A Hybrid Approach of Failed Disk Recovery Using RAID-6 Codes: Algorithms and Performance Evaluation

LIPING XIANG and YINLONG XU,
University of Science and Technology of China
JOHN C.S. LUI,
The Chinese University of Hong Kong
and
QIAN CHANG, YUBIAO PAN, and RUNHUI LI,
University of Science and Technology of China

The current parallel storage systems use thousands of inexpensive disks to meet the storage requirement of applications. Data redundancy and/or coding are used to enhance data availability, e.g., Row-diagonal parity (RDP) and EVENODD codes, which are widely used in RAID-6 storage systems, provide data availability with up to *two disk failures*. To reduce the probability of data unavailability, whenever a single disk fails, disk recovery will be carried out. We find that the conventional recovery schemes of RDP and EVENODD codes for a single failed disk only use one parity disk. However, there are two parity disks in the system, and both can be used for single disk failure recovery. In this paper, we propose a hybrid recovery approach which uses both parities for single disk failure recovery and we design efficient recovery schemes for RDP code (RDOR-RDP) and EVENODD code (RDOR-EVENODD). Our recovery scheme has the following attractive properties: (1) "*read optimality*" in the sense that our scheme issues the smallest number of disk reads to recover a single failed disk and it reduces approximately 1/4 of disk reads compared with conventional schemes; (2) "*load balancing property*" in that all surviving disks will be subjected to the same (or almost the same) amount of additional workload in rebuilding the failed disk.

We carry out performance evaluation to quantify the merits of RDOR-RDP and RDOR-EVENODD on some widely used disks with DiskSim. The off-line experimental results show that RDOR-RDP and RDOR-EVENODD outperform the conventional recovery schemes of RDP and EVENODD codes in terms of total recovery time and recovery workload on individual surviving disk. However, the improvements are less than the theoretical value (approximately 25%), as RDOR-RDP and RDOR-EVENODD change the disk access pattern from purely sequential to a more random one compared with their conventional schemes.

Categories and Subject Descriptors: B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; D.4.2 [**Operating Systems**]: Storage Management—*Secondary storage*

General Terms: Algorithms, Reliability, Theory

Additional Key Words and Phrases: Disk failure, EVENODD code, RAID recovery, RDP code, recovery algorithm

---

This is an expanded version of a paper presented at International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS) 2010.
This work is supported by the National Nature Science Foundation of China under Grant No. 61073038 and the National High Technology Research and Development Program of China under Grant No. 2009AA01A348.
Authors' addresses: L. Xiang, Y. Xu, Q. Chang, Y. Pan, R. Li, School of Computer Science and Technology, University of Science and Technology of China; email:ylxu@ustc.edu.cn; John.C.S. Lui, Department of Computer Science and Engineering, The Chinese University of Hong Kong; email:cslui@cse.cuhk.edu.hk.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

ACM Reference Format:
Xiang, L, Xu, Y., Lui, J. C. S., Chang, Q., Pan, Y., Li, R. 2011. A Hybrid Approach of Failed Disk Recovery Using RAID-6 Codes: Algorithms and Performance Evaluation. ACM Trans. Storage V, N, Article A (January YYYY), 34 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Today, we often need to store and process a huge amount of digital information and this makes designing a massive data storage system more challenging. In [Lyman and Varian 2003], authors argued that there were around five exabytes of new information generated in the year 2002, and the amount of information produced in the world increases by 30% every year. This massive amount of information translates to a demand for large, cost-effective, and reliable storage systems, and the approach taken is to use thousands or even hundreds of thousands of inexpensive disks to preserve large volumes of data [Kubiatowicz et al. 2000; Joukov et al. 2007; Ghemawat et al. 2003].

The continuing increase of system size and the usage of inexpensive but less reliable components (for economical consideration) make component failures, such as disk failure, more common. When disk failure occurs, it has a huge impact on the reliability and data availability of large scale storage systems [Schroeder and Gibson 2007; Pinheiro et al. 2007]. However, an important requirement of building large storage systems is to make sure information is reliable and available even under the presence of component failures. System reliability and data availability can be achieved by maintaining a sufficient amount of redundancy in the storage system. For example, in recent years, several RAID-6 codes such as RDP [Corbett et al. 2004], EVENODD [Blaum et al. 1995], and X-code [Xu and Bruck 1999] were proposed to protect data against up to *two disk failures* and can maintain higher level reliability as compared to the conventional RAID-5 systems.

Data will be lost (or unavailable) when the number of failed disks is more than the number of parity disks. Therefore, to guarantee that the storage system operates at a high reliable level, a key approach is to repair and recover a failed component as quickly as possible [Baker et al. 2006]. In general, the recovery process requires multiple reads to surviving disks, and the total data that must be processed during recovery plays a crucial role in the overall recovery time [Corbett et al. 2004; Muntz and Lui 1990]. Moreover, as the storage capacity of modern disk is growing at a much faster rate than the disk I/O speed, the disk recovery process for modern disks takes much longer time. This implies the lengthening of time in the *window of vulnerability* (WOV) and it increases the probability of data loss [Xin et al. 2003]. Therefore, if one can reduce the total disk reads in the recovery process, the system reliability can be improved.

One important point we want to emphasize is that most of the modern storage systems have an online failure recovery mode [Holland 1994], in which the system still provides service to the users during the disk recovery period. However, multiple reads from disks for recovery consume I/O bandwidth of the system and influence the system service performance [Holland et al. 1993]. Therefore, if one can reduce the number of disk reads for disk recovery, this also implies that we can improve the service performance to users during the online failure recovery mode. Other benefits for reducing disk reads include relieving the communication load of the entire storage system and the possibility to conserve energy, which is becoming more important for data centers which deploy large storage systems.

Storage systems based on RAID-6 codes (e.g., RDP, EVENODD) usually contain two parity disks to protect against double disk failures. However, the frequency of single disk failure is much higher than double disk failures (this is especially true when the aggregated disk failure rate is much lower than the disk recovery rate) [Chen et al. 1994; Corbett et al. 2004]. Therefore, it is necessary to design efficient recovery schemes for a single failed disk in storage systems that based on RAID-6 codes. However, most commonly adopted disk recovery schemes only use single parity for single failure recovery [Blaum et al. 1995; Corbett et al. 2004; Xu and Bruck 1999], and there is no research which focuses on designing fast and efficient recovery schemes of single failure recovery for RAID-6 codes.

In RAID-6 systems, each data block is protected by at least two different parity blocks, and an erased block can be recovered from each of its parity blocks. We find that if the two parity disks in the system are both used to recover a single failed disk, there are many overlapping blocks which will be read twice during recovery. If an overlapping block is stored in memory after it is read from a disk at the first time, it can be read directly from memory at the second time which can reduce the amount of data to be read from disks for recovery.

Recent studies show that memory read is about 100 times faster than disk read. For instance, a state-of-the-art SATA disk driver has a linear read speed of approximately 60MB/sec, while DDR2-800 memory read speed is 6400MB/sec [Wikipedia 2010]. If some disk reads can be substituted by memory reads, one can speed up the disk recovery process and reduce the communication load of the storage system.

The aim of this paper is to explore how to design efficient recovery schemes for single disk failure in storage systems that use RAID-6 codes. We consider single disk failure recovery on two important RAID-6 codes, RDP code and EVENODD code. We show that one can exploit both parity disks to reduce the number of disk reads for recovery and propose hybrid recovery schemes RDOR-RDP [Xiang et al. 2010] and RDOR-EVENODD for RDP and EVENODD codes respectively. By using RDOR-RDP and RDOR-EVENODD, the numbers of disk reads for single disk failure recovery of RDP and EVENODD codes can be reduced by approximately 25% compared with their conventional schemes. The main contributions of this paper are:

— We propose a new hybrid recovery approach for single disk failure which can reduce the number of disk reads and therefore improves system recovery performance for RAID-6 codes[1].
— We derive the tight lower bounds of the number of disk reads needed for any single failure recovery of RDP and EVENODD codes respectively.
— We propose efficient recovery schemes RDOR-RDP and RDOR-EVENODD for RDP and EVENODD codes respectively which match their disk read lower bounds and also have the load-balancing property that all surviving disks will experience the same amount of workload during recovery.

This paper is organized as follows. We will briefly introduce the RAID-6 storage system, the RDP and EVENODD codes in Section 2. Section 3 presents the main idea of our hybrid approach using a simple example of RDP code. In Section 4 and Section 5, we will theoretically analyze the lower bounds of disk reads and propose optimal recovery schemes for RDP and EVENODD codes respectively. In Section 6 and Section 7, we will present and discuss the experimental results to show the advantage of our hybrid approach. Section 8 introduces related work on disk recovery. Conclusions are presented in Section 9.

[1]The hybrid recovery approach can only be applied to parity array codes for RAID-6 systems.

	Disk0	Disk1	Disk2	Disk3	Disk4	Disk5
Stripe 0	D_0	D_1	D_2	D_3	P_{0-3}	Q_{0-3}
Stripe 1	D_5	D_6	D_7	P_{4-7}	Q_{4-7}	D_4
Stripe 2	D_{10}	D_{11}	P_{8-11}	Q_{8-11}	D_8	D_9
Stripe 3	D_{15}	P_{12-15}	Q_{12-15}	D_{12}	D_{13}	D_{14}
Stripe 4	P_{16-19}	Q_{16-19}	D_{16}	D_{17}	D_{18}	D_{19}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Fig. 1. An Example of RAID-6 System Implementation of $k = 4$.

2. BACKGROUND

In this section, we first briefly introduce the specification and the current implementation of RAID-6 storage systems, and then present two important coding schemes, RDP code and EVENODD code, which are commonly used in RAID-6 systems.

2.1. RAID-6 Storage System Implementations

A RAID storage system consists of some disks of the same size arranged in an array. Some of the disks in a RAID system hold information data and the rests hold redundant data. RAID-6 is a specification of RAID systems which protect information data against two disk failures at the same time. A RAID-6 storage system consists of $k + 2$ disks, where k disks hold information data and the rest 2 disks hold coded data. The coded data are generated from the information data and are often termed as *parity* [Plank et al. 2009]. Each disk in the system is divided into *strips* of a fixed size, and a *parity strip* is calculated from the data strips. A *stripe* consists of k data strips and their corresponding 2 parity strips.

Figure 1 shows a particular RAID-6 system of $k = 4$. There are 6 disks in the system, Disk 0 is divided into strips D_0, D_5, $\cdots$, Disk 1 is divided into strips D_1, D_6, $\cdots$, etc. Stripe 0 is composed of data strips D_0 to D_3 and parity strips P_{0-3}, Q_{0-3}, where P_{0-3}, Q_{0-3} are generated from D_0 to D_3. To avoid hot spots in the system, the parity strips are rotated between stripes, so all the disks will get the same workload. See Figure 1 for illustration.

Although RAID-6 systems can tolerant double disk failures, double disk failures are comparatively more rare than single disk failure [Chen et al. 1994]. Moreover, once a disk failure occurs, the system will be in a degrade mode both on users' service performance and system reliability level. For example, suppose in Figure 1 Disk 0 fails, a user's read request on data strip D_0 will be considered as reading D_1, D_2, D_3, P_{0-3}, and XORing them to get D_0. Thus, it is important to design efficient recovery algorithm for single disk failure to minimize the degradation on system performance and reliability under failure.

