \beta = 1)$ code of blocklength $B = k\alpha$, with $(n\alpha \times B)$ generator matrix G. Let $G^{(i)}$ be the $(\alpha \times B)$ submatrix corresponding to the i-th node. If $\mathcal{C}$ supports repair of a systematic node ν via RBT with helper nodes comprising the remaining $(k-1)$ systematic nodes and $d - (k-1) = \alpha$ other parity nodes, then for each parity i, the corresponding generator-submatrix $G^{(i)}$ has one row with sparsity $\leq d$.*

In particular, the row of $G^{(i)}$ corresponding to the symbol transferred for the repair of node ν is supported on at most the following coordinates:

- *The α coordinates corresponding to symbols stored by node ν.*
- *For each of the remaining $(k-1)$ participating systematic nodes $\mu \neq \nu$: one coordinate corresponding to a symbol stored by node μ.*

Proof (sketch). Say systematic node ν fails, and is repaired via RBT by the $(k-1)$ other systematic nodes, and α other parity nodes. Intuitively, Theorem 4 holds because the symbols from systematic helpers can only "cancel interference" in $(k-1)$ coordinates, and the α parity helpers must allow the repair of node ν's α coordinates, and thus cannot contain more interference. □

As a corollary, MSR codes which support the *RBT-SYS pattern* have the following sparsity.

Theorem 5. *If $\mathcal{C}$ supports the* RBT-SYS pattern*, then for each parity i, the corresponding generator-submatrix $G^{(i)}$ has $\min(\alpha, k)$ rows that are d-sparse. In particular, if $d \leq (2k-1)$, then all rows of G are d-sparse.*

VI. Explicit Sparse, Systematic PM Codes for $d > (2k-2)$

Section V provides a strong connection between repair-by-transfer and sparsity in systematic MSR codes. This connection allows us to construct explicit systematic PM codes for $d > (2k-2)$, in which all encoded symbols are d-sparse (or better). In particular, it is possible to construct PM codes with the *RBT-SYS* pattern, as in [12]. We make use of this construction and the notion of *code shortening* to construct *sparse* $d > (2k-2)$ PM codes from a class of $d = (2k-2)$ PM codes. The following theorem presents this result.

Theorem 6. *Consider a $[n, k, d > (2k-2)]$ systematic PM code $\mathcal{C}$ constructed by shortening a $[n' = n+i, k' = k+i, d' = (2k'-2)]$ systematic PM code $\mathcal{C}'$ that supports RBT-SYS, where $i := (d - (2k-2))$. Let G denote the generator matrix for code $\mathcal{C}$. Letting $G^{(j)}$ denote the $(\alpha \times k\alpha)$ submatrix of G for node j, the following sparsity holds for all nodes j:*

- *The first $(d - 2k + 2)$ rows of $G^{(j)}$ are k-sparse.*
- *The remaining $(k-1)$ rows of $G^{(j)}$ are d-sparse.*

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *ACM SIGOPS*, 2003.

[2] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster," in *Proc. USENIX HotStorage*, 2013.

[3] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing Elephants: Novel Erasure Codes for Big Data," in *VLDB Endowment*, 2013.

[4] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed Storage Codes with Repair-by-Transfer and Non-achievability of Interior Points on the Storage-Bandwidth Tradeoff," *IEEE Transactions on Information Theory*, Mar. 2012.

[5] K. 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," *Information Theory, IEEE Transactions on*, 2011.

[6] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference Alignment in Regenerating Codes for Distributed Storage: Necessity and Code Constructions," *IEEE Transactions on Information Theory*, 2012.

[7] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Transactions on Information Theory*, 2013.

[8] D. Papailiopoulos, A. Dimakis, and V. Cadambe, "Repair Optimal Erasure Codes through Hadamard Designs," *IEEE Transactions on Information Theory*, 2013.

[9] A. G. Dimakis, P. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, 2010.

[10] S. Jiekak, A.-M. Kermarrec, N. Le Scouarnec, G. Straub, and A. Van Kempen, "Regenerating codes: A system perspective," *ACM SIGOPS*, 2013.

[11] N. L. Scouarnec, "Fast Product-Matrix Regenerating Codes," *CoRR*, vol. abs/1412.3022, 2014.

[12] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having Your Cake and Eating It Too: Jointly Optimal Erasure Codes for I/O, Storage, and Network-bandwidth," in *USENIX FAST*, 2015.

[13] P. Nakkiran, K. V. Rashmi, and K. Ramchandran, "Optimal Systematic Distributed Storage Codes with Fast Encoding," *CoRR*, vol. abs/1509.01858, 2015. [Online]. Available: http://arxiv.org/abs/1509.01858