MDR Codes: A New Class of RAID-6 Codes with Optimal Rebuilding and Encoding

Yan Wang, *Student Member, IEEE,* Xunrui Yin, Xin Wang, *Member, IEEE,*

Abstract

As storage systems grow in size, device failures happen more frequently than ever before. Given the commodity nature of hard drives employed, a storage system needs to tolerate a certain number of disk failures while maintaining data integrity, and to recover lost data with minimal interference to normal disk I/O operations. RAID-6, which can tolerate up to two disk failures with the minimum redundancy, is becoming widespread. However, traditional RAID-6 codes suffer from high disk I/O overhead during recovery. In this paper, we propose a new family of RAID-6 codes, the Minimum Disk I/O Repairable (MDR) codes, which achieve the optimal disk I/O overhead for single failure recoveries. Moreover, we show that MDR codes can be encoded with the minimum number of bit-wise XOR operations. Simulation results show that MDR codes help to save about half of disk read operations than traditional RAID-6 codes, and thus can reduce the recovery time by up to 40%.

Index Terms

RAID-6 codes, disk I/O, encoding complexity, distributed storage systems, erasure codes.

Y. Wang, and X. Wang are with the School of Computer Science, Fudan University (e-mail: {11110240029, xinw}@fudan.edu.cn). X. Yin is with the Department of Computer Science, University of Calgary (e-mail: xunyin@ucalgary.ca). He was with Fudan University when the main work was done. Y. Wang is also with the School of Software, East China Jiao Tong University.

I. INTRODUCTION

To satisfy the storage demand of "big data" in data centers, distributed storage systems are typically constructed from a large number of commodity servers and hard drives. As the capacity grows, disk failures happen more frequently than ever before. RAID-6 systems, which can tolerate two disk failures with the minimum redundancy, have been widely used.

Measurement studies in the literature suggest that single disk failures represent 98.08% of recoveries [1]. When there is one or two disk failures, the system has to run at a reduced speed. Hence minimizing the time of single failure recovery is important. Since disk I/O time represents a dominant component in recovery time [2], the most promising approach to improve the recovery performance is to reduce the amount of data read from each disk [3].

In its general specification, RAID-6 does not impose restrictions on the specific coding method. In fact, one may apply any maximum distance separable (MDS) codes that can tolerate 2 erasures, as exemplified by the rather popular MDS array codes with a row parity block on each row. Many such codes have been designed, such as EVENODD[4], RDP[5], Liberation Codes[6]. The row parity block and data blocks stored in the same row are called a *row parity set*. If a single data disk fails, the conventional way of repair is to calculate each block by XORing the blocks remaining on the surviving disks in the row parity set.

However, for the two existing RAID-6 codes, RDP and EVENODD, Xiang *et al.* and Wang *et al.* showed that if the other coding disk is used in the repair, the failed disk can be recovered by reading $3/4$ blocks from each surviving disk [7][8][9]. Furthermore, Tamo *et al.* and En Gad *et al.* showed that the repair disk I/O can be further reduced if the Q disk is designed carefully [10][11].

Take Fig. 1 for example. Disks D_1, D_2 and D_3 are the data disks, each holding 4 un-coded information blocks. Disk D_4 is called a P disk, which holds the row parity of data blocks. Disk D_5 is called a Q disk. Fig. 1(a) shows the conventional way to repair D_1, which requires reading
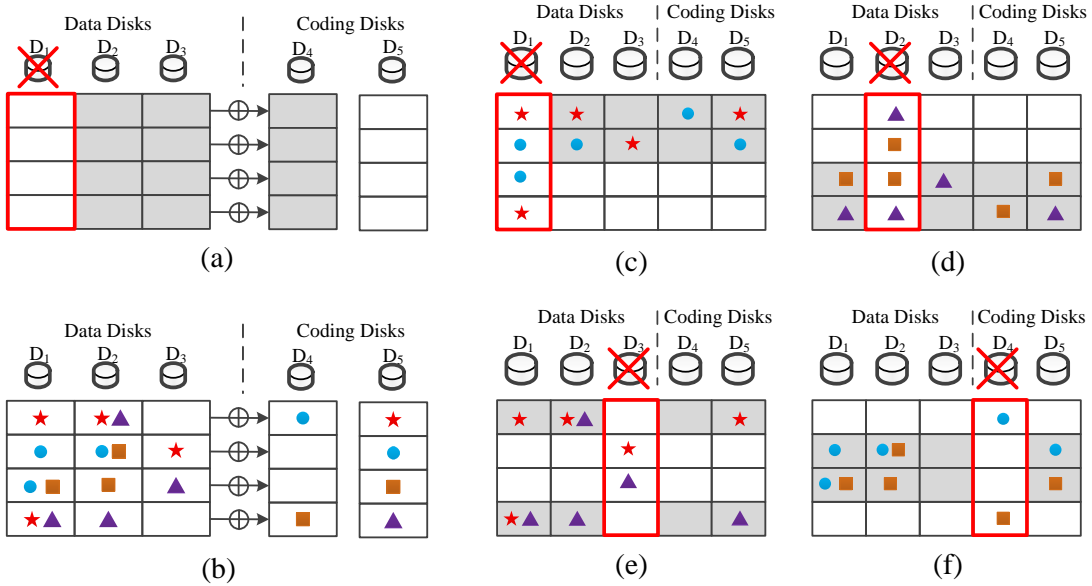
Fig. 1: An example RAID-6 code with minimum disk I/O for a single failure recovery. When a data disk or the row parity disk fails, only half of the blocks in each surviving disks are read in the recovery.

12 blocks. Fig. 1(b) shows a RAID-6 code. The blocks on the Q disk are calculated as the parity of the blocks with the same mark labeled in the figure. Fig. 1(c)-(f) show the repair strategies for a single failure of each disk except the Q disk, where shaded blocks are read to repair the failed disk. For example, as shown in Fig. 1(c), if disk D_1 fails, the first two rows of blocks on the surviving disks are read to memory, so that we can calculate the first two blocks of D_1 by row parities, and then the last two blocks by the parity sets marked with "○" and "⋆", since all the other blocks in the two parity sets are known. As a result, only 8 blocks are read to repair D_1, saving 33.3% disk I/O over conventional repair.

In this work, we study the problem of minimizing disk I/O for the repair of a single disk failure with MDS array codes constructed over $\mathbb{F}_2$, *i.e.*, all coded blocks can be generated with bit-wise XOR operations only. Our contributions include the following:

1) We prove exact lower bounds on the minimum disk I/O: at least $(k+1)r/2$ blocks must be read to recover a data disk or the P disk, and at least kr blocks must be read to repair the Q disk, where k is the number of data disks and r is the number of rows in the array. Furthermore, we prove that in the repair of a data disk or the P disk, $r/2$ blocks of the failed disk must be repaired by row parity, in order to achieve the minimum disk I/O.

2) We develop an equivalent condition (Theorem 4) for the optimal repair in RAID-6 codes. With this condition, we find the example repair-optimal code shown in Fig. 1 and construct the MDR codes, which minimize the repair disk I/O not only for the data disks but also for the coding disks.

3) We show that MDR codes can be encoded with the minimum number of XOR operations. We achieve this by utilizing the intermediate result in the calculation of P disk. To our knowledge, MDR codes represent the first family of codes that minimize both repair disk I/O and computational overhead.

The rest of this paper is organized as follows. We review related literatures in section II and the specification of RAID-6 codes and basic notations in section III. In section IV, we propose a generator matrix approach for studying the minimum disk I/O problem. Along this approach, we prove lower bounds on the minimum disk I/O and develop the equivalent condition for the optimal repair. In section V, we propose the construction of MDR codes. In section VI, we show how to minimize the computational overhead with MDR codes. We discuss the drawbacks of MDR codes in section VII and present the simulation results in section VIII. Section IX concludes this paper.

II. RELATED WORK

In the design of RAID-6 codes, many researchers focus on minimizing the computational overhead of encoding, updating, and decoding. For example, the EVENODD codes [4] achieve near optimal computational complexity in both encoding and decoding, and the RDP codes [5]

further improve updating complexity. Plank proposed the Liberation codes [6] that are freely available and achieve either optimal or close to optimal in the encoding, updating, and decoding complexity.