2.2. Erasure Codes

To implement RAID-6 storage systems, erasure codes are used to generate the parity data. In information theory, an erasure code is a forward error correction (FEC) code which encodes a message of k symbols into a coded word of n $(n \geq k)$ symbols such that the original message can be recovered from a subset of the n symbols [Pless 1998]. Recently, a large number of erasure codes are proposed for RAID-6 systems. In the

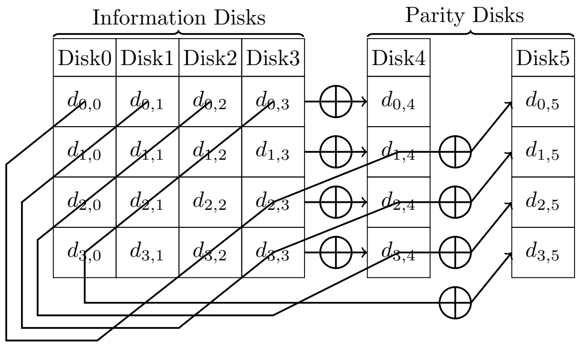

Information Disks
Parity Disks
Disk0
Disk1
Disk2
Disk3
Disk4
Disk5
$d_{0,0}$ $d_{0,1}$ $d_{0,2}$ $d_{0,3}$ $d_{0,4}$ $d_{0,5}$
$d_{1,0}$ $d_{1,1}$ $d_{1,2}$ $d_{1,3}$ $d_{1,4}$ $d_{1,5}$
$d_{2,0}$ $d_{2,1}$ $d_{2,2}$ $d_{2,3}$ $d_{2,4}$ $d_{2,5}$
$d_{3,0}$ $d_{3,1}$ $d_{3,2}$ $d_{3,3}$ $d_{3,4}$ $d_{3,5}$

Fig. 2. RDP coding in a stripe of $p = 5$.

following of this section, we introduce two widely used RAID-6 codes studied in this paper.

— **RDP code** is one of the most important RAID-6 codes which achieves optimality both in computation efficiency and I/O efficiency. It is also named as RAID-DP in NetApp's commercial products [Lueth 2004]. The RDP encoding takes a $(p-1) \times (p+1)$ two-dimensional array (or what we call a stripe), where p is a prime number greater than 2. The first $p-1$ columns in the array are information columns and the last two are parity columns. In actual implementations, the identities of information and parity columns are rotated between stripes [Plank 2008].
The two parity columns in the array, named as the *row parity column* and the *diagonal parity column* respectively, ensure all information is recoverable when there are no more than two disk failures. A row parity symbol[2], just like in RAID-4, is generated by XOR-summing all the information symbols in that row, and a diagonal parity symbol is by XOR-summing all symbols along the same diagonal. Figure 2 shows the construction mechanism of RDP code when $p = 5$, where $d_{i,j}$ is the i-th symbol in column j. The first four disks, Disk 0 to Disk 3 are information disks, while the last two disks, Disk 4 and Disk 5, are parity disks. Disk 4 contains all the row parity symbols, e.g., $d_{0,4}$ is the XOR-sum of data blocks $\{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}\}$, while Disk 5 contains the diagonal parity symbols, e.g., $d_{0,5}$ is the XOR-sum of data blocks $\{d_{0,0}, d_{3,2}, d_{2,3}, d_{1,4}\}$. Diagonal $p-1$ ($\{d_{i,j} | i + j \equiv p-1 \pmod p\}$) is called the *missing diagonal* as it does not have a corresponding diagonal parity. The missing diagonal consists of symbols $\{d_{0,4},\ d_{1,3},\ d_{2,2},\ d_{3,1}\}$. Refer to [Corbett et al. 2004] for details.

— **EVENODD code** is one of the most widely studied RAID-6 codes. The EVENODD encoding considers a $(p-1) \times (p+2)$ two-dimensional array (or a stripe) with a prime number p greater than 2. Column p and $p+1$ are two parity columns while the other p columns are information columns.
The two parity columns, column p and column $p+1$, can also be named as row parity column and diagonal parity column. A row parity symbol $d_{i,p}$ in column p is computed as the XOR-sum of all information symbols in row i. The XOR-sum of symbols along diagonal $p-1$, which computed as $h = \oplus_{j=1}^{p-1} d_{p-1-j,j}$, is called the EVENODD adjuster and used to generate each parity symbol in diagonal parity column $p+1$. A diagonal parity symbol $d_{j,p+1}$ in column $p+1$ is generated by XOR-summing all information symbols along diagonal j ($\{d_{i,r} | i + r \equiv j \pmod p\}$) and the adjuster h. Figure 3 shows a particular EVENODD code of $p = 5$. Refer to [Blaum et al. 1995] for details.

[2]We use the term of symbols to represent device blocks, and a symbol corresponds to a set of consecutive sectors of a disk.

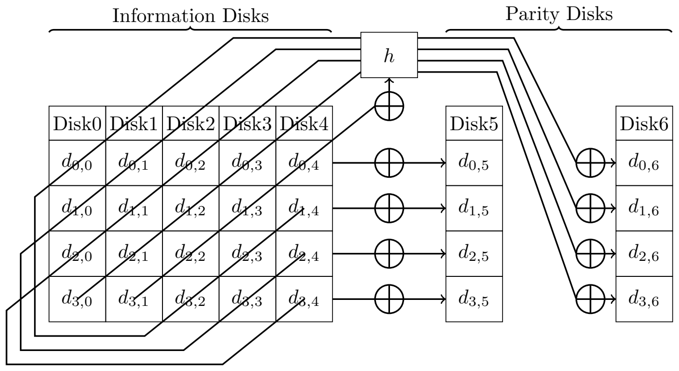

Information Disks
Parity Disks
h
Disk0
Disk1
Disk2
Disk3
Disk4
Disk5
Disk6
$d_{0,0}$ $d_{0,1}$ $d_{0,2}$ $d_{0,3}$ $d_{0,4}$ $d_{0,5}$ $d_{0,6}$
$d_{1,0}$ $d_{1,1}$ $d_{1,2}$ $d_{1,3}$ $d_{1,4}$ $d_{1,5}$ $d_{1,6}$
$d_{2,0}$ $d_{2,1}$ $d_{2,2}$ $d_{2,3}$ $d_{2,4}$ $d_{2,5}$ $d_{2,6}$
$d_{3,0}$ $d_{3,1}$ $d_{3,2}$ $d_{3,3}$ $d_{3,4}$ $d_{3,5}$ $d_{3,6}$

Fig. 3. EVENODD coding in a stripe of $p = 5$.

In RAID-6 systems, when a disk fails, to maintain the system reliability level, the data in the failed disk should be reconstructed by reading corresponding information and parity data from the surviving disks, and the reconstructed data should be stored in a spare disk as soon as possible. To deal with disk failures, each erasure code has its specific recovery algorithm. In the next section, we introduce the conventional recovery schemes for a failed disk in RDP and EVENODD storage systems.

2.3. Conventional Recovery Schemes for RDP and EVENODD Storage Systems

Let us discuss the "*conventional*" approach of recovering a failed disk in a storage system which uses RDP or EVENODD codes.

The conventional recovery schemes for RDP and EVENODD codes are similar to each other. For RDP and EVENODD codes, in the case of a single disk failed, if the failed disk is an information disk, the conventional approach is to use row parity and information symbols in that row to recover each erasure symbol (an erasure symbol is an unreadable symbol in the failed disk). If the failed disk is a parity disk, recovering the disk is equivalent to recomputing parity.

Consider the example of RDP code in Figure 2, if Disk 0 fails, one can read row parity symbol $\{d_{0,4}\}$ and information symbols $\{d_{0,1}, d_{0,2}, d_{0,3}\}$ to recover $d_{0,0}$. If the failed disk is the row parity disk (Disk 4) one can XOR-sum $\{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}\}$ to reconstruct $d_{0,4}$, and if the diagonal parity disk (Disk 5) fails, symbols $\{d_{3,0}, d_{2,1}, d_{1,2}, d_{0,3}\}$ are used to reconstruct $d_{3,5}$.

For EVENODD code in Figure 3, if the failed disk is Disk 0, symbols $\{d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}, d_{0,5}\}$ are used to recover $d_{0,0}$. If Disk 5 fails, one can XOR-sum $\{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}\}$ to reconstruct $d_{0,5}$, and if Disk 6 fails, according to the encoding algorithm of EVENODD code, we should reconstruct the adjuster h ($h = d_{3,1} \oplus d_{2,2} \oplus d_{1,3} \oplus d_{0,4}$) and then use symbols $\{d_{3,0}, d_{2,1}, d_{1,2}, d_{0,3}\}$ and h to reconstruct erasure symbol $d_{3,6}$ in Disk 6.

Figure 4 shows the conventional recovery scheme of RDP code with $p = 7$. Assume that Disk 0 (column 0) fails. The six information symbols $d_{i,0}$ $(0 \leq i \leq 5)$ are to be recovered. With the conventional recovery scheme, $d_{i,0}$ $(0 \leq i \leq 5)$ are all recovered from row parity (or Disk 6). During recovery, we need to read all the symbols in Disk 6 and the rest information disks to reconstruct the erasure symbols in Disk 0, and then write the reconstructed data to a spare disk. Therefore, 36 symbols need to be read from disks for the recovery. As illustrated in Figure 4, the erasure symbols are labeled with "$\times$", and the symbols used for recovery are labeled with "$\bigcirc$".

We like to point out that the conventional schemes for single disk failure recovery of RDP and EVENODD codes only use *one parity column*. However, because all data

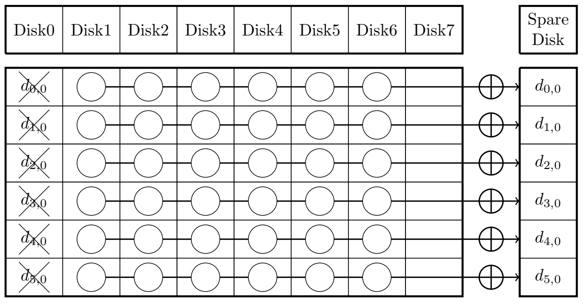


Fig. 4. Conventional scheme to recover Disk 0 of RDP code.

blocks are protected by two different parity groups, there are ways for us to take advantage of this feature to speed up the recovery process. In the following, we will present a novel approach which uses both parities. The interesting property of the new hybrid approach is that it reduces the number of disk reads for the recovery.

3. NEW RECOVERY APPROACH FOR SINGLE FAILURE

To illutrate our recovery process, let us first use the example of RDP code in Figure 4 to demonstrate the idea of the hybrid approach for single disk failure recovery, and then we will formally present the recovery process, both for the RDP and EVENODD codes.

Minimizing Disk Reads: Because an information symbol can be recovered from either row parity or diagonal parity, the conventional recovery scheme is only one of many possible schemes. Suppose that in Figure 4, if symbols $d_{0,0}$, $d_{1,0}$, $d_{2,0}$ are recovered from row parity and $d_{3,0}$, $d_{4,0}$, $d_{5,0}$ are recovered from diagonal parity, there are 9 overlapping symbols which will be read *twice* for the recovery process. Figure 5(a) illustrates this case, where the symbols labeled with "◯" are read for the recovery by row parity and the symbols labeled with "□" are read for the recovery by diagonal parity. If an overlapping symbol is stored in memory after it is read from a disk at the first time, it can be read from memory for the consecutive recovery of other information symbols. So a total of $36 - 9 = 27$ symbols will be read from disk for the recovery, while 9 symbols are read from memory. Because the speed of memory read is much faster than that of disk read, reading symbols from memory instead of from disk will speed up recovery process. On the other hand, by using memory read instead of disk read, the communication load of the storage system will also be reduced. The main idea of using previously read symbols to recover data so as to reduce I/O operations is appealing. Theoretically, we would like to answer the following questions:

— What is the lower bound of disk reads for single disk failure recovery?
— How to design a recovery scheme which matches this lower bound?

Balancing Disk Reads: During recovery, one would expect the recovery read requests are uniformly distributed among all surviving disks to speed up the recovery process. For conventional recovery scheme, this can be achieved by parity rotation between stripes. However, by using double parities for the recovery, the numbers of symbols read from each surviving disk are determined by the recovery scheme we choose and can not be balanced simply by rotating parities between stripes. For example, simply rotating parities between stripes, if the recovery scheme in Figure 5(a) is carried out in each stripe, nearly all the data from Disk 1 needs to be read for the recovery of the failed Disk 0, while only half from Disk 4.

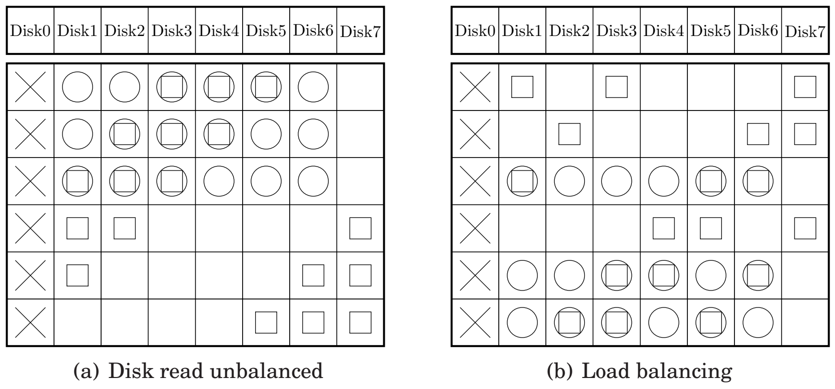


Fig. 5. (a) Reducing the number of disk reads for recovering Disk 0 of RDP code. (b) Load balancing in recovering Disk 0.

Consider another hybrid recovery scheme in Figure 4, where $d_{2,0}$, $d_{4,0}$, $d_{5,0}$ are recovered from row parity and $d_{0,0}$, $d_{1,0}$, $d_{3,0}$ are recovered from diagonal parity. Figure 5(b) depicts the symbols being read for the recovery with this scheme. In Figure 5(b), 27 symbols are read from disks for the recovery, the same as that in Figure 5(a). However, in Figure 5(b), the numbers of symbols we need to read from Disk 1 to Disk 6 for recovery are all equal to 4. In other words, the additional workload to recover data in Disk 0 is *perfectly balanced* among all surviving disks except for the diagonal parity disk. One interesting question we would like to address is:

— Whether there exists a *balanced* recovery scheme with which the numbers of read operations from each disk are the same?

Remark 3.1. Using hybrid approach to recover a failed disk can reduce the amount of data needs to be read from disk for recovery. However, the hybrid approach changes the access pattern of disk read from purely sequential accesses to more random accesses. For example, in Figure 4, if there is no foreground workload, the conventional approach will read six sequential symbols in each surviving disk (except for Disk 7) for recovery. However, the read requests of hybrid approach are non-sequential. See Figure 5 for an illustration. In Section 6, we will analyze the impact of non-sequential reads on disk recovery. Besides, in Section 7, we will show that the hybrid approach is more applicable in some practical cases.

The main idea of our hybrid approach is using double parities for single failure recovery to reduce the number of disk reads for recovery. For RAID-6 codes, each symbol is protected by at least two parity groups, so the hybrid approach can be generalized to other RAID-6 codes such as EVENODD and X-code. In this paper, we focus on two widely accepted RAID-6 codes, RDP code and EVENODD code. We will analyze the minimum disk reads and balanced disk reads for the recovery of RDP and EVENODD codes. We will also design recovery schemes of single disk failure for them which match the minimum disk reads and balanced disk reads simultaneously.

4. THE RECOVERY OF SINGLE DISK FAILURE FOR RDP CODE

4.1. Lower Bound of Disk Reads for Single Failure Recovery

In this section, we will derive the lower bound of the number of symbols to be read from the disks for any single column erasure recovery under RDP code. Because an erasure symbol can be recovered either from row parity or diagonal parity, we will show that there are many choices for the recovery of an erasure column. Among the different recovery choices, we want to find one which minimizes the number of disk reads. To describe the different recovery choices, we define the following notations.

Definition 4.1. Define $R_i = \{d_{i,r} | 0 \leq r \leq p-1\}$ as the i-th row parity set, $0 \leq i \leq p-2$, and $D_j = \{d_{i,r} | (i+r) \bmod p = j, 0 \leq i \leq p-2, 0 \leq r \leq p\}$ as the j-th diagonal parity set, $0 \leq j \leq p-2$.

Based on Definition 4.1, we can see that $|R_i| = p$ for $0 \leq i \leq p-2$ and $|D_j| = p$ for $0 \leq j \leq p-2$ (*Note that* $d_{j,0} \in D_j$ *and* $d_{j,p} \in D_j$).

R_i is the set of symbols in row i except for the diagonal parity symbol $d_{i,p}$ in column p. According to RDP code, $d_{i,p-1} \in R_i$ is the XOR-sum of all the other symbols in R_i, i.e. $d_{i,p-1} = \bigoplus_{j=0}^{p-2} d_{i,j}$. So given a symbol $d \in R_i$, d can be recovered by XOR-summing all the other symbols in $R_i - \{d\}$. While D_j is the set of symbols in the diagonal which contains $d_{j,p}$, the j-th symbol in column p. With the same reason, given a symbol $d \in D_j$, d can be recovered by XOR-summing all the other symbols in $D_j - \{d\}$. So we have the following Lemma 4.2.

LEMMA 4.2. *Given a symbol d in an RDP disk array,*

(*1*) *if $d \in R_i$, d can be recovered by XOR-summing all symbols in $R_i - \{d\}$; (For the ease of presentation, we say that d can be recovered from R_i throughout this paper.)*
(*2*) *if $d \in D_j$, d can be recovered from D_j;*
(*3*) *d can only be recovered from 1) and/or 2)*

Remark 4.3. Erasure symbols can only be recovered from their parity sets, but the surviving symbols in the corresponding parity set for the recovery can be either directly read from disks or further generated from other parity sets. In this paper, we only consider that all the surviving symbols are read directly from disks. In the appendix, we will show that the lower bounds of disk reads of RDP and EVENODD codes in the following Theorem 4.7 and Theorem 5.4 can not be further reduced by generating surviving symbols from other parity sets.

Consider the example in Figure 2. We have $R_0 = \{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}\}$, $D_0 = \{d_{0,0}, d_{3,2}, d_{2,3}, d_{1,4}, d_{0,5}\}$, $d_{0,0} \in R_0$ and $d_{0,0} \in D_0$. $d_{0,0}$ can be recovered from R_0 or from D_0. $d_{0,4} \in R_0$, but $d_{0,4} \notin D_j$ for $0 \leq j \leq p-2$. So $d_{0,4}$ can only be recovered from R_0.

Assume that Disk k in the disk array fails. We need to recover all the erasure symbols $d_{i,k}$ $(0 \leq i \leq p-2)$ in column k $(0 \leq k \leq p)$. Based on RDP code, we have the following Lemma 4.4.

LEMMA 4.4. *Given an erasure column k, $0 \leq k \leq p$,*

(*1*) *If $0 \leq k \leq p-1$, which means that column k is not the diagonal parity column.*
  (*a*) *If $i \neq p-1-k$, $d_{i,k} \in R_i$ and $d_{i,k} \in D_{<i+k>_p}$*[3]*. Symbol $d_{i,k}$ can be recovered from either R_i or $D_{<i+k>_p}$.*
  (*b*) *If $i = p-1-k$, only $d_{i,k} \in R_{p-1-k}$. Symbol $d_{p-1-k,k}$ is in the missing diagonal and can only be recovered from R_{p-1-k}.*
(*2*) *If $k = p$, which means that column k is the diagonal parity column, then $d_{i,p} \in D_i$. Symbol $d_{i,p}$ can only be recovered from D_i.*

Remark 4.5. Lemma 4.4 shows the possible choices for the recovery of an erasure symbol. Case (a) indicates that an erasure symbol, which is not in the diagonal parity

[3]For the ease of presentation, we denote $r \bmod p$ as $< r >_p$ in this paper.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
0:0	0:1	0:2	0:3	0:null	null:0
1:1	1:2	1:3	1:null	1:0	null:1
2:2	2:3	2:null	2:0	2:1	null:2
3:3	3:null	3:0	3:1	3:2	null:3

Fig. 6. RDP coding with R_i, D_j representation.

column and not in missing diagonal, can be recovered from a row parity set and from a diagonal parity set. Case (b) indicates that an erasure symbol, which is not in diagonal parity column but in missing diagonal, can be recovered only from a row parity set. Case (2) indicates that all symbols in the diagonal parity column can be recovered only from diagonal parity sets.

Figure 6 shows the possible recovery choices of each erasure symbol with $p = 5$. Instead of using $d_{i,j}$ to denote the data at the i^{th} row and the j^{th} column, we use a pair of numbers to identify the parity sets the corresponding symbol belongs to. The first number in the pair is the subscript of the row parity set, and the second number is the subscript of the diagonal parity set. The symbols in column 5 only belong to their diagonal parity sets, and the symbols along the missing diagonal only belong to their row parity sets. For example, (2, 3) at row 2, column 1 means that $d_{2,1}$ can be recovered from R_2 or D_3; (1, null) at row 1, column 3 means that $d_{1,3}$ can only be recovered from R_1.

We can choose a combination of parity sets (*Note: We call it recovery combination in the following.*) to recover the $p-1$ erasure symbols in a column. Consider Disk 1 in Figure 6, we can choose a combination of (R_0, D_2, R_2, R_3) to recover Disk 1, which means that R_0 is used to recover $d_{0,1}$, D_2 is used to recover $d_{1,1}$, and so on. Since there are two choices for the recovery of $d_{0,1}$ (from R_0 or D_1), $d_{1,1}$ and $d_{2,1}$ respectively, and only one choice for the recovery of $d_{3,1}$ (from R_3 only), there are $2 \times 2 \times 2 \times 1 = 8$ recovery combinations to reconstruct symbols in Disk 1. The conventional recovery scheme, (R_0, R_1, R_2, R_3), is just one of possible recovery combinations. Since Disk p (or column p) is the diagonal parity column, which can only be recovered by diagonal parity sets, there is only one recovery combination for column p. So we only consider the recovery of Disk k, with $k \neq p$ in this paper.

Our objective is to find the *read optimal* one from all possible recovery combinations which has the minimum number of disk reads. This is equivalent to find a recovery combination with most overlapping symbols. We have the following Lemma 4.6 indicating the property of overlapping symbols between two parity sets.

LEMMA 4.6.

(1) *There is just one overlapping symbol between each pair of R_i and D_j for $0 \le i, j \le p-2$, and the overlapping symbol is $R_i \cap D_j = \{d_{i,<j-i>_p}\}$.*
(2) *There is no overlapping symbol between each pair of R_i and R_j, or D_i and D_j, i.e. $R_i \cap R_j = \emptyset$ and $D_i \cap D_j = \emptyset$ for $0 \le i, j \le p-2$, $i \neq j$.*

It is interesting to point out that the conventional recovery scheme for single disk failure only uses row parity sets for the recovery. Lemma 4.6 shows that there is no overlapping symbol during the recovery process with the conventional recovery scheme. Therefore, one cannot reduce the read operations using the conventional recovery scheme. Also according to Lemma 4.6, if we use both the row and the diagonal parity sets to recover Disk $k\,(k \neq p)$, we can reduce disk read operations by putting overlapping symbols in memory. The following Theorem 4.7 gives a lower bound of disk reads for any single erasure column recovery.

THEOREM 4.7.

(*1*) *For any erasure column k ($k \neq p$), the minimum number of disk reads to recover column k of RDP code is* $3(p-1)^2/4$.

(*2*) *Any recovery combination which consists of* $(p-1)/2$ *row parity sets and* $(p-1)/2$ *diagonal parity sets will match the minimum number of disk reads. There are* $\binom{p-1}{(p-1)/2}$ *possible recovery combinations which match the minimum number of disk reads.*

PROOF. (1) Let the erasure column be column $k\,(0 \leq k \leq p-1)$. We need to recover all the $p-1$ symbols $d_{i,k}(0 \leq i \leq p-2)$ in column k. Erasure symbol $d_{i,k}$ can only be recovered by the parity sets to which it belongs. From Lemma 4.6, we know that there is one overlapping symbol between each pair of R_i and D_j. If t erasure symbols (*except for* $d_{p-1-k,k}$) are recovered from diagonal parity sets and the remaining $p-1-t$ symbols are recovered from row parity sets, the number of overlapping symbols is

$$t(p-1-t) = -(t-(p-1)/2)^2 + (p-1)^2/4.$$

When $t = (p-1)/2$, the number of overlapping symbols, $t(p-1-t)$, is maximized, which equals to $(p-1)^2/4$. During the recovery process, if an overlapping symbol is read from a disk to recover an erasure symbol, it will be stored in memory for the recovery of another erasure symbol. While recovering the later erasure symbol, the overlapping symbol can be read from memory instead of performing another disk read. So the maximum of $(p-1)^2/4$ symbols can be reduced from disk read. The minimum number of disk reads for the recovery is

$$(p-1)^2 - (p-1)^2/4 = 3(p-1)^2/4,$$

and a read optimal recovery combination which matches the lower bound should consist of $(p-1)/2$ row parity sets and $(p-1)/2$ diagonal parity sets.