Recently, reducing the repair disk I/O attracts more and more attentions. Studies on reducing disk I/O can be divided into two classes: one is to develop clever algorithms for existing RAID-6 codes, and the other is to design new RAID-6 codes. For the former class, Xiang *et al.* [7][9] and Wang *et al.* [8] used both parity disks to reduce the disk I/O in single disk failure recovery. They designed efficient recovery algorithms for RDP codes and EVENODD codes. The proposed optimal recovery strategies can reduce approximately $1/4$ disk reads compared with conventional recovery algorithms. Khan *et al.* [3][12] proved that the problem of finding minimum repair disk I/O for a given XOR-based erasure code is NP-hard in general, and Zhu *et al.* [13] proposed a polynomial-time approximation algorithm for this problem.

The problem of designing new RAID-6 codes to optimize repair disk I/O has been studied in the more general context of optimizing disk I/O for distributed storage systems. Inspired by network coding, Dimakis *et al.* [14] proved a lower bound on the minimum bandwidth consumption in the recovery. As the amount of data transmitted is always no more than the amount of data read, the repair disk I/O is at least the minimum repair bandwidth. Therefore, Dimakis' lower bound on the latter directly implies a lower bound on the former, which implies that reading at least $(k + 1)r/2$ blocks is necessary for the repair in RAID-6 codes with k data disks and r rows in the array. According to the study of minimum bandwidth with exact repairs [15], it is impossible for a $(k > 4, r = 2)$ RAID-6 code to achieve this lower bound in the repair of every single disk.

However, Tamo *et al.* and En Gad *et al.* recently showed that the bound $(k+1)r/2$ is achievable for the repair of a data disk. Specifically, Tamo *et al.* proposed the Zigzag MDS array codes that minimize the repair disk I/O [10]. Their codes require coding over a field of size at least 3 and achieve optimal update as well. Furthermore, Zigzag codes have strip size $r = 2^{k-1}$, and they

proved that this strip size is optimal for all systematic, update-optimal and repair-optimal MDS codes. En Gad *et al.* [11] also proposed a family of RAID-6 array codes over $\mathbb{F}_2$ that achieve the optimal repair disk I/O for a data disk recovery.

Our work differs from the above in the following aspects. First, they only optimized disk I/O for the repair of *data disks*, while we consider the repair of *every disk*. Both of their codes require reading kr blocks to repair the row parity disk, but MDR codes require reading only $(k+1)r/2$ blocks. We further prove that the minimum disk I/O to repair the Q disk is at least kr in RAID-6 codes with a row parity disk. Second, MDR codes also minimize the computational overhead, which is not considered in their works. To the best of our knowledge, MDR codes are the first that minimize repair disk I/O and computational overhead at the same time. Third, we proposed a generic approach for constructing repair-optimal RAID-6 codes from an initial code satisfying certain conditions, which can be found by computer search.

Compared with Zigzag codes, a drawback of MDR codes is that we trade-off update disk I/O for restricting coding operations to over $\mathbb{F}_2$, the same as in En Gad's codes. However, recent reports show that there are many archive-style storage systems where update operations are rare. For example, in Windows Azure [16], the storage system is used in an append-only way. We also trade-off the strip size for the optimal encoding complexity — the strip size of MDR codes is twice as much as in the Zigzag codes and En Gad's codes.

TABLE I: Comparison between repair-optimal codes.

	Field Size	#Disk Repairs Improved	Strip Size	Disk I/O in Update	Encoding Complexity
Zigzag codes [10]	≥ 3	k	2^{k-1}	2	—
En Gad's codes[11]	2	k	2^{k-1}	$1/2 \cdot \lfloor k/2 \rfloor + 2$	$k + k/2 \cdot \lfloor k/2 \rfloor$
MDR codes	2	$k+1$	2^k	$(k+7)/4$	$k-1$

We summarize the comparison in Table I, where "#Disk Repairs Improved" refers to the

number of disks that can be repaired with reading $(k+1)r/2$ blocks, "Disk I/O in Update" refers to the average number of parity blocks changed in the update of a data block, and "Encoding Complexity" refers to the number of XORs to compute each block of the Q disk. For encoding complexity, Zigzag codes require $k-1$ additions and up to k multiplications over a field of size at least 3. We note that the optimal encoding complexity of MDR codes is achieved under the condition that the P disk is computed at the same time. The repair disk I/O, encoding complexity, and update disk I/O of MDR codes are analyzed in Sec. V, Sec. VI and Sec. VII, respectively.

III. PRELIMINARIES AND NOTATIONS

A. Erasure Codes and RAID-6 Specification

Erasure codes ensure data reliability by encoding a message of k symbols into n symbols, so that the message can be recovered even if some symbols are lost. An optimal erasure code can tolerate the loss of any $m = n - k$ symbols. We can find such an optimal erasure code in linear codes, which is also called an (n, k)-MDS code. Compared with replication, MDS codes are storage efficient, since replication requires to store mk symbols instead of $k + m$ symbols to provide the same reliability against m erasures.

In the specification of RAID-6, there are $k + 2$ storage nodes each holding the same amount of data, and up to two node failures can be tolerated. A RAID-6 code can therefore be viewed as an $(n = k + 2, k)$-MDS code. RAID-6 requires k data disks to store original information, and hence is a *systematic code*. Two coding disks further store coded data. In this work, we use $D_1, D_2, \cdots, D_k$ to denote the k data disks, and D_{k+1}, D_{k+2} to denote the two coding disks, which are called P disk and Q disk, respectively.

B. Parity Array Coding Technique

In a parity array code, data stored on each disk is grouped into r blocks of equal length, which are called a strip. Bit-wise XOR is applied to these blocks to generate parity blocks. Blocks in
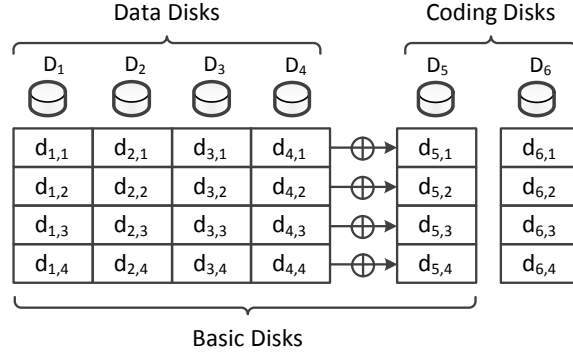
Fig. 2: Block arrangement in RAID-6 codes with a row parity disk.

disks $D_1, D_2, \cdots, D_{k+2}$ are typically arranged into an array of r rows and $k + 2$ columns. Let $d_{i,j}, (1 \leq i \leq k + 2, 1 \leq j \leq r)$ denote the j-th block in disk D_i, and d be the column vector $[d_{1,1} \; d_{1,2} \; \cdots \; d_{k+2,r}]^T$.

Most implementations of RAID-6 array codes use the first coding disk D_{k+1} as a row parity, *i.e.*, $\forall j = 1, \cdots, r, d_{k+1,j} = d_{1,j} + d_{2,j} + \cdots + d_{k,j}$, where addition is over the finite field $\mathbb{F}_2$. This makes it easily extendable from RAID-5 by simply adding another coding disk. Fig. 2 illustrates the general idea of how parity blocks are calculated in such codes. Due to the similarity between the row parity disk D_{k+1} and the data disks $D_1, D_2, \cdots, D_k$, we call them the *basic disks*.

C. Notations

For a positive integer n, we use $[n]$ to represent the set $\{1, 2, \cdots, n\}$. For an m-by-n matrix A, a row index set $R \subset [m]$ and a column index set $C \subset [n]$, we use $A|_{R,C}$ to denote the sub-matrix of A induced by the R rows and C columns. We refer to a RAID-6 code supporting k data disks with r rows in the array as a (k, r) RAID-6 code. For simplicity, we assume r is even.
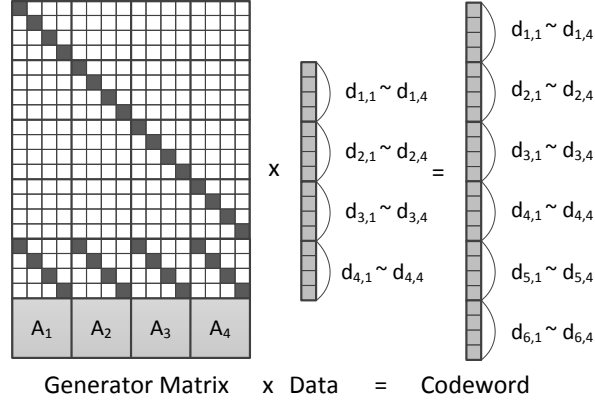
Fig. 3: A RAID-6 code with a row parity disk is uniquely determined by its generator sub-matrices $A_1, A_2, \cdots, A_k$.