(2) The correctness of condition (2) follows directly from the proof of condition (1). Therefore, Theorem 4.7 concludes. □

Based on Theorem 4.7, a read optimal recovery scheme can be stated as: for a single failed Disk k, choose any $(p-1)/2$ symbols (*Note:* $d_{p-1-k,k}$ *must be included*), reconstruct them from row parity sets and reconstruct the remaining $(p-1)/2$ symbols from diagonal parity sets.

However, the number of disk reads on individual disk of some recovery combinations are the same (or almost the same), while that of others varies greatly. In the next subsection, we will study how to balance the disk reads.

4.2. Balancing Disk Reads

Let the erasure column be column $k\,(0 \leq k \leq p-1)$. The minimum number of symbols to be read from disks for the recovery is $3(p-1)^2/4$. Since any read optimal recovery combination contains $(p-1)/2$ diagonal parity sets, $(p-1)/2$ symbols will always be read from Disk p (*diagonal parity disk*). Then the average number of symbols to be read from the other surviving disks (*except for Disk k and Disk p*) is

$$\frac{3(p-1)^2/4 - (p-1)/2}{p-1} = (3p-5)/4.$$

As $(p-1)/2$ symbols will always be read from Disk p, now we only consider how to provide a read balanced recovery scheme on the remaining disks (*except for Disk k and Disk p*).

Given a prime number $p > 2$, we have to consider two cases: either $p \equiv 3 \pmod 4$ or $p \equiv 1 \pmod 4$.

Case 1. If $p \equiv 3 \pmod 4$, then $3p-5$ is divisible by 4. A balanced and read optimal recovery combination should read $(3p-5)/4$ symbols from each of the disks (*except for Disk k and Disk p*).
Case 2. If $p \equiv 1 \pmod 4$, then $3p-5$ isn't divisible by 4. A balanced and read optimal recovery combination should read $\lceil (3p-5)/4 \rceil$ symbols from some disks and $\lfloor (3p-5)/4 \rfloor$ symbols from the rest.

For example, if $p = 7$ and $k = 0$, as $7 \equiv 3 \pmod 4$, a balanced and read optimal recovery combination of Disk 0 should read $(3p-5)/4 = 4$ symbols from Disk 1 to Disk 6. See Figure 5(b) for an illustration.

In the following, we will only present the analysis for the case of $p \equiv 3 \pmod 4$. The analysis of the case $p \equiv 1 \pmod 4$ is similar. So we will only list the conclusions of the case $p \equiv 1 \pmod 4$ at the end of this section and omit their analysis for brevity.

To simplify the analysis of balanced disk read, we introduce a recovery sequence $x_0, x_1, \ldots, x_{p-2}$, where $x_i = 0$ means that $d_{i,k}$ is recovered from its row parity set, and $x_i = 1$ means that $d_{i,k}$ is recovered from its diagonal parity set. So a recovery combination can be exactly represented by a recovery sequence.

Our analysis of balanced disk read is based on the finite field F_p. To facilitate the analysis, we add an additional row $p-1$ with all symbols $d_{p-1,j} = 0\,(0 \le j \le p)$ to the encoded array throughout this section. So the row number set $\{0, 1, \ldots, p-1\}$ consists of all the elements of F_p. With the additional row, the disk array is now a $p \times (p+1)$ array. Since all symbols in row $p-1$ are 0, $d_{p-1,k}$ in row $p-1$ can be regarded as being recovered from its row parity set. Because $d_{p-1,k}$ is recovered from its row parity set, $x_{p-1} = 0$. Moreover, $d_{p-1-k,k}$ is in missing diagonal and only belongs to its row parity set, which can only be recovered by its row parity set. So, $x_{p-1-k} = 0$. With the additional row, a recovery sequence is $x_0, x_1, \ldots, x_{p-2}, x_{p-1}$ with $x_{p-1} = 0$ and $x_{p-1-k} = 0$.

A balanced recovery scheme will read the same (or almost the same) number of symbols from each of the surviving disks. The number of symbols need to be read from each surviving disk is decided by the recovery sequence we choose. Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$, we consider that how many symbols need to be read from each surviving disk under this recovery sequence. We still take $p = 7$ and $k = 0$ to illustrate, suppose the recovery sequence being $\{x_i\}_{0 \le i \le 6} = \{1101000\}$, its corresponding recovery scheme is shown in Figure 5(b).

According to the recovery sequence, as $x_2 = x_4 = x_5 = x_6 = 0$, symbols $d_{2,0}$, $d_{4,0}$, $d_{5,0}$, $d_{6,0}$ are recovered from R_2, R_4, R_5, R_6 respectively, then $d_{2,j} \in R_2$, $d_{4,j} \in R_4$, $d_{5,j} \in R_5$, and $d_{6,j} \in R_6$ in Disk j $(1 \le j \le 6)$ will be read for recovery. As $x_0 = x_1 = x_3 = 1$, symbols $d_{0,0}$, $d_{1,0}$, $d_{3,0}$ are recovered from D_0, D_1, D_3 respectively, then $d_{p-j,j} \in D_0$, $d_{<1-j>_p,j} \in D_1$, and $d_{<3-j>_p,j} \in D_3$ in Disk j will be read for recovery.

Now we can find that if a symbol $d_{i,j}$ in Disk j needs to be read for recovery, either R_i or $D_{<i+j>_p}$ is chosen for recovery, which equals to $x_i = 0$ or $x_{<i+j>_p} = 1$.

Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$, the following Lemma 4.8 indicates that how many symbols will not be read from a surviving disk with the recovery combination corresponding to $\{x_i\}_{0 \le i \le p-1}$. It helps to understand Theorem 4.9.

LEMMA 4.8. *Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$. If an erasure column $k(0 \le k \le p-1)$ is recovered by the recovery combination corresponding to $\{x_i\}_{0 \le i \le p-1}$, the number* *of symbols which are not read from Disk j is:*

$$\sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+j-k>_p}.$$

PROOF. We define $c_{i,j}$ to identify whether $d_{i,j}$ is read for the recovery. If $d_{i,j}$ is read for the recovery, $c_{i,j} = 0$; otherwise $c_{i,j} = 1\,(0 \leq i, j \leq p-1)$.

If $x_i = 0$, symbol $d_{i,k}$ is recovered by row parity set R_i, $d_{i,j}$ is read to recover $d_{i,k}$ because $d_{i,j} \in R_i$. If $x_{<i+j-k>_p} = 1$, symbol $d_{<i+j-k>_p,k}$ is recovered by diagonal parity set $D_{<i+j>_p}$, $d_{i,j}$ is read to recover $d_{<i+j-k>_p,k}$ because $d_{i,j} \in D_{<i+j>_p}$. Given $d_{i,j}$, either $d_{i,j} \in R_i$ or $d_{i,j} \in D_{<i+j>_p}$. So only when $x_i = 0$ or $x_{<i+j-k>_p} = 1$, $d_{i,j}$ is read for recovery. Therefore, only when $x_i = 0$ or $x_{<i+j-k>_p} = 1$, $c_{i,j} = 0$. So $c_{i,j} = x_i(1 - x_{<i+j-k>_p})$.

The number of symbols which do not need to be read from Disk $j\,(0 \leq j \leq p-1, j \neq k)$ for the recovery of Disk k is

$$\sum_{i=0}^{p-1} c_{i,j} = \sum_{i=0}^{p-1} x_i(1 - x_{<i+j-k>_p}) = \sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+j-k>_p}.$$

Therefore, Lemma 4.8 concludes. □

The following Theorem 4.9 provides a sufficient condition for a recovery sequence to be both read optimal and balanced.

THEOREM 4.9. *If a recovery sequence $\{x_i\}_{0 \leq i \leq p-1}$ of Disk $k\,(0 \leq k \leq p-1)$ satisfies that:*

(*1*) $x_0 + x_1 + \cdots + x_{p-2} + x_{p-1} = (p-1)/2$*;*
(*2*) $x_{p-1-k} = x_{p-1} = 0$*;*
(*3*) $\forall t,\ 1 \leq t \leq p-1,\ \sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-3)/4$.

$\{x_i\}_{0 \leq i \leq p-1}$ *is a read optimal and balanced recovery sequence for the recovery of Disk $k\,(0 \leq k \leq p-1)$.*

PROOF. Condition (1) just means that $(p-1)/2$ symbols are recovered from row parity sets and $(p-1)/2$ symbols are recovered from diagonal parity sets. While condition (2) means that $d_{p-1-k,k}$ (which is in missing diagonal) and $d_{p-1,k}$ (which is an additional symbol) are recovered from their row parity sets. From Theorem 4.7, that condition (1) and condition (2) hold is equivalent to that $\{x_i\}_{0 \leq i \leq p-1}$ is a read optimal sequence.

In the following of the proof, we concentrate on condition (3).

As any read optimal recovery will read $(3p-5)/4$ symbols, on average, from each of the surviving disks (except for Disk p), if the disk read of recovery is balanced,

$$(p-1) - (3p-5)/4 = (p+1)/4$$

symbols will not be read from each of the disks. From Lemma 4.8, The number of symbols which don't need to be read from Disk $j\,(0 \leq j \leq p-1, j \neq k)$ is

$$\sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+j-k>_p}. \tag{1}$$

Let $t =< j-k >_p$, then $< i+j-k >_p =< i+t >_p$. Since $0 \leq j \leq p-1$, $j \neq k$ and $0 \leq k \leq p-1$, $1 \leq t \leq p-1$.

Therefore,

$$\sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} = \sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+t>_p}. \tag{2}$$

Because the recovery is read optimal, from Theorem 4.7, $(p-1)/2$ symbols in column k are recovered from their diagonal parity sets. So $\sum_{i=0}^{p-1} x_i = (p-1)/2$. According to condition (3), $\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-3)/4$.

Therefore,

$$\sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-1)/2 - (p-3)/4 = (p+1)/4. \tag{3}$$

Equation (3) means that if condition (3) holds, $(3p-5)/4$ symbols are read from each of the disks (*except for Disk k and Disk p*). So $\{x_i\}_{0 \le i \le p-1}$ is balanced.

Therefore, Theorem 4.9 holds. □

Conditions (1) and (2) of Theorem 4.9 ensure that a recovery sequence is read optimal, and condition (3) ensures a recovery sequence is read balanced. Now to find a read optimal and balanced recovery sequence is equivalent to find a recovery sequence satisfies the three conditions of Theorem 4.9. And the most difficult part is how to find a recovery sequence which satisfies Condition (3).

In the following of this subsection, we will use difference set [van Lint et al. 1993] to present a read optimal and balanced recovery scheme. Firstly, we introduce some definitions and properties of multiset and difference set.

Definition 4.10. A multiset $\mathcal{M}$ is a generalization of a set, in which there maybe multiple instances of an element. $\mathcal{M}$ is denoted as

$$\mathcal{M} = \{k_1 \times a_1, k_2 \times a_2, \ldots, k_n \times a_n\},$$

where $a_1, a_2, \ldots, a_n$ are all the distinct elements in $\mathcal{M}$, and there are k_i instances of a_i in $\mathcal{M}$, k_i is called the multiplicity of $a_i (1 \le i \le n)$.

Definition 4.11. Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$. Define $A_x = \{i | x_i = 1, 0 \le i \le p-1\}$ and the multiset $\mathcal{M}_{A_x} = \{a_1 - a_2 | a_1, a_2 \in A_x, a_1 \ne a_2\}$.

For example, considering the example in Figure 5(b), the recovery combination is (D_0, D_1, R_2, D_3, R_4, R_5), and its corresponding recovery sequence is $\{x_i\}_{0 \le i \le 6} = \{1101000\}$. According to Definition 4.11, $A_x = \{0, 1, 3\}$ and $\mathcal{M}_{A_x} = \{(0-1) \mod 7, (0-3) \mod 7, (1-0) \mod 7, (1-3) \mod 7, (3-0) \mod 7, (3-1) \mod 7\} = \{6, 4, 1, 5, 3, 2\}$.

LEMMA 4.12. *Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$. Then*

$$\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-3)/4 \quad \textit{for } 1 \le t \le p-1$$

is equivalent to that $A_x = \{i | x_i = 1, 0 \le i \le p-1\}$ satisfies $\mathcal{M}_{A_x} = \{[(p-3)/4] \times 1, [(p-3)/4] \times 2, \ldots, [(p-3)/4] \times (p-1)\}$, i.e. for any t, $1 \le t \le p-1$, the multiplicity of t in $\mathcal{M}_{A_x}$ is $(p-3)/4$.

PROOF. Given t, $1 \le t \le p-1$. If $\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-3)/4$, there are $i_{t_1}, i_{t_2}, \ldots, i_{t_{(p-3)/4}}$ with $0 \le i_{t_u}, i_{t_v} \le p-1$ and $i_{t_u} \ne i_{t_v}$ for $t_u \ne t_v$, such that

$$x_{i_{t_u}} = x_{<i_{t_u}+t>_p} = 1$$

for $1 \leq u \leq (p-3)/4$. According to Definition 4.11, this is equivalent to

$$x_{i_{t_u}}, x_{<i_{t_u}+t>_p} \in A_x$$

for $1 \leq u \leq (p-3)/4$, and further equivalent to that for $1 \leq u \leq (p-3)/4$,

$$(i_{t_u} + t) - i_{t_u} = t \in \mathcal{M}_{A_x}.$$

So for any t, $1 \leq t \leq p-1$, the multiplicity of t in $\mathcal{M}_{A_x}$ is $(p-3)/4$. Lemma 4.12 concludes. □

Consider the example in Figure 5(b), (D_0, D_1, R_2, D_3, R_4, R_5) is a read optimal and balanced recovery combination of Disk 0. Therefore, the corresponding recovery sequence $\{x_i\}_{0\leq i\leq 6}$ = $\{1101000\}$. It is easy to verify that for $1 \leq t \leq 6$, $\sum_{i=0}^{6} x_i x_{<i+t>_7} = (p-3)/4 = 1$. For example, when $t = 1$, $\sum_{i=0}^{6} x_i x_{<i+1>_7} = x_0x_1 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_0 = 1$. According to Definition 4.11, $A_x = \{0, 1, 3\}$ and $\mathcal{M}_{A_x} = \{a_1 - a_2 | a_1, a_2 \in A_x, a_1 \neq a_2\} = \{1, 2, 3, 4, 5, 6\}$, which also satisfies that $\forall t$ $(1 \leq t \leq 6)$, t has a multiplicity of $(p-3)/4 = 1$ in the multiset $\mathcal{M}_{A_x}$.

According to Lemma 4.12, to find a recovery sequence satisfying Condition (3) of Theorem 4.9 is equivalent to finding a set A_x which satisfies that the multiplicity of t in $\mathcal{M}_{A_x}$ is $(p-3)/4$ for any t $(1 \leq t \leq p-1)$.

Definition 4.13. Let G be an additive group of order v and S be a subset of G with k elements. S is called a $(v,\ k,\ \lambda)$-difference set if each nonzero element in G has a multiplicity of λ in the multiset $\mathcal{M}_S = \{s_1 - s_2 | s_1,\ s_2 \in S,\ s_1 \neq s_2\}$.

Definition 4.14. Let p be a prime number. $S_p = \{i^2 \bmod p | 1 \leq i \leq p-1\}$ is called the nonzero square set of F_p. $S'_p = F_p \setminus (S_p \cup \{0\})$ is called the non-square set of F_p.

THEOREM 4.15. *[van Lint et al. 1993] Let p be a prime number with $p \equiv 3 \pmod 4$, S_p and S'_p both are $\left(p,\ \frac{p-1}{2},\ \frac{p-3}{4}\right)$-difference sets in the additive group of F_p.*

Since $(p-i)^2 \equiv (p^2 - 2pi + i^2) \equiv i^2 \pmod p$, the nonzero square set $S_p = \{i^2 \bmod p | 1 \leq i \leq p-1\} = \{i^2 \bmod p | 1 \leq i \leq (p-1)/2\}$. When $p = 7$, $S_p = \{1^2 \bmod 7, 2^2 \bmod 7, 3^2 \bmod 7\} = \{1, 4, 2\}$. From Theorem 4.15, we can find that the set S_p satisfies that the multiplicity of t in $\mathcal{M}_{S_p}$ is $(p-3)/4$ for any t $(1 \leq t \leq p-1)$.

The following theorem gives a read optimal and balanced recovery sequence $\{x_i\}_{0\leq i\leq p-1}$ for any failed Disk k $(k \neq p)$, when $p \equiv 3 \pmod 4$.

THEOREM 4.16. *Given a prime number p with $p \equiv 3 \pmod 4$. For the nonzero square set S_p and the non-square set S'_p of F_p,*

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0\leq i\leq p-1}$ for the recovery of Disk k $(k \neq p)$, where $S'_p - (k+1) = \{s - (k+1) | s \in S'_p\}$ and $S_p - (k+1) = \{s - (k+1) | s \in S_p\}$.

PROOF. We only prove the case of $k \in S_p$. The proof of the case of $k \notin S_p$ is similar and therefore omitted.

According to $A = S'_p - (k+1)$, define a recovery sequence $\{x_i\}_{0\leq i\leq p-1}$ as that if $i \in A$, $x_i = 1$, otherwise $x_i = 0$. In the following, we will prove that recovery sequence $\{x_i\}_{0\leq i\leq p-1}$ is read optimal and balanced. According to Theorem 4.9, we need to prove that $\{x_i\}_{0\leq i\leq p-1}$ satisfies the three conditions of Theorem 4.9.

(1) Because $|A| = |S'_p| = (p-1)/2$, $x_0 + x_1 + \cdots + x_{p-2} + x_{p-1} = (p-1)/2$. So $\{x_i\}_{0\leq i\leq p-1}$ satisfies condition (1) of Theorem 4.9.

(2) Given $k \in S_p$ and $A = S'_p - (k+1)$,

$$k \in S_p \Rightarrow k \notin S'_p \Rightarrow k - (k+1) \notin S'_p - (k+1) \Rightarrow p-1 \notin A \Rightarrow x_{p-1} = 0$$
$$p \notin S'_p \Rightarrow p - (k+1) \notin S'_p - (k+1) \Rightarrow p - (k+1) \notin A \Rightarrow x_{p-1-k} = 0$$

So $\{x_i\}_{0\le i\le p-1}$ satisfies condition (2) of Theorem 4.9.

(3) According to Theorem 4.15, S'_p is a $\left(p,\ \frac{p-1}{2},\ \frac{p-3}{4}\right)$-difference set in the additive group of F_p. So $\forall t\ (1 \le t \le p-1)$, t has a multiplicity of $(p-3)/4$ in the multiset $\mathcal{M}_{S'_p} = \{s_1 - s_2 | s_1, s_2 \in S'_p,\ s_1 \ne s_2\}$.

Because $A = S'_p-(k+1)$, multiset $\mathcal{M}_A = \{a_1-a_2|a_1, a_2 \in A,\ a_1 \ne a_2\} = \{s_1-s_2|s_1, s_2 \in S'_p,\ s_1 \ne s_2\} = \mathcal{M}_{S'_p}$. So $\forall t\ (1 \le t \le p-1)$, t has a multiplicity of $(p-3)/4$ in $\mathcal{M}_A$, which is equivalent to

$$\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-3)/4$$

for any $t,\ 1 \le t \le p-1$. Thus, $\{x_i\}_{0\le i\le p-1}$ satisfies condition (3) of Theorem 4.9.

From (1) to (3), $\{x_i\}_{0\le i\le p-1}$ is a read optimal and balanced recovery sequence when $k \in S_p$. □

From Theorem 4.16, we give the read optimal and balanced recovery scheme of Disk k, and it can be generalized as:

(1) Figure out the nonzero square set S_p and non-square set S'_p of F_p.
(2) If $k \in S_p$, let A be $S'_p - (k+1)$; if $k \notin S_p$, let A be $S_p - (k+1)$.
(3) For the failed Disk k, there is a read optimal and balanced recovery scheme corresponding to A, which means when $i \in A$, $d_{i,k}$ is recovered by diagonal parity and otherwise $d_{i,k}$ is recovered by row parity.

Now we use an example to explain the read optimal and balanced recovery scheme of Disk k $(k \ne p)$, when $p \equiv 3 \pmod 4$. Assume that $p = 7$, then $S_7 = \{i^2 \bmod 7 | 1 \le i \le (7-1)/2\} = \{1^2 \bmod 7, 2^2 \bmod 7, 3^2 \bmod 7\} = \{1,2,4\}$, and $S'_7 = F_7 \setminus (S_7 \cup \{0\}) = \{3,5,6\}$.

When Disk $k = 1$ fails, to recover the failed Disk 1, we need to find a read optimal and balanced recovery sequence $\{x_i\}_{0\le i\le 6}$. As $k = 1 \in S_7$, from Theorem 4.16, $A = S'_p - (k+1) = S'_p - (1+1) = \{(s-2) \bmod 7 | s \in S'_p\} = \{1,3,4\}$ corresponds to a recovery sequence $\{x_i\}_{0\le i\le 6} = \{0101100\}$. Thus, the read optimal and balanced recovery scheme is that symbols $d_{0,1}$, $d_{2,1}$, $d_{5,1}$ are recovered by row parity, and symbols $d_{1,1}$, $d_{3,1}$, $d_{4,1}$ are recovered by diagonal parity.

When Disk $k = 3$ fails, $k = 3 \notin S_7$, from Theorem 4.16, $A = S_p - (k+1) = S_p - (3+1) = \{(s-4) \bmod 7 | s \in S_p\} = \{0,4,5\}$ corresponds to a recovery sequence $\{x_i\}_{0\le i\le 6} = \{1000110\}$. Thus, the read optimal and balanced recovery scheme is that symbols $d_{1,3}$, $d_{2,3}$, $d_{3,3}$ are recovered by row parity, and symbols $d_{0,3}$, $d_{4,3}$, $d_{5,3}$ are recovered by diagonal parity.

The following Theorem 4.17 to Theorem 4.20 give a read optimal and balanced recovery scheme for any single disk failure with $p \equiv 1 \pmod 4$. They are similar to Theorem 4.9 to Theorem 4.16 of the case of $p \equiv 3 \pmod 4$. Therefore they can be proved in the similar way and we omit them for brevity.

THEOREM 4.17. *Let p be a prime number with $p \equiv 1 \pmod 4$. If a recovery sequence $\{x_i\}_{0\le i\le p-1}$ of Disk k $(0 \le k \le p-1)$ satisfies that:*

(1) $x_0 + x_1 + \cdots + x_{p-2} + x_{p-1} = (p-1)/2$*;*
(2) $x_{p-1} = x_{p-1-k} = 0$*;*

(3) $\forall t\,(1 \le t \le p-1)$, *either* $\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-1)/4$, *or* $\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-5)/4$.