IV. THE GENERATOR MATRIX APPROACH

In this section, we use the generator matrix to formulate the problem of minimizing repair disk I/O, and develop an equivalent condition for optimal repair.

Let d_i denote the column vector of the r blocks $[d_{i,1} \ d_{i,2} \ \cdots \ d_{i,r}]^T$ in disk D_i. The generator matrix is illustrated in Fig. 3, where the shaded elements of the matrix are ones and the other elements are zeros. As we consider RAID-6 codes with a row parity disk, we have $d_{k+1} = d_1 + d_2 + \cdots + d_k$. Therefore, to design a RAID-6 code, we only need to specify how the coding disk D_{k+2} is coded. According to the generator matrix, d_{k+2} can be written as

$$d_{k+2} = A_1 d_1 + A_2 d_2 + \cdots + A_k d_k$$

where $A_1, A_2, \cdots, A_k$ are square matrices of size r.

We call $A_1, A_2, \cdots, A_k$ the *generator sub-matrices*. An XOR-based RAID-6 code with a row parity disk is uniquely determined by its generator sub-matrices.

RAID-6 requires that the system can be reconstructed from any two disk failures. According to the study of Blaum and Roth [17], the code described by matrices $A_1, A_2, \cdots, A_k$ satisfies

such RAID-6 specification if and only if $A_1, A_2, \cdots, A_k$ satisfy the following conditions, which we refer to as the MDS property:

- A_i is non-singular for all $i \in [k]$

- $A_i + A_j$ is non-singular for all $i, j \in [k]$, $i \neq j$

A. A Matrix Representation of the Minimum Disk I/O Problem

With two parity disks, each data block $d_{i,j}$ can be represented by the sum of other data blocks in multiple ways. This makes it difficult to find the minimum repair disk I/O. In fact, Khan [3] showed that solving this problem for an XOR-based code is NP-hard in general. In this subsection, we rewrite this problem in terms of the generator sub-matrices $A_1, A_2, \cdots, A_k$.

The problem of finding minimum disk I/O is essentially the same as representing the lost data blocks by a set of surviving data blocks of minimum size. A key observation is that all representations are based on the following equation in terms of parity-check matrix H

$$
Hd = \begin{bmatrix} I & I & \cdots & I & I & 0 \\ A_1 & A_2 & \cdots & A_k & 0 & I \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{k+2} \end{bmatrix} = 0 \tag{1}
$$

Each row of H describes an equation that can be interpreted as a representation of each block involved in it. Take the case of $r = 2, k = 2$ as an example, the first row of the parity-check matrix $\begin{bmatrix} I_{2\times2} & I_{2\times2} & I_{2\times2} & 0 \\ A_1 & A_2 & 0 & I_{2\times2} \end{bmatrix}$ is $[1\ 0\ \ 1\ 0\ \ 1\ 0\ \ 0\ 0]$, which means $d_{1,1} + d_{2,1} + d_{3,1} = 0$ and equivalently, any one of $d_{1,1}, d_{2,1}, d_{3,1}$ can be represented as the sum of the other two. In fact, each representation is equivalent to an equation that can be derived as a linear combination of the rows in equation (1). This observation leads to the following theorem.

Theorem 1: Let $\hat{N}(A)$ denote the number of non-zero columns in matrix A, then the minimum

disk I/O to recover the row parity disk D_{k+1} equals

$$\min_X \hat{N}([I + XA_1 \quad I + XA_2 \quad \cdots \quad I + XA_k \quad X])$$

where X is a square matrix of size r. The minimum disk I/O to recover the coding disk D_{k+2} equals

$$\min_X \hat{N}([X + A_1 \quad X + A_2 \quad \cdots \quad X + A_k \quad X])$$

Proof: According to our previous analysis, every representation is a linear combination of rows in equation (1), which can be described as: $[v_1 \ v_2 \ \cdots \ v_{2r}] Hd = 0$. To recover the disk D_{k+1} is equivalent to represent the lost data d_{k+1} by r equations. Group the equations into the form $V_{r \times 2r} Hd = 0$ and rewrite $V_{r \times 2r} = [Y \ Z]$, where Y, Z are square matrices of size r:

$$[Y \ Z] \begin{bmatrix} I & I & \cdots & I & I & 0 \\ A_1 & A_2 & \cdots & A_k & 0 & I \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{k+2} \end{bmatrix} = \sum_{i=1}^{k}(Y + ZA_i)d_i + Yd_{k+1} + Zd_{k+2} = 0$$

In order to solve d_{k+1}, its coefficient matrix Y must be invertible. Block $d_{i,j}$ is used in the recovery if and only if the j-th column of the coefficient matrix of d_i contains some 1's. Therefore, the total number of blocks used in the recovery equals the number of non-zero columns in the matrix $[Y + ZA_1 \quad Y + ZA_2 \quad \cdots \quad Y + ZA_k \quad Z]$. For a column vector z, $Y^{-1}z$ is a zero vector if and only if $z = 0$, since Y^{-1} is of full rank. Therefore, left-multiplying the matrix with Y^{-1} does not change its number of zero columns, and the case of recovering D_{k+1} is proved with $X = Y^{-1}Z$. For the case of D_{k+2}, Z must be non-singular, and the statement can be derived by left-multiplying the matrix with Z^{-1} and letting $X = Z^{-1}Y$. ∎

To calculate the minimum disk I/O for repairing a data disk $D_i, i \in [k]$, we may first treat disk D_i as the row parity disk by eliminating sub-matrix A_i in the parity-check matrix H, and then applying Theorem 1. For example, consider the case of recovering D_1, we may left-multiply

equation (1) with $\begin{bmatrix} I & 0 \\ A_1 & I \end{bmatrix}$, so that the parity-check matrix H is transformed into:

$$\begin{bmatrix} I & I & \cdots & I & I & 0 \\ 0 & A_2 + A_1 & \cdots & A_k + A_1 & A_1 & I \end{bmatrix}$$

Then the minimum disk I/O can be calculated in a similar way as in Theorem 1.

B. A Lower Bound on the Minimum Recovery Disk I/O

In this subsection, we use the generator matrix approach to prove the lower bound on the disk I/O of repairing a single disk failure in any RAID-6 codes with a row parity disk. Note that Dimakis *et al.* [14] have proved an achievable lower bound on the minimum repair bandwidth for functional repair, which implies that the number of blocks read is at least $(k+1)r/2$. Theorem 2 strengthens this result for exact repair in RAID-6 codes in two aspects: 1) for the repair of the Q disk, we prove that the minimum disk I/O is at least kr; 2) Dimakis' theorem assumes each disk transmits the same amount of information. We drop this assumption and prove that each surviving disk must read at least $r/2$ blocks to repair a basic disk.

Theorem 2: The minimum disk I/O to recover a basic disk $D_i, i \in [k+1]$, is at least $(k+1)r/2$. Further, the amount of data read from each surviving disk must be no less than $r/2$. To recover the coding disk D_{k+2}, the minimum disk I/O is at least kr.

Proof: Firstly, consider the case of repairing disk D_{k+1}. Let X be a matrix that maximizes the number of zero columns in matrix $[I + XA_1 \quad I + XA_2 \quad \cdots \quad I + XA_k \quad X]$. According to the MDS property, for any $i, j \le k, i \neq j$, matrix $\begin{bmatrix} I & I \\ A_i & A_j \end{bmatrix}$ is of full rank, which implies matrix $\begin{bmatrix} I + XA_i & I + XA_j \\ A_i & A_j \end{bmatrix}$ is non-singular, and therefore, the rank of matrix $[I + XA_i \quad I + XA_j]$ must be r. So the total number of zero columns in $[I + XA_i \quad I + XA_j]$ is at most

r. Similarly, as matrix $\begin{bmatrix} I & 0 \\ A_i & I \end{bmatrix}$ has full rank, we can conclude that the total number of zero columns in $[I + XA_i \quad X]$ is no more than r. Let $z_i, i \in [k]$ denote the number of zero columns in $I + XA_i$, and z_{k+1} denote the number of zero columns in X. From the following optimization problem:

$$\max \quad \sum_{i=1}^{k+1} z_i$$

$$\text{subject to:} \quad z_i + z_j \leq r \quad \forall i, j \leq k+1, i \neq j$$

we can see that the maximum total number of zero columns is $(k+1)r/2$ for $k \geq 2$, and the optimal value is achieved only with $z_1 = z_2 = \cdots = z_{k+1} = r/2$. According to Theorem 1, the minimum disk I/O is at least $(k+1)r/2$ and is only achieved by reading $r - r/2 = r/2$ blocks from each surviving disk.