$\{x_i\}_{0\le i\le p-1}$ *is a read optimal and balanced recovery sequence for the recovery of Disk* k $(0 \le k \le p-1)$ *in a disk array with RDP code.*

Definition 4.18. Let G be an additive group of order v and S be a subset of G with k elements. S is called a (v, k, λ, μ)-partial difference set of G, if each nonzero element $s \in S$ has a multiplicity of λ, and each nonzero element $s' \in G\backslash S$ has a multiplicity of μ in the multiset $\mathcal{M}_S = \{s_1 - s_2 | s_1, s_2 \in S,\ s_1 \ne s_2\}$.

THEOREM 4.19. *[Ma 1994] Let* $p \equiv 1 \pmod 4$ *be a prime number,* S_p *be the set of all nonzero squares in* F_p *and* S'_p *is the non-square set of* F_p*.* S_p *and* S'_p *both are* $(p, \frac{p-1}{2}, \frac{p-5}{4}, \frac{p-1}{4})$*-partial difference sets in the additive group of* F_p *.*

THEOREM 4.20. *Given a prime number* p *with* $p \equiv 1 \pmod 4$*. For the nonzero square set* S_p *and the non-square set* S'_p *of* F_p*,*

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0\le i\le p-1}$ *for the recovery of Disk* $k\ (k \ne p)$ *in RDP code storage systems.*

4.3. Optimal Recovery Scheme for Single Disk Failure of RDP code

In the last subsection, we presented a recovery combination which is read optimal and balanced for any erasure column $k\,(k \ne p)$. In the following, we propose the RDOR-RDP recovery scheme, which is a read optimal, balanced recovery scheme of RDP code and pre-stores $(p-1)/2$ extra symbols.

To recover any single column k, a read optimal and balanced recovery combination we choose is according to the recovery sequence demonstrated in Theorem 4.16 and Theorem 4.20. For $0 \le i \le p-2$, if $i \in A$, symbol $d_{i,k}$ is recovered by diagonal parity set $D_{<i+k>_p}$; otherwise $d_{i,k}$ is recovered by row parity set R_i. Any overlapping symbol d can be located in the disk array by calculating intersections of the selected row parity sets and diagonal parity sets for the recovery.

The recovery process of RDOR-RDP is shown in Algorithm 1, it is executed in a "*row-parity-first*" manner, i.e., we first recover the $(p-1)/2$ erasure symbols $d_{j,k}$ with $j \notin A$ $(0 \le j \le p-2)$ by R_j, and then recover the rest $(p-1)/2$ symbols $d_{i,k}$ with $i \in A$ $(0 \le i \le p-2)$. $(p-1)/2$ memory units are reserved to recover the $(p-1)/2$ symbols $d_{i,k}$ with $i \in A$.

According to RDOR-RDP, to recover $p-1$ erasure symbols in a stripe, we need to allocate $(p-1)/2$ memory units for overlapping symbols. Consider an RDP code storage system of $p = 19$, and a symbol size of 1M, the allocated memory size is 9 MB. Once the $p-1$ erased symbols in a stripe are successfully recovered, the allocated memory units will be released for the recovery of the next stripe. Thus, we can find that the allocated memory size for the recovery of a failed disk is only on the order of a few MBs. In practical storage systems, the extra memory usage of a few MBs will bring very little impact on the recovery performance.

We compare RDOR-RDP with the conventional recovery scheme of RDP code in the following three aspects.

(1) **Number of Disk Reads:** The conventional recovery performs $(p-1)^2$ disk reads while RDOR-RDP needs to read $3(p-1)^2/4$ symbols for recovery per stripe. RDOR-

ALGORITHM 1: Row-Diagonal Optimal Recovery of RDP code (RDOR-RDP)

```
1. $(p-1)/2$ memory units $M_i$ $(i \in A)$ are allocated for each erasure symbol $d_{i,k}$ $(i \in A)$ and the
   symbol $d_i$ stored in $M_i$ is initialized to 0.
2. Recover the $(p-1)/2$ symbols $d_{j,k}$ $(0 \le j \le p-2,\ j \notin A)$.
   (a) Recover symbol $d_{j,k}$ by XOR-summing all available symbols in $R_j$;
   (b) During the recovery process, for each symbol $d \in R_j$, if it is an overlapping symbol
       which should be used to recover some $d_{i,k}$ $(i \in A)$ later, it is XOR-summed with the
       symbol $d_i$ stored in its corresponding memory locations $M_i$ $(i \in A)$ (in other words,
       $d_i \leftarrow d_i \oplus d$).
3. Recover the rest $(p-1)/2$ symbols $d_{i,k}$ $(i \in A)$.
   (a) While recovering $d_{i,k}$ $(i \in A)$, for each symbol $d \in D_{<i+k>_p}$, if it is not an overlapping
       symbol, read it from disk and XOR-sum it to $d_i$.
   (b) After 3(a) finished, recovered symbol $d_{i,k}$ is stored in $M_i$.
4. Once the $p-1$ erasure symbols are successfully recovered, the recovery process can move
   on to the next stripe, and the $(p-1)/2$ memory units $M_i$ can be reused.
```

RDP can reduce by approximately 25% disk reads compared with the conventional scheme at the expense of a proper extra memory usage.

(2) **Read Balance:** Both RDOR-RDP and the conventional scheme achieve disk read balance during recovery.

(3) **Computational Complexity:** The computational cost of recovery scheme is determined as the total number of exclusive or (XOR) operations needed for recovery. Each parity set contains p symbols, so to recover any erasure symbol no matter which kind of parity was chosen for recovery, $p-2$ XOR operations on $p-1$ symbols are required. Thus the total number of XOR operations required for recovery of RDOR-RDP is equal to the conventional scheme.

5. THE RECOVERY OF SINGLE DISK FAILURE FOR EVENODD CODE

With the conventional recovery scheme of EVENODD code, whenever a single column is in erasure, it should read $(p-1) \times p$ symbols to recover it. Figure 7 gives an example of EVENODD code with $p = 5$, to recover the erasure column 0, $4 \times 5 = 20$ symbols need to be read. In this section, we derive the lower bound of disk reads for single disk failure recovery of EVENODD code and propose a single failure recovery scheme which achieves this lower bound.

5.1. Lower Bound of Disk Reads for Single Failure Recovery

Similar to the analysis of RDP code, we will firstly analyze the possible choices for the recovery of an erasure column in EVENODD code.

In EVENODD code, the row parity set R_i is defined as the set of all the information symbols which are used to generate $d_{i,p}$ $(0 \le i \le p-2)$ (*excluding the row parity symbol* $d_{i,p}$). Diagonal parity set D_j consists of all the information symbols which are used to generate $d_{j,p+1}$ $(0 \le j \le p-2)$ (*excluding the diagonal parity symbol* $d_{j,p+1}$). The formal descriptions of R_i and D_j are presented in Definition 5.1.

Definition 5.1.

(1) Define $R_i = \{d_{i,r} | 0 \le r \le p-1\}$ as the i-th row parity set, $0 \le i \le p-2$;
(2) Define $H = \{d_{i,p-1-i} | 0 \le i \le p-2\}$ as the adjuster set;
(3) Define $D'_j = \{d_{i,r} | (i+r) \bmod p = j,\ 0 \le i \le p-2,\ 0 \le r \le p-1\}$. $D_j = D'_j \cup H$ is called the j-th diagonal parity set, $0 \le j \le p-2$.

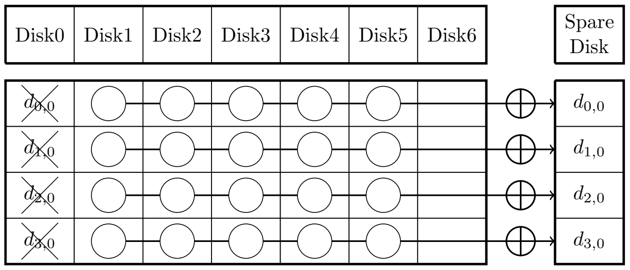


Fig. 7. Conventional scheme to recover Disk 0 of EVENODD code.

From Definition 5.1, $|R_i| = p$ for $0 \leq i \leq p-2$ and $|D_j| = |D'_j \cup H| = 2(p-1)$ for $0 \leq j \leq p-2$ (*Note that parity sets of EVENODD code only consist of information symbols*). Consider the example in Figure 3, $R_0 = \{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}\}$, and $D_0 = D'_0 \cup H = \{d_{0,0}, d_{3,2}, d_{2,3}, d_{1,4}\} \cup \{d_{3,1}, d_{2,2}, d_{1,3}, d_{0,4}\}$.

According to EVENODD code, given a symbol $d \in R_i$, d can be recovered by XOR-summing all the other symbols in $R_i - \{d\}$ and the row parity symbol $d_{i,p}$. Besides, for $d \in D_j$, d can be recovered by XOR-summing all the other symbols in $D_j - \{d\}$ and the diagonal parity symbol $d_{j,p+1}$. In the following, we say that d can be recovered from R_i or D_j.

Assume that the erasure column is column $k\,(0 \leq k \leq p+1)$, all symbols $d_{i,k}\,(0 \leq i \leq p-2)$ in this column are to be recovered. Based on EVENODD code, we have the following Lemma 5.2.

LEMMA 5.2. *Given an erasure column k,*

(*1*) $0 \leq k \leq p-1$, *if the symbol $d_{i,k} \in H$, it can be recovered from either R_i or D_{k-1}, and if the symbol $d_{i,k} \notin H$, it can be recovered from either R_i or $D_{<i+k>_p}$.*
(*2*) $k = p$, *column k is the row parity column. Symbol $d_{i,p}$ can only be recovered from R_i;*
(*3*) $k = p+1$, *column k is the diagonal parity column. Symbol $d_{i,p+1}$ can only be recovered from D_i.*

Since column p can only be recovered from row parity sets and column $p+1$ can only be recovered from diagonal parity sets, we only consider the recovery of column k with $k \neq p, p+1$ in the following.

Symbol $d_{i,k}$ can only be recovered from its parity sets. Similar to which in RDP code, to find the lower bound of disk reads for single disk failure recovery of EVENODD code is also equivalent to find a recovery combination with most overlapping symbols. Lemma 5.3 shows the possible overlapping symbols between two parity sets.

LEMMA 5.3.

(*1*) $R_i \cap D'_j = \{d_{i,<j-i>_p}\}$ *for* $0 \leq i, j \leq p-2$;
(*2*) $R_i \cap H = \{d_{i,p-1-i}\}$ *and* $D'_j \cap H = \emptyset$ *for* $0 \leq i, j \leq p-2$;
(*3*) $R_i \cap R_j = \emptyset$ *and* $D'_i \cap D'_j = \emptyset$ *for* $0 \leq i, j \leq p-2$, $i \neq j$.

According to Lemma 5.3, the following theorem gives a lower bound of disk reads for any single erasure column recovery.

THEOREM 5.4.

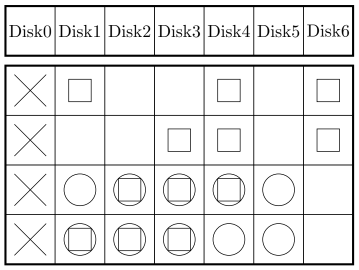


Fig. 8. Reducing number of reads to recover Disk 0 of EVENODD code.

(*1*) *For any erasure column* $k\,(k \neq p, p+1)$*, the minimum number of disk reads to recover column* k *of EVENODD code is* $(p-1)(3p+1)/4$.
(*2*) *Any recovery combination which consists of* $(p-1)/2$ *row parity sets and* $(p-1)/2$ *diagonal parity sets will match the minimum number of disk reads.*

PROOF. (1) $p-1$ symbols in column k need to be recovered, and erasure symbol $d_{i,k}\,(0 \leq i \leq p-2)$ in column k can be recovered either from row parity set or from diagonal parity set. Suppose that t erasure symbols are recovered from diagonal parity sets and the other $p-1-t$ symbols are recovered from row parity sets. According to Lemma 5.3, the number of overlapping symbols is

$$\sum_{i,j} |R_i \cap D_j| = \sum_{i,j} |R_i \cap D'_j| + \sum_{i} |R_i \cap H| = t(p-1-t) + (p-1-t).$$

Then the number of symbols to be read for recovery is

$$\begin{aligned} &(p-1-t)p + (t(p-1) + (p-1)) - (t(p-1-t) + (p-1-t)) \\ &= \left(t - \frac{p-1}{2}\right)^2 + \frac{(p-1)(3p+1)}{4}. \end{aligned}$$

When $t = (p-1)/2$, the number of disk reads for recovery is minimized which is equal to $(p-1)(3p+1)/4$. The read optimal recovery combination which matches the lower bound should consist of $(p-1)/2$ row and $(p-1)/2$ diagonal parity sets.

(2) The correctness of condition (2) follows directly from the proof of condition (1). □

From Theorem 5.4, a read optimal recovery scheme of EVENODD code can be stated as: for a single failed Disk $k\,(k \neq p, p+1)$, use row parity sets to recover any $(p-1)/2$ symbols, and use diagonal parity sets to recover the rest $(p-1)/2$ symbols.

Consider an example in Figure 8 with $p = 5$ and $k = 0$, one of the read optimal recovery combination to recover the four erasure symbols is (D_0, D_1, R_2, R_3), from which we need to read a number of 16 symbols for recovery. However, the number of read operations on individual disks of this recovery combination varies from 2 to 4. In the next subsection, we will present a recovery combination which matches the disk read lower bound and balances the disk reads.

5.2. Balancing Disk Reads

As the analysis of balancing disk reads for EVENODD code is similar to that of RDP code, the discussions and proofs of the similar results about EVENODD code are shown in the appendix. In this section, we only present the results.

The following theorem gives a read optimal and balanced recovery sequence $\{x_i\}_{0 \leq i \leq p-1}$ for any failed Diks $k\,(0 \leq k \leq p-1)$ of EVENODD code with $p \equiv 1 \pmod 4$.

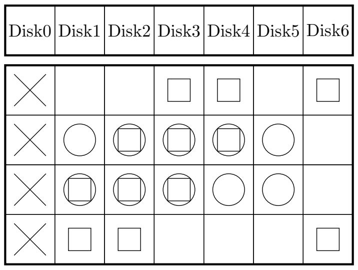


Fig. 9. Load balancing in recovering Disk 0 of EVENODD code.

THEOREM 5.5. *Given a prime number p with $p \equiv 1 \pmod 4$. For the nonzero square set S_p and the non-square set S'_p of F_p,*

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0 \le i \le p-1}$ for the recovery of Disk k $(k \neq p, p+1)$ in EVENODD code storage systems.

To illustrate, consider $p = 5$ and $k = 0$ as an example, $S_p = \{1, 4\}$ and $S'_p = \{2, 3\}$. Because $k \notin S_p$, $A = S_p - (k+1) = S_p - 1 = \{0, 3\}$ corresponds to a read optimal and balanced recovery sequence $\{10010\}$. See Figure 9 for an illustration. The following Theorem 5.6 gives a read optimal and balanced recovery scheme for any single disk failure with $p \equiv 3 \pmod 4$.

THEOREM 5.6. *Given a prime number p with $p \equiv 3 \pmod 4$. For the nonzero square set S_p and the non-square set S'_p of F_p,*

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0 \le i \le p-1}$ for the recovery of Disk $k(k \neq p, p+1)$ in EVENODD code storage systems.

5.3. Optimal Recovery Scheme for Single Disk Failure of EVENODD code

The read optimal and balanced recovery scheme of EVENODD code, RDOR-EVENODD, is very similar to that of RDP code. It needs one more memory unit to store the overlapping symbols in H than RDOR-RDP. The recovery combination we choose is according to the recovery sequence demonstrated in Theorem 5.5 and Theorem 5.6. The recovery process of RDOR-EVENODD is also executed in a "row-parity-first" way which is shown in Algorithm 2.

We also compare RDOR-EVENODD with the conventional recovery scheme of EVENODD code in the following three aspects.

(1) **Number of Disk Reads:** The conventional recovery of EVENODD code performs $p(p-1)$ disk reads while RDOR-EVENODD needs to read $(p-1)(3p+1)/4$ symbols for recovery per stripe. As

$$\frac{(p-1)(3p+1)/4}{p(p-1)} = \frac{3}{4} + \frac{1}{4p},$$

with a large number of p, RDOR-EVENODD can reduce by approximately 25% disk reads compared with the conventional scheme at the expense of a proper extra memory usage.

ALGORITHM 2: Row-Diagonal Optimal Recovery of EVENODD code (RDOR-EVENODD)

```
(1) $(p+1)/2$ memory units $M_i$ $(i \in A)$ and $M_h$ are allocated for each erasure symbol $d_{i,k}$ $(i \in A)$
    and adjuster $h$. The symbols $d_i$, $d_h$ stored in $M_i$ $(i \in A)$ and $M_h$ are all initialized to 0.
(2) Recover the $(p-1)/2$ symbols $d_{j,k}$ $(0 \leq j \leq p-2,\ j \notin A)$ from row parity set $R_j$. During
    recovery, for each overlapping symbol $d \in R_j$
    (a) If $d \in H$, it is XOR-summed with the symbol $d_h$ stored in $M_h$ $(d_h \leftarrow d_h \oplus d)$.
    (b) If $d \in D'_{<i+k>_p}$, which should be used to recover some $d_{i,k}$ $(i \in A)$ later, it is
        XOR-summed with the symbol $d_i$ stored in $M_i$ $(i \in A)$ $(d_i \leftarrow d_i \oplus d)$.
(3) Recover the rest $(p-1)/2$ symbols $d_{i,k}$ $(i \in A)$ from diagonal parity set $D_{<i+k>_p}$.
    (a) While recovering $d_{i,k}$ $(i \in A)$, for each symbol $d \in H$, if it is not an overlapping symbol,
        read it from disk and XOR-sum it to $d_h$.
    (b) For each symbol $d \in D'_{<i+k>_p}$, if it is not an overlapping symbol, read it from disk and
        XOR-sum it to $d_i$.
    (c) Symbol $d_h$ is XOR-summed to each $d_i$ $(i \in A)$ to reconstruct $d_{i,k}$ $(i \in A)$.
(4) Once the $p-1$ symbols are successfully recovered, it can move on to the next stripe, and the
    $(p+1)/2$ memory units can be reused.
```

(2) **Read Balance:** Both RDOR-EVENODD and the conventional scheme achieve disk read balance during recovery.
(3) **Computational Complexity:** The total number of XOR operations required for recovery of RDOR-EVENODD is equal to the conventional scheme.

6. PERFORMANCE EVALUATION

To evaluate the performance of our hybrid approach, we conduct experiments which compare the read optimal recovery schemes RDOR-RDP and RDOR-EVENODD with their conventional schemes respectively in terms of total recovery time and individual disk access time.

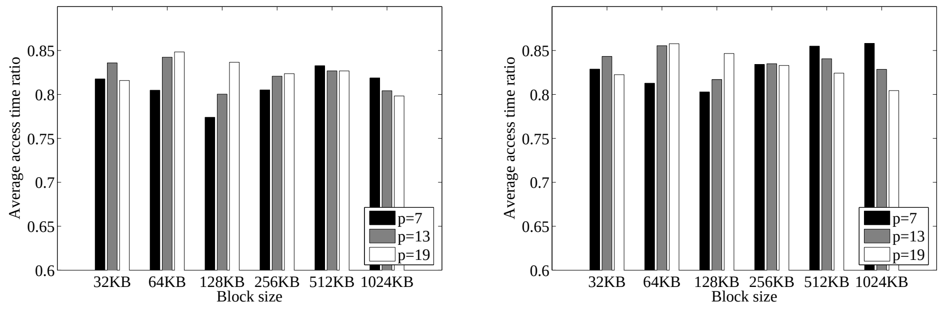
6.1. Evaluation Methodology

We conduct the experiments by using a widely accepted disk simulator, DiskSim [Bucy et al. 2008]. For RDP code, the simulated disk array subsystem is composed of $p+1$ disks including two redundant disks, and for EVENODD code there are $p+2$ disks in the simulated disk array subsystem. The logical layout is rotated between different parity stripes as in common RAID-6 implementations. Each disk is attached to a controller, and all these disk controllers are under the same interleaved I/O bus. The failed disk is reconstructed in an off-line mode, without any interruptions of outside requests while recovering. We take it as a simplified approach to compare the performance of RDOR-RDP and RDOR-EVENODD schemes with their conventional schemes under the same circumstances.

For the recovery of a single failed disk, the recovery schemes are implemented into the simulation environment. For each recovery scheme, during recovering each stripe, we first read all the data blocks needed for recovery and then generate the failed blocks. As soon as all the failure blocks are regenerated in memory, they are written to the spare disk at once. Two metrics, total recovery time and individual disk access time, are evaluated. We use recovery time ratio and average disk access time ratio to show the comparison results. The recovery time ratio and average disk access time ratio are the ratios of recovery time and average disk access time of our hybrid approach to the conventional scheme. Disk access time refers to the total time of a disk in media access during recovery. It consists of seek time, rotation time, transfer time, etc.

Table I. Disk array simulation parameters.

Parameter	Value
I/O bus bandwidth	512MBps
XOR bandwidth	1GBps
Disk capacity	146.8G
Sustainable disk transfer rate	99MBps
Rotation speed	15000RPM
Single track seek	0.3ms
Full seek	7.2ms



(a) RDOR-RDP vs. Conventional Scheme of RDP Code

(b) RDOR-EVENODD vs. Conventional Scheme of EVENODD code

Fig. 10. (a) Average disk access time ratio of RDOR-RDP to conventional scheme with different block sizes when $p = 7$, 13, and 19. (b) Average disk access time ratio of RDOR-EVENODD to conventional scheme with different block sizes when $p = 7$, 13, and 19.

Note that, the data block size and the disk array size affect the system recovery performance. Our experiments are conducted with different block sizes and different number of disks. The simulated disk's parameters are extracted from a modern enterprise hard disk [Seagate 2007], and Table I lists all the parameters used in this section.

6.2. Performance with Different Block Sizes

To evaluate the recovery performance under different block sizes, we vary the block size from 32KB to 1024KB with fixed $p = 7$, 13, and 19 respectively.

6.2.1. Disk access time. RDOR-RDP and RDOR-EVENODD improve the read access efficiency by reducing the number of disk reads in each surviving disk. Figure 10(a) shows that the average disk read access time of RDOR-RDP is reduced by $15.16\% \sim 22.60\%$ compared with the conventional recovery scheme of RDP code. However, the ratio of disk access time doesn't show a stable tendency with block size varying from 32KB to 1024KB.

Figure 10(b) shows the average disk access time ratio of RDOR-EVENODD with its conventional recovery scheme. From Figure 10(b), the average disk read access time of RDOR-EVENODD is reduced by $14.18\% \sim 19.73\%$ compared with the conventional recovery scheme.

RDOR-RDP and RDOR-EVENODD can reduce disk reads for recovery but change the access pattern of disk read from purely sequential accesses to a more random pattern, therefore incurs some additional disk seeks. The individual disk access time is determined not only by the amount of data read but also by the number of disk seeks and seek distances while reading. And it may be the main reason that the access time improvements shown in Figure 10(a) and Figure 10(b) are less than the theoretical value, approximately 25%.

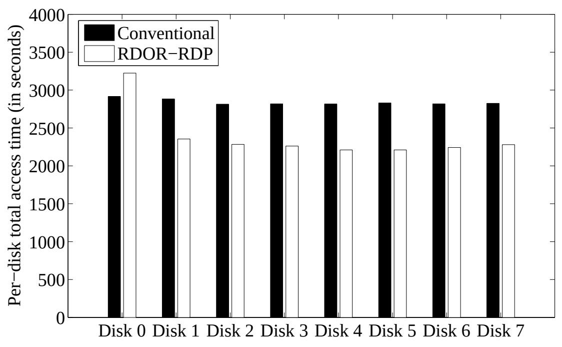


Fig. 11. Disk access time of RDOR-RDP and conventional scheme of each disk when $p = 7$ and block size is 32KB. Disk 0 is the failed and thereafter replaced disk.