Secondly, for the case of repairing a data disk $D_i, i \in [k]$, we may consider disk D_i as the row parity disk with generator sub-matrices $\{A_i, A_j + A_i \mid i, j \in [k], i \neq j\}$. The same result can be concluded in a similar way.

Finally, consider the case of repairing the coding disk D_{k+2}. We can see that the indices of zero columns in $X + A_1, X + A_2, \cdots, X + A_k, X$ can not be the same, since if $X + A_i$ and $X + A_j$ has a zero column at the same position, the matrix $X + A_i + X + A_j = A_i + A_j$ has a zero column, which conflicts with the MDS property that $A_i + A_j$ is nonsingular. Similarly, if $X + A_i$ and X has a zero column at the same position, we will obtain A_i is singular, which conflicts with the MDS property. Therefore, there are at most r zero columns in matrix

$$[X + A_1 \quad X + A_2 \quad \cdots \quad X + A_k \quad X]$$

According to Theorem 1, the minimum disk I/O to repair D_{k+2} is at least kr. ∎

C. An Equivalent Condition for Optimal Repair

From the above analysis, we conclude that the minimum disk I/O to repair coding disk D_{k+2} is kr, which is achieved by reading all data blocks. Thus we only need to consider the case of repairing a basic disk.

In particular, a repair strategy is represented by the set of blocks read from each surviving disk. Let $C_{i,j}$ denote the rows index set of blocks read from disk D_i to repair disk D_j. For example, if $d_{2,1}, d_{2,3}, d_{2,4}$ are read from disk D_2 to repair disk D_1, then $C_{2,1} = \{1, 3, 4\}$. According to Theorem 2, $|C_{i,j}| = r/2$. The following theorem shows that we actually do not need so many sets to describe a repair strategy. In an optimal strategy, the row indices for each basic disk must be the same.

Theorem 3: In the repair of a basic disk $D_j, j \in [k+1]$, the minimum disk I/O is achieved only by reading the same rows of the surviving basic disks, i.e., $C_{1,j} = C_{2,j} = \cdots = C_{j-1,j} = C_{j+1,j} = \cdots = C_{k+1,j}$. Namely, $r/2$ lost blocks must be recovered by row parity.

Proof: Without loss of generality, suppose we are repairing disk D_1. We need to prove that $C_{2,1} = C_{3,1} = \cdots = C_{k+1,1}$. We may regard each block $d_{i,j}$ as a random variable and consider their entropy. For simplicity, assume each block contains only 1bit, *i.e.*, $H(d_{i,j}) = 1$.

In the repair of D_1, let S be the set of blocks read from $D_2, D_3, \cdots, D_{k+1}$. Let T be the set of blocks read from D_{k+2}, and $R_i (i \in [r])$ be the i-th row of basic disks $D_1, D_2, \cdots, D_{k+1}$. Consider the entropy of blocks $D_1 \cup S$. As D_{k+1} is the row parity disk, blocks from different rows are independent. In particular, we have $H(D_1, S) = \sum_{i=1}^{r} H(R_i \cap (D_1 \cup S))$. For each row i, $H(R_i \cap (D_1 \cup S)) = |R_i \cap (D_1 \cup S)| - 1$ only if $D_1 \cup S$ contains the entire row R_i. Otherwise, $H(R_i \cap (D_1 \cup S)) = |R_i \cap (D_1 \cup S)|$. Noting that $|\cap_{l=2}^{k+1} C_{l,1}|$ indicates the number of rows fully contained in $D_1 \cup S$, we have

$$H(D_1, S) = \sum_{i=1}^{r} |R_i \cap (D_1 \cup S)| - |\cap_{l=2}^{k+1} C_{l,1}| = (k+2)r/2 - |\cap_{l=2}^{k+1} C_{l,1}|$$

As $|C_{2,1}| = |C_{3,1}| = \cdots = |C_{k+1,1}| = r/2$, if the row indices $C_{2,1}, C_{3,1}, \cdots, C_{k+1,1}$ are not the

same, $|\cap_{l=2}^{k+1} C_{l,1}| < r/2$ and $H(D_1, S) > (k+1)r/2$. As D_1 can be reconstructed with $S \cup T$,

$$H(S,T) = H(D_1, S, T) \geq H(D_1, S) > (k+1)r/2$$

which is a contradiction since there are only $(k+1)r/2$ blocks in $S \cup T$. Therefore, the row indices $C_{2,1}, C_{3,1}, \cdots, C_{k+1,1}$ must be the same. ∎

Therefore, in the recovery of a basic disk D_j, $j \in [k+1]$, we may use $C_j \subset [r]$ to denote the index set of blocks read from the surviving basic disks and $R_j \subset [r]$ to denote the index set of blocks read from the Q disk. The following theorem rewrites the condition of optimal repair in terms of generator sub-matrices.

Theorem 4: For a $(k \geq 2, r)$ RAID-6 code described by generator sub-matrices $A_1, A_2, \cdots, A_k$, each basic disk $D_i, i \in [k+1]$ can be repaired by a repair strategy $R_i, C_i, |R_i| = |C_i| = r/2$ if and only if there exists a matrix B_{k+1}, such that the series of matrices $B_j = A_j + B_{k+1}, j \in [k]$ and B_{k+1} satisfy the following conditions:

1) For each $i \in [k+1]$, the sub-matrix $B_i|_{R_i, \overline{C_i}}$ is non-singular, and

2) For any $i, j \in [k+1], i \neq j$, $B_j|_{R_i, \overline{C_i}} = 0$.

Proof: Please refer to the appendix for the proof. ∎

According to the definition of $B_i, i \in [k]$ in Theorem 4, the generator sub-matrices $A_i, i \in [k]$ can be derived as $A_i = B_i + B_{k+1}$. Therefore, a RAID-6 code can also be described by the $B_i, i \in [k+1]$ matrices. Note that the series of B_i matrices is not unique for a RAID-6 code. Theorem 4 states that if a RAID-6 code can be repaired with minimum disk I/O, it must has a series of B_i matrices satisfying the two conditions 1) and 2).

Fig. 4 illustrates this idea. We represent the example RAID-6 code shown in Fig. 1 with B_1, B_2, B_3, B_4. The repair strategy for disk D_1 is reading the first two rows, thus $R_1 = C_1 = \{1, 2\}$ and $\overline{C_1} = \{3, 4\}$. As the sub-matrix $B_1|_{R_1, \overline{C_1}}$ is non-singular and the corresponding sub-matrices of B_2, B_3 and B_4 are zero, D_1 can be repaired with optimal disk I/O.
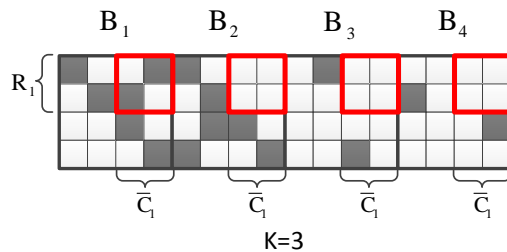
Fig. 4: An example illustrating the equivalent condition of optimal repair.

Recall that $d_{k+1} = \sum_{i=1}^{k} d_i$, we have

$$d_{k+2} = \sum_{i=1}^{k} A_i d_i = \sum_{i=1}^{k} B_i d_i + B_{k+1} \sum_{i=1}^{k} d_i = \sum_{i=1}^{k+1} B_i d_i$$

As for $i, j \in [k]$, $A_i + A_j = B_i + B_j$, the MDS property is equivalent to

- $B_i + B_j$ is non-singular for all $i, j \in [k+1], i \neq j$

V. CONSTRUCTION OF MDR CODES

In this section, we propose the recursive construction approach that derives MDR codes. The approach starts with a repair-optimal RAID-6 code with small k and r that satisfies the following condition:

P1) For each $i \in [k-1]$, B_i is non-singular;

P2) $R_i = C_i$, $\forall i \in [k+1]$.

We will construct a $(k' = k+1, 2r)$ RAID-6 code with optimal repair disk I/O which also satisfies conditions (P1) and (P2). Denote the new code with matrices B_i', $i \in [k'+1]$, and repair

strategy $R_i', C_i', i \in [k'+1]$.

$$
B_i' = \begin{cases}
\begin{bmatrix} B_i + B_{k+1} & 0 \\ 0 & B_i + B_{k+1} \end{bmatrix} & \text{if } i \in [k'-1] \\[2em]
\begin{bmatrix} 0 & I_{r \times r} \\ 0 & 0 \end{bmatrix} & \text{if } i = k' \\[2em]
\begin{bmatrix} 0 & 0 \\ I_{r \times r} & 0 \end{bmatrix} & \text{if } i = k'+1
\end{cases}
$$

$$
R_i' = C_i' = \begin{cases}
\{x | x \in R_i \text{ or } x - r \in R_i\} & \text{if } i \in [k'-1] \\[1em]
[r] & \text{if } i = k' \\[1em]
[2r] \backslash [r] & \text{if } i = k'+1
\end{cases}
$$

Theorem 5: The constructed $(k' = k+1, 2r)$ code is a RAID-6 code with optimal repair disk I/O and satisfies conditions (P1) and (P2).

Proof: We first prove the constructed code $B_i', i \in [k'+1]$, is a RAID-6 code by showing that $B_i' + B_j'$ is non-singular for any $i, j \in [k'+1], i \neq j$. As the code B_i is a RAID-6 code, $B_i + B_j$ has full rank for any $i, j \in [k+1], i \neq j$. The possible results of $B_i' + B_j'$ are $\begin{bmatrix} B_i + B_j & 0 \\ 0 & B_i + B_j \end{bmatrix}$, $\begin{bmatrix} B_i + B_{k+1} & I_{r \times r} \\ 0 & B_i + B_{k+1} \end{bmatrix}$, $\begin{bmatrix} B_i + B_{k+1} & 0 \\ I_{r \times r} & B_i + B_{k+1} \end{bmatrix}$, $\begin{bmatrix} 0 & I_{r \times r} \\ I_{r \times r} & 0 \end{bmatrix}$, which are all non-singular. Therefore, the constructed code is a RAID-6 code.

Second, we use Theorem 4 to show that the constructed code can recover from single disk failure with repair strategy $R_i', C_i', i \in [k'+1]$. According to the hypothesis, $R_i, C_i, i \in [k+1]$ is the optimal recover strategy for code B_i, we have $B_{k+1}|_{R_i, \overline{C_i}} = 0$ for $i \in [k]$ by Theorem 4.
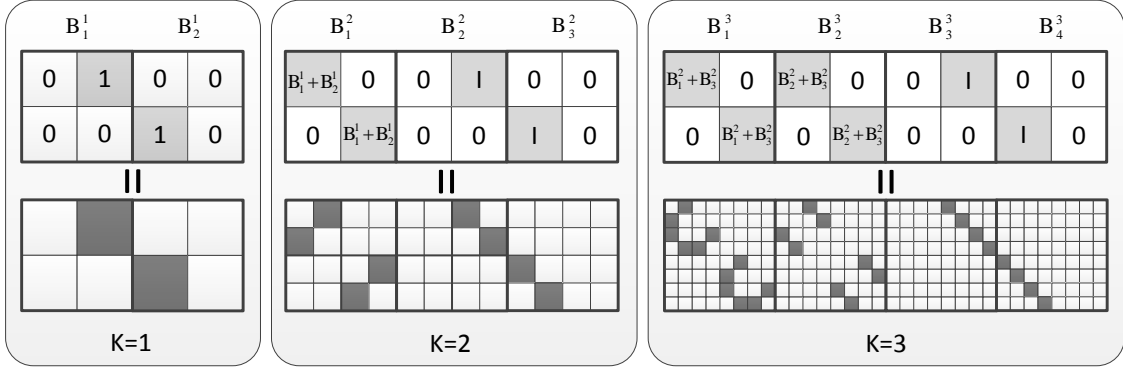
Fig. 5: The construction of MDR codes.

For $i, j \in [k]$,

$$B'_j|_{R'_i, \overline{C'_i}} = \begin{bmatrix} B_j|_{R_i, \overline{C_i}} + B_{k+1}|_{R_i, \overline{C_i}} & 0 \\ 0 & B_j|_{R_i, \overline{C_i}} + B_{k+1}|_{R_i, \overline{C_i}} \end{bmatrix} = \begin{bmatrix} B_j|_{R_i, \overline{C_i}} & 0 \\ 0 & B_j|_{R_i, \overline{C_i}} \end{bmatrix}$$

which is non-singular if $i = j$, and zero matrix otherwise. For $i \in [k'-1]$, as $R_i = C_i$, $I|_{R_i, \overline{C_i}} = 0$.

Thus $B'_{k+1}|_{R'_i, \overline{C'_i}} = \begin{bmatrix} 0 & I|_{R_i, \overline{C_i}} \\ 0 & 0 \end{bmatrix} = 0$. For the same reason, we have $B'_{k'+1}|_{R'_i, \overline{C'_i}} = 0$. Thus disk

$D_i, i \in [k'-1]$ can be recovered by the strategy R'_i, C'_i. For the case of $i = k'$ and $i = k'+1$,

we can see that R'_i, C'_i satisfy the optimal repair conditions of Theorem 4.

Finally, condition (P2) is directly satisfied from the construction, and condition (P1) is also

satisfied, since for $i \in [k'-1]$, $B'_i = \begin{bmatrix} B_i + B_{k+1} & 0 \\ 0 & B_i + B_{k+1} \end{bmatrix}$ is non-singular according to

the previous analysis. ∎

In order to generate MDR codes with this approach, we need an initial repair-optimal RAID-6

code satisfying (P1) and (P2). The following $(k = 1, r = 2)$ RAID-6 code can be applied:

$$B_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad , \quad B_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$R_1 = C_1 = \{1\} \quad , \quad R_2 = C_2 = \{2\}$$

We can verify that this code satisfies the requirement of our approach. The resulting MDR codes are shown in Fig. 5 for the case $k = 1, 2, 3$.

VI. ACHIEVING THE MINIMUM ENCODING OVERHEAD WITH MDR CODES

Besides disk I/O overhead, another important metric of a RAID-6 code is coding complexity. In this section, we will show that we can achieve the minimum encoding complexity with MDR codes.

A. Encoding Complexity

The encoding complexity considers the number of XORs to generate the coding disks P, Q. As the row parity disk P can be directly computed using $k - 1$ XORs for each block, we only need to consider computing the Q disk. A direct way to compute the Q disk is to calculate $d_{k+2} = \sum_{i=1}^{k} A_i d_i$. But with the knowledge of row parity disk, we may calculate the Q disk with fewer XOR operations.

We accomplish this in a recursive way. Let y_t denote the number of XORs to calculate the Q disk in the case $k = t$. Let $B_i, i \in [t+1]$ represent the $(t-1, 2^{t-1})$ MDR code and $B'_i, i \in [t]$ represent the $(t, 2^t)$ MDR code constructed from B_i. Let $d'_i, i \in [t+2]$ denote the blocks in disk D_i,

$$
d'_{t+2} = \sum_{i=1}^{t-1} \begin{bmatrix} B_i & 0 \\ 0 & B_i \end{bmatrix} d'_i + \begin{bmatrix} B_t & 0 \\ 0 & B_t \end{bmatrix} \sum_{i=1}^{t-1} d'_i + \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} d'_t + \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix} d'_{t+1}
$$

Carefully checking the above formula, we can see that calculating the upper half of the first two terms is equivalent to calculating the Q disk in the $(t-1, 2^{t-1})$ MDR code, with the upper half of $d'_i, i \in [t-1]$ as the data blocks. According to the induction hypothesis, if we know the corresponding P disk of the $(t-1, 2^{t-1})$ MDR code, we can calculate the first two terms with $2y_{t-1}$ XORs. Fortunately, we are able to acquire this information, *i.e.,* the value of $\sum_{i=1}^{t-1} d'_i$, in the calculation of the P disk of the $(t, 2^t)$ MDR code. Thus, we have

$$
y_t = 2y_{t-1} + 2^{t-1} + 2^{t-1}
$$

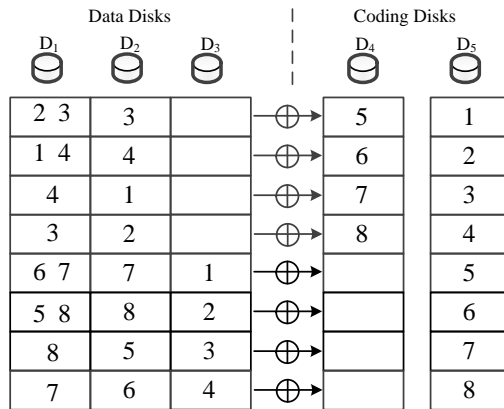| Data Disks			Coding Disks	
D_1	D_2	D_3	D_4	D_5
2 3	3		5	1
1 4	4		6	2
4	1		7	3
3	2		8	4
6 7	7	1		5
5 8	8	2		6
8	5	3		7
7	6	4		8

Fig. 6: The MDR code with $k = 3$. The Q disk is calculated as the parity of blocks containing the same number.

Note that for the initial case $k = 1$, blocks in the P disk and Q disk are replications of data blocks, which means $y_1 = 0$. Solving this recursive equation, we obtain $y_t = (t - 1)2^t$. As there are $r = 2^t$ blocks in the coding disk, the average number of XORs to compute one coded block is $k - 1$, which is optimal [6].

Example. Consider the $(k = 3, r = 8)$ MDR code shown in Fig. 6. We use this example to explain how the Q disk can be calculated with $k - 1 = 2$ XORs for each block. For the first block in the Q disk, we may directly compute it with $d_{5,1} = d_{1,2} + d_{2,3} + d_{3,5}$, which takes 2 XORs. For the third block in the Q disk, $d_{5,3} = d_{1,1} + d_{1,4} + d_{2,1} + d_{3,7}$. As we may catch the intermediate result $d_{1,1} + d_{2,1}$ in the computation of P disk, we can see that $d_{5,3}$ can also be computed with 2 XORs.

B. Recovery Complexity

The recovery complexity counts the average number of XORs to regenerate a failed block in a single disk failure recovery.

Theorem 6: With MDR codes, a failed basic disk can be recovered by $k - 1$ XORs for

computing each lost block.

Proof: Please refer to the appendix for the proof. ∎

Note that if only the Q disk fails, we are currently unable to recover it with $k - 1$ XORs, since applying the method in Sec. VI-A requires computing the P disk at the same time, which results in $2(k - 1)$ XORs for rebuilding each block of the Q disk.

VII. DISCUSSIONS

We have seen in previous sections that MDR codes achieve optimal repair disk I/O and encoding complexity. To achieve this, however, we need a large update disk I/O and a large strip size, which will be discussed in this section.

A. Update Disk I/O

Update disk I/O refers to the number of parity blocks that are affected by changing the content of a data block. As the number may vary for updating different data blocks, we focus on the average value here. In terms of the parity-check matrix H, the update disk I/O equals the average number of ones in each column of H.

For a $(k, 2^k)$ MDR code, let x_k denote the average number of ones in each column of matrices $A_i = B_i + B_{k+1}, i \in [k]$. Then $k2^k x_k$ is the total number of 1's in the matrix $[A_1 \ A_2 \ \cdots \ A_k]$. According to the construction in Sec. V, we have

$$k2^k x_k = 2(k-1)2^{k-1}x_{k-1} + (k+1)2^{k-1}$$

And for the case of $k = 1$, we have $x_1 = 1$. Solving this recursive equation,

$$
\begin{aligned}
kx_k &= (k-1)x_{k-1} + \frac{k+1}{2} \\
&= (2-1)x_1 + \sum_{i=2}^{k} \frac{i+1}{2} = \frac{1}{2}(k(k+1)/2 + k)
\end{aligned}
$$

we obtain $x_k = (k + 3)/4$ for $k \geq 2$. Note that x_k is actually the average number of parity blocks in the Q disk that are affected by the updating. As there is a row parity block affected as well, the update disk I/O of MDR codes is $x_k + 1 = (k + 7)/4$.

B. Strip Size

Here we use the term "strip size" to denote the number of rows in a RAID-6 array code. According our construction, the MDR codes have strip size 2^k. The other two repair-optimal codes, Zigzag codes [10] and En Gad's codes[11], have strip size 2^{k-1}. Tamo *et al.* proved that, in order to achieve both optimal repair and optimal update at the same time, the minimum strip size is 2^{k-1} [10]. If the assumption of optimal updating is dropped, however, the minimum strip size for optimal repair is unknown yet. Inspired by the works [10, 11], we carried out a brute-force search (based on Theorem 4) for the minimum strip size and verified that the minimum strip size is indeed 2^{k-1} for $2 \leq k \leq 4$. For $k = 5$, the strip size is no less than $2^{k-1} - 2 = 14$. We conjecture that for a repair-optimal RAID-6 code over $\mathbb{F}_2$, the minimum strip size is at least exponential to the number of data disks k.

In practice, the impact of large strip sizes is that we need a large memory to cache these blocks during the repair process. We note that, for a $(k, r = 2^k)$ MDR code, we may carry out the repair by caching only $r/2 + 2$ blocks, because $r/2$ lost blocks are recovered by row parity, which can be computed one by one, with one block for the intermediate result and another for the current block read from the disk. Assume the block size is set to 512 Bytes [18], the memory overhead of MDR codes with 16 data disks is about $512\text{B} \times 2^{16}/2 = 16\text{MB}$, which is acceptable.

VIII. SIMULATION

A. Simulation Setup

To evaluate the performance of MDR codes, we use the popular disk simulator Disksim [19] to simulate the recovery process of a RAID-6 system.

In the simulation, we set up 10 disks in all, which are connected to an interleaved I/O bus. Our simulation module acts as an I/O driver of a RAID-6 system, which directly generates I/O requests and handles the request completion interrupts from each individual disk. The logical layout is rotated among stripes of the disk array. As I/O tasks on different disks can be executed simultaneously, we pipeline the recovery of sequential stripes, *i.e.*, we write the recovered blocks of the last stripe at the same time of reading blocks of the current stripe. To simulate the workload during an online repair, we also generate random I/O requests to surviving disks.

We implement three different recovery algorithms: the conventional recovery algorithm that uses the row parity, the RDOR [7] recovery algorithm for the RDP code and our recovery algorithm for the MDR codes. Performance of the conventional recovery algorithm is used as a benchmark, so that we use the performance ratio of the other recovery algorithms to the conventional algorithm to measure the improvements.

Two metrics are tested. One is recovery time, and the other is the average access time of the surviving disks, which is measured as the sum of the response time of each I/O request and thus reflects the load on each disk.

B. Impact of Strip Size

We vary the block size from 512B to 8KB, so that the strip size changes from 32KB to 512KB. Simulations are carried out with 8 disks in the array, and both online repair mode and offline repair mode are tested.

Fig. 7(a) shows the average access time with different strip sizes. As disk access time of the recovery process is not affected by I/O requests of the other application process, we only show the result of online repair mode. We can see that, compared with the conventional repair algorithm, the RDOR algorithm reduces access time to about $80\% \sim 85\%$, and our codes further reduce the access time to $65\% \sim 70\%$. As strip size grows, access times decrease for all the three recovery algorithms due to sequential reads, but the improvement ratio does not exhibit
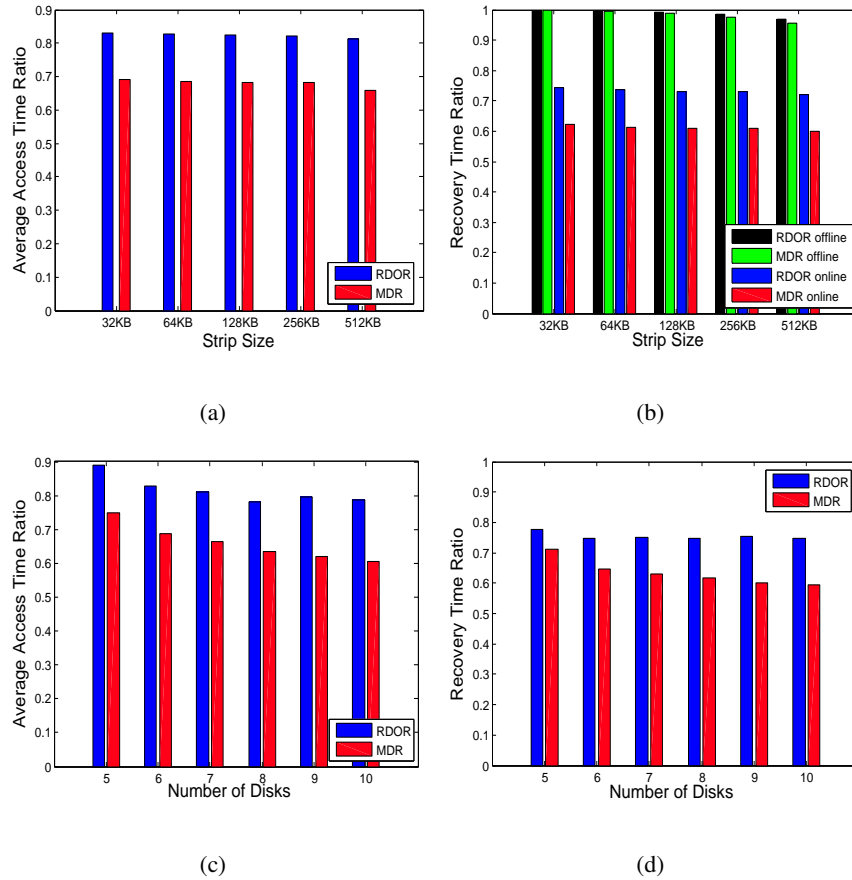
(a)

(b)

(c)

(d)

Fig. 7: Simulation results.

any characterizable trend.

Fig. 7(b) shows the recovery time with different strip sizes. When the recovery process is carried out in the off-line mode, both the RDOR and our recovery algorithm can hardly reduce the recovery time. This is because we pipeline the disk read and write, and multiple disk reads at different disks can be executed simultaneously. Hence disk write becomes the bottleneck. The recovery time ratio decreases slightly as strip size grows. This is because we can not pipeline for recovering the first strip, whose recovery time is reduced by reading less blocks from the surviving disks.

C. Impact of the Number of Disks

We increase the number of disks from 5 to 10 to evaluate its impact on the performance of these recovery methods.

Fig. 7(c) shows the online average access time with different number of disks. As the number of disks grows, the access time of RDOR does not show a tendency, while the access time of our codes decreases. This is because the ratio of read disk reads by RDOR depends on the smallest prime that is larger than the number of systematic disks k and at the proper primes, the ratio approaches 75% as k increases. On the other hand, our ratio approaches 50% as k increases. For the recovery time, Fig. 7(d) shows a similar result.

IX. CONCLUSION

We studied the problem of minimizing disk I/O for every single disk repair in a RAID-6 system, including not only the repair of a data disk but also the repair of a coding disk. We solved this problem by proving a lower bound on the minimum repair disk I/O and constructing the MDR codes that achieve this bound. We also showed that the MDR codes achieve the minimum computational overhead among all RAID-6 codes when the P disk is calculated at the same time. The construction approach is a generic one, which may be used to generate new repair-optimal RAID-6 codes with different initial codes. The main drawback of MDR codes is that its strip size is 2^k, which is twice as much as that of Zigzag codes and En Gad's codes. Inspired by these codes, we recently modified the construction method for MDR codes and constructed a new family of RAID-6 codes with strip size 2^{k-1}, which achieve the optimal repair disk I/O in the repair of every disk and can be encoded with k XORs per each coding block.

We implemented our codes and tested their performance through simulations. Results show that our codes can efficiently reduce the reading overhead of surviving disks to about half of that in the conventional way, and the total recovery time can be reduced by up to 40%.

REFERENCES

[1] K. 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 *5th USENIX Workshop on Hot Topics in Storage and File Systems(HotStorage)*, Jun. 2013.

[2] "Disk I/O bottleneck." [Online]. Available: http://www.enterprisestorageforum.com/techno\ logy/features/article.php/3856121/IO-Bottlenecks-Biggest-Threat-to-Data-Storage.htm

[3] O. Khan, R. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proceedings of the 10th Usenix Conference on File and Storage Technologies(FAST)*, Feb. 2012.

[4] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[5] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies(FAST)*, Mar. 2004.

[6] J. S. Plank, "The RAID-6 liberation codes," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies(FAST)*, Feb. 2008.

[7] L. Xiang, Y. Xu, J. C. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems(SIGMETRICS)*, Jun. 2010.

[8] Z. Wang, A. Dimakis, and J. Bruck, "Rebuilding for array codes in distributed storage systems," in *Proceedings of GLOBECOM Workshops (GC Wkshps)*, Dec. 2010.

[9] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using RAID-6 codes: Algorithms and performance evaluation," *ACM Transactions on Storage (TOS)*, vol. 7, no. 3:11, Oct. 2011.

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

[11] E. En Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-optimal MDS array codes over GF(2)," in *Proceedings of International Symposium on Information Theory Proceedings (ISIT)*, Jul. 2013.

[12] O. Khan, R. Burns, J. Plank, and C. Huang, "In search of I/O-optimal recovery from disk failures," in *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*, Jun. 2011.

[13] Y. Zhu, P. P. Lee, Y. Hu, L. Xiang, and Y. Xu, "On the speedup of single-disk failure recovery in XOR-coded storage systems: Theory and practice," in *Proceedings of IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, Apr. 2012.

[14] A. G. Dimakis, P. B. Godfrey, M. J. W. Y. Wu, and K. Ram-chandran, "Network coding for distributed storage system," in *Proceedings of IEEE Conference on Computer Communications(INFOCOM)*, May 2007.

[15] 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*, vol. 58, no. 4, pp. 2134 – 2158, Apr. 2012.

[16] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci *et al.*, "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, Oct. 2011.

[17] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE transactions on Information Theory*, vol. 45, no. 1, pp. 46 – 59, Jan. 1999.

[18] "EMC CLARiiON RAID 6 Technology." [Online]. Available: http://www.emc.com/colla\teral/hardware/white-papers/h2891-clariion-raid-6.pdf

[19] S. S. J. Bucy, J. Schindler and G. Ganger, "The disksim simulation environment (v4.0)." [Online]. Available: http://www.pdl.cmu.edu/DiskSim

APPENDIX

A. Proof of Theorem 4

Proof: For the "if" part: Consider the repair of disk $D_i, i \in [k+1]$. As we read the C_i rows of surviving basic disk, we can rebuild the C_i blocks of D_i with row parity. For a r-dimension column vector d and a index set $C \subset [d]$, we use $d|_C$ to denote the column vector formed by the C elements of d. According to the definition of $B_j, j \in [k+1]$, we have $d_{k+2} = \sum_{j=1}^{k+1} B_j d_j$. Consider the R_i rows of this equation:

$$d_{k+2}|_{R_i} = \sum_{j=1}^{k+1}(B_j|_{R_i,C_i}d_j|_{C_i} + B_j|_{R_i,\overline{C_i}}d_j|_{\overline{C_i}}) = B_i|_{R_i,\overline{C_i}}d_i|_{\overline{C_i}} + \sum_{j=1}^{k+1}B_j|_{R_i,C_i}d_j|_{C_i}$$

where the second equality is because $B_j|_{R_i,\overline{C_i}} = 0$ for $j \neq i$. As the blocks $d_{k+2}|_{R_i}$, $d_j|_{C_i}$ are read (or calculated by row parity) and $B_i|_{R_i,\overline{C_i}}$ is non-singular, we can see that blocks $d_i|_{\overline{C_i}}$ can be recovered.

For the "only if" part: Assume there is a (k,r) repair-optimal RAID-6 code described by the generator sub-matrices $A_1, A_2, \cdots, A_k$ with repair strategy $R_i, C_i, i \in [k+1]$. Let $A_{k+1} = 0$ for simplicity. As disk D_i can be recovered by reading R_i blocks from the Q disk and C_i blocks from each surviving basic disk, d_i can be represented by these read blocks. Similar to the proof of Theorem 1, as every representation is derived as a linear combination of rows of equation (1), we can use matrix $[Y Z]$ to denote the linear combination such that $Y + ZA_i = I$ and d_i is solved through the equation $\sum_{j=1}^{k+1}(Y + ZA_j)d_j + Zd_{k+2} = 0$. Let E denote the universal

set $\{1, 2, \cdots, r\}$. In this equation, the coefficients of unread blocks must be zero, which means $(Y + ZA_j)|_{E,\overline{C_i}} = 0$ for $j \in [k+1], j \neq i$, and $Z|_{E,\overline{R_i}} = 0$. Subsequently, for $j \in [k+1], j \neq i$

$$
\begin{aligned}
(Y + ZA_j)|_{E,\overline{C_i}} &= Y|_{E,\overline{C_i}} + Z|_{E,\overline{R_i}} A_j|_{\overline{R_i},\overline{C_i}} + Z|_{E,R_i} A_j|_{R_i,\overline{C_i}} \\
&= Y|_{E,\overline{C_i}} + Z|_{E,R_i} A_j|_{R_i,\overline{C_i}} = 0
\end{aligned}
$$

Therefore, $Z|_{E,R_i} A_j|_{R_i,\overline{C_i}} = Y|_{E,\overline{C_i}} = (I + ZA_i)|_{E,\overline{C_i}}$. Consider the $\overline{C_i}$ rows of the this equation,

$$
\begin{aligned}
Z|_{\overline{C_i},R_i} A_j|_{R_i,\overline{C_i}} &= I|_{\overline{C_i},\overline{C}} + Z|_{\overline{C_i},E} A_i|_{E,\overline{C_i}} \\
&= I_{r/2 \times r/2} + Z|_{\overline{C_i},R_i} A_i|_{R_i,\overline{C_i}} + Z|_{\overline{C_i},\overline{R_i}} A_i|_{\overline{R_i},\overline{C_i}} \\
&= I_{r/2 \times r/2} + Z|_{\overline{C_i},R_i} A_i|_{R_i,\overline{C_i}}
\end{aligned}
$$

where the last equality is because $Z|_{E,\overline{R_i}} = 0$. Namely, we obtain that for $j \in [k+1], j \neq i$,

$$
Z|_{\overline{C_i},R_i} (A_j + A_i)|_{R_i,\overline{C_i}} = I_{r/2 \times r/2}
$$

which means the sub-matrix $(A_j + A_i)|_{R_i,\overline{C_i}}$ are invertible and the same for $j \in [k+1], j \neq i$. Note that $A_{k+1} = 0$. When $i \neq k+1$, we have $(A_j + A_i)|_{R_i,\overline{C_i}} = (A_{k+1} + A_i)|_{R_i,\overline{C_i}} = A_i|_{R_i,\overline{C_i}}$ for $j \in [k], j \neq i$, which implies $A_j|_{R_i,\overline{C_i}} = 0$ and $A_i|_{R_i,\overline{C_i}}$ is invertible. When $i = k+1$, we have $A_1|_{R_{k+1},\overline{C_{k+1}}} = A_2|_{R_{k+1},\overline{C_{k+1}}} = \cdots = A_k|_{R_{k+1},\overline{C_{k+1}}}$ is invertible.

We define the r-by-r matrix B_{k+1} as $B_{k+1}|_{R_{k+1},\overline{C_{k+1}}} = A_1|_{R_{k+1},\overline{C_{k+1}}}$ and the other parts of B_{k+1} are all zero, i.e., $B_{k+1}|_{\overline{R_{k+1}},E} = 0$ and $B_{k+1}|_{E,C_{k+1}} = 0$. Let $B_i = A_i + B_{k+1}$, for $i \in [k]$. Note that for $i \in [k]$, $B_{k+1}|_{R_i,\overline{C_i}} = 0$ since otherwise $A_1|_{R_i,\overline{C_i}} = A_2|_{R_i,\overline{C_i}} \neq 0$, conflicting the fact $A_j|_{R_i,\overline{C_i}} = 0$ for $i \neq j$. We now verify the series of matrix B_i satisfy the two properties.

Property 1) holds because $B_{k+1}|_{R_{k+1},\overline{C_{k+1}}} = A_1|_{R_{k+1},\overline{C_{k+1}}}$ is non-singular and for each $i \in [k]$, $B_i|_{R_i,\overline{C_i}} = A_i|_{R_i,\overline{C_i}} + B_{k+1}|_{R_i,\overline{C_i}} = A_i|_{R_i,\overline{C_i}}$ is non-singular.

Property 2) holds because for the case $i = k+1$, $j \in [k]$, $B_j|_{R_{k+1},\overline{C_{k+1}}} = A_j|_{R_{k+1},\overline{C_{k+1}}} + A_1|_{R_{k+1},\overline{C_{k+1}}} = 0$; for the case $i \in [k]$, $j = k+1$, $B_{k+1}|_{R_i,\overline{C_i}} = 0$; for the case $i, j \in [k], i \neq j$, $B_j|_{R_i,\overline{C_i}} = A_j|_{R_i,\overline{C_i}} + B_{k+1}|_{R_i,\overline{C_i}} = 0$. ∎

B. Proof of Theorem 6

Proof: For repairing a basic disk $D_i, i \in [k+1]$ with the MDR codes, we can use the row parity to compute block $d_{i,j}$ for $j \in C_i$, which needs $k-1$ XORs. So we only need to consider the case of computing a block $d_{i,j}, j \notin C_i$. Let $d_i|_{\overline{C}_i}$ denote the column vector composed of these blocks.

As shown in the proof of Theorem 4, we use the following equation to calculate $d_i|_{\overline{C}_i}$:

$$d_{k+2}|_{R_i} = B_i|_{R_i, \overline{C}_i} d_i|_{\overline{C}_i} + \sum_{j=1}^{k+1} B_j|_{R_i, C_i} d_j|_{C_i}$$

According to the construction of MDR codes, we can see that $B_i|_{C_i, \overline{C}_i} = I$ holds for the initial case and is preserved in the construction. Let $B_j'', j \in [k]$ be the $(k-1, 2^{k-1})$ MDR codes, R_j'', C_j'' be the corresponding repair strategy. Let y_k denote the number of XORs to compute $\sum_{j=1}^{k+1} B_j|_{R_i, C_i} d_j|_{C_i}$. If $i = k+1$,

$$\sum_{j=1}^{k+1} B_j|_{R_i, C_i} d_j|_{C_i} = \sum_{j=1}^{k-1} B_j'' d_j|_{C_i} + B_k'' \sum_{j=1}^{k-1} d_j|_{C_i}$$

which is equivalent to calculating the Q disk of the $(k-1, 2^{k-1})$ MDR codes, which takes $(k-2)2^{k-1}$ XORs according to the analysis in Sec. VI-A. If $i < k+1$,

$$\sum_{j=1}^{k+1} B_j|_{R_i, C_i} d_j|_{C_i} = \sum_{j=1}^{k-1} \begin{bmatrix} B_j''|_{R_i'', C_i''} & 0 \\ 0 & B_j''|_{R_i'', C_i''} \end{bmatrix} d_j|_{C_i} + \begin{bmatrix} B_k''|_{R_i'', C_i''} & 0 \\ 0 & B_k''|_{R_i'', C_i''} \end{bmatrix} \sum_{j=1}^{k-1} d_j|_{C_i}$$

$$+ \begin{bmatrix} 0 & I_{r/2 \times r/2} \\ 0 & 0 \end{bmatrix} d_k|_{C_i} + \begin{bmatrix} 0 & 0 \\ I_{r/2 \times r/2} & 0 \end{bmatrix} d_{k+1}|_{C_i}$$

Note that the first two terms can be calculated recursively with $2y_{k-1}$ XORs. Thus, $y_k = 2y_{k-1} + 2^{k-1}$. Combining the two cases, we have $y_k = (k-2)2^{k-1}$. As $d_i|_{\overline{C}_i} = d_{k+2}|_{R_i} + \sum_{j=1}^{k+1} B_j|_{R_i, C_i} d_j|_{C_i}$, we can see that $d_i|_{\overline{C}_i}$ can be calculate with $y_k + 2^{k-1} = (k-1)2^{k-1}$ XORs. As there are 2^{k-1} blocks in $d_i|_{\overline{C}_i}$, the average number of XORs to repair each block is $k-1$. ∎