Moreover, the sequentiality of reads on individual surviving disks in RDOR-RDP and RDOR-EVENODD are not the same. For an example of RDP code in Figure 5(b), reading all the needed data in Disk 1 is broken into 3 non-sequential read requests, while that of Disk 4 can be completed in only one read request. Therefore, the access time of reading in a stripe on individual surviving disk may be different. As parities are rotated between stripes, the access time of each disk for the recovery is balanced, which is shown in Figure 11.

Figure 11 shows the access time of each disk (Disk 0 is the failed and replaced disk) of RDOR-RDP scheme with $p = 7$ and block size of 32KB. From Figure 11, the access times of different disks (excluding the backup Disk 0) with RDOR-RDP are almost the same, and the average disk access time is decreased by 18.25% compared with conventional scheme.

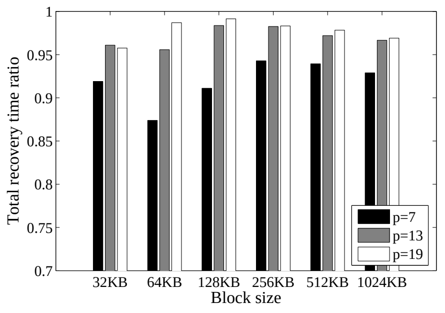
6.2.2. Total recovery time. Figure 12(a) and Figure 12(b) show the recovery time ratio of RDOR-RDP and RDOR-EVENODD to their conventional schemes respectively, from which we can see that RDOR-RDP and RDOR-EVENODD constantly outperform their conventional schemes as block size varies from 32KB to 1024KB.

For example, when $p = 7$, the total recovery time of RDOR-RDP is reduced by 5.72% $\sim$ 12.60%. With $p = 7$ and block size of 256KB, the total recovery times of RDOR-RDP and its conventional scheme are 6272.62s and 6652.92s respectively. And with $p = 7$ and block size of 1024KB, the total recovery times of RDOR-RDP and its conventional approach are 5337.87s and 5746.12s respectively.

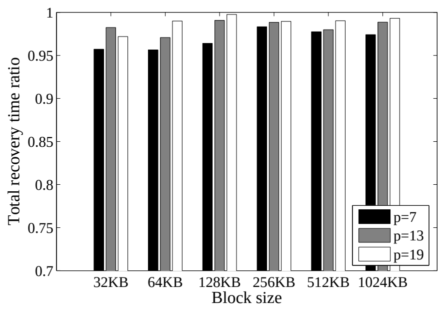
As shown in Figure 12(a), the improvements of recovery time decrease as block size varies from 32KB to 128KB first and then increase as block size varies from 128KB to 1024KB. Figure 12(b) shows that the total recovery time of RDOR-EVENODD is reduced by 1.67% $\sim$ 4.37% compared with the conventional recovery scheme of EVENODD code when $p = 7$.

However, in Figure 12(a) and Figure 12(b), the total recovery time ratios are mostly larger than 0.9. Although RDOR-RDP and RDOR-EVENODD can reduce approximately 25% disk reads, the total recovery time is bottlenecked by the slowest surviving disk in recovering each stripe (or stipe set). Also when there are no foreground request served, recovery buffer write to the spare disk becomes another bottleneck. Thus, the reduction on disk reads cannot be translated into equivalent saving of recovery time.

In the case of an storage system still serving the foreground requests while recovery, the whole system will go into a degraded mode and the recovery process usually runs in a lower priority. Then, reduction on disk read access time to surviving disks can be translated into more saving of recovery time in online scenario. The performance

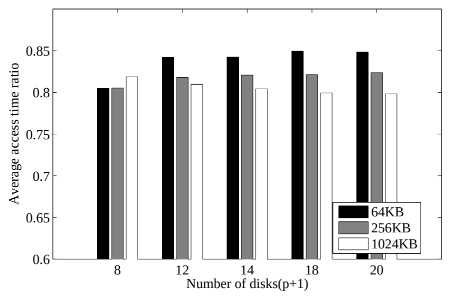


(a) RDOR-RDP vs. Conventional Scheme of RDP Code

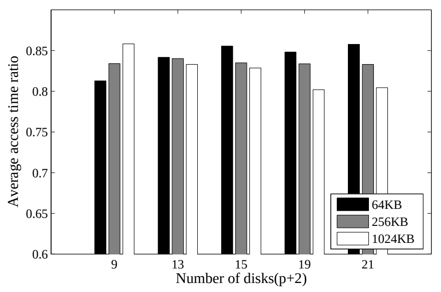


(b) RDOR-EVENODD vs. Conventional Scheme of EVENODD code

Fig. 12. (a) Recovery time ratio of RDOR-RDP to conventional scheme with different block sizes when $p = 7$, 13, and 19. (b) Recovery time ratio of RDOR-EVENODD to conventional scheme with different block sizes when $p = 7$, 13, and 19.



(a) RDOR-RDP vs. Conventional Scheme of RDP Code



(b) RDOR-EVENODD vs. Conventional Scheme of EVENODD code

Fig. 13. (a) Average disk access time ratio of RDOR-RDP to conventional scheme with different number of disks when block size is 64KB, 256KB, and 1024KB. (b) Average disk access time ratio of RDOR-EVENODD to conventional scheme with different number of disks when block size is 64KB, 256KB, and 1024KB.

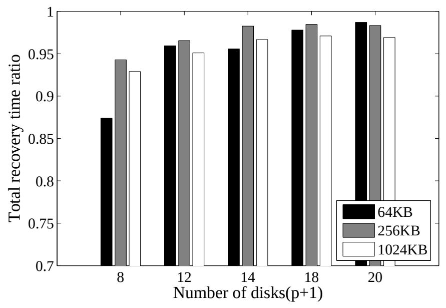
of RDOR-RDP and RDOR-EVENODD is expected to be better in online recovery than off-line.

6.3. Performance with Different Number of Disks

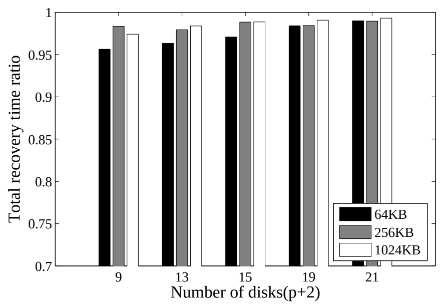
In this subsection, we analyze the impact of different number of disks on recovery performance. We conduct experiments with different number of p from 7 to 19 with fixed block sizes of 64KB, 256KB, and 1024KB respectively.

6.3.1. Disk access time. Figure 13(a) and Figure 13(b) show that the disk access time ratios are almost the same with different number of disks when the block size is fixed to 64KB, 256KB, and 1024KB respectively, which implies RDOR-RDP and RDOR-EVENODD can be implemented in fairly large storage systems to improve the read access efficiency.

As shown in Figure 13(a), with block size of 1024KB, the disk access time of RDOR-RDP is reduced by 18.13% $\sim$ 20.17%. With block size of 1024KB, the disk access time of RDOR-EVENODD which is shown in Figure 13(b) is reduced by 14.18% $\sim$ 19.80%.



(a) RDOR-RDP vs. Conventional Scheme of RDP Code



(b) RDOR-EVENODD vs. Conventional Scheme of EVENODD code

Fig. 14. (a) Recovery time ratio of RDOR-RDP to conventional scheme with different number of disks when block size is 64KB, 256KB, and 1024KB. (b) Recovery time ratio of RDOR-EVENODD to conventional scheme with different number of disks when block size is 64KB, 256KB, and 1024KB.

6.3.2. Total recovery time. In Figure 14(a) and Figure 14(b), the total recovery time ratios increase as the number of disks increase. In Figure 14(a), with block size of 1024KB, the total recovery time ratio increases from 0.929 to 0.971 as the number of disks increases from 8 to 20. With block size of 1024KB and $p = 7$, the total recovery times of RDOR-RDP and its conventional approach are 5337.87s and 5746.12s respectively. With block size of 1024KB and $p = 17$, the total recovery times of RDOR-RDP and its conventional approach are 9432.54s and 9714.97s respectively.

Since the access time of reading in a stripe on individual surviving disk with RDOR-RDP may be different and the write requests to the spare disk should wait until the data has been reconstructed, the recovery process of RDOR-RDP may cost more time in waiting and synchronizing with large number of disks. Thus, RDOR-RDP and RDOR-EVENODD provide comparatively smaller improvements in recovery time with large number of disks.

The experimental results can be generalized as follows.

(1) By using our hybrid approach, the access time of an individual disk for recovery is reduced compared with the conventional scheme. Less disk access time indicates that the disk array system can further accelerate the recovery process in online recovery scenario.
(2) Our hybrid approach can be used to reduce the total recovery time compared with the conventional scheme. With block size varies from 32KB to 1024KB, RDOR-RDP and RDOR-EVENODD constantly outperforms their conventional recovery schemes which indicate that RDOR-RDP and RDOR-EVENODD scale well as block size increases.

7. FURTHER DISCUSSIONS

According to the experiment results, RDOR-RDP and RDOR-EVENODD don't show great improvements on recovery time compared with their conventional approaches. There are mainly two reasons, the first is that RDOR-RDP and RDOR-EVENODD change the recovery access pattern from pure sequential to a more random one. Thus, in off-line mode, RDOR-RDP and RDOR-EVENODD will cost more time on disk seeking and rotating compared with their conventional approaches. The second is that the recovery time is mainly composed of two parts, the time of reading symbols from surviving disks and the time of writing recovered erased symbols to a spare disk. Thus, the reduction on disk reads cannot be translated into equivalent saving of recovery time.

The above reasons indicates that RDOR-RDP and RDOR-EVENODD will be more applicable in the following cases.

1) There are foreground workloads during recovery. In online recovery mode, the foreground requests will interrupt the recovery process which will also break the sequentiality of disk access of the conventional approach.
2) RAID-6 systems use flash solid-state-drives (SSDs) instead of hard disk drives (HDDs). Commonly used HDDs store data on rapidly rotating platters with magnetic surfaces. The disk access time is composed of seek time, rotational latency, and data transfer time. In contrast to HDDs, flash SSDs do not have any mechanically moving parts but use NAND flash memory chips to store the data [Emrich et al. 2010]. So, the random disk access of RDOR-RDP and RDOR-EVENODD may not be an important problem during recovery, which indicates RDOR-RDP and RDOR-EVENODD may perform better with SSDs than traditional HDDs.
3) Peer-to-peer distributed storage systems. RDOR-RDP and RDOR-EVENODD can also be extended to *distributed* storage systems based on RAID-6 codes. In *distributed* storage systems, to repair a failed node, a key goal is to minimize the repair bandwidth [Dimakis et al. 2010]. RDOR-RDP and RDOR-EVENODD can be used to reduce the amount of data which need to be transmitted in the system for recovery.
4) RDOR-RDP and RDOR-EVENODD are applied together with clustered RAID. RDOR-RDP and RDOR-EVENODD focus on reducing the amount of data needs to be read from surviving disks. However, during recovery, writing recovered data to a spare disk is another bottleneck. The proposed clustered RAID [Muntz and Lui 1990] organization in which data plus parity are distributed over a larger number of disks achieves that the recovery writes are distributed among all the surviving disks. So RDOR-RDP and RDOR-EVENODD can be applied together with the clustered RAID technique to further improve the recovery performance.

8. RELATED WORK

System designers expect better recovery solutions to minimize the time taken for recovery and the degradation of user performance in RAID storage systems. Recent years a large number of approaches have been proposed for recovering the failed disk more efficiently.

Several approaches [Holland and Gibson 1992; Holland et al. 1994; Xin et al. 2004; Greenan et al. 2010] address the problem by trading storage capacity for improved failure recovery performance. Menon and Mattson [1992] showed that instead of using a dedicated spare disk, there are many advantages to distributing the spare space across the disk array. In such an array, the recovery writes are distributed as well as the recovery reads. Clustered RAID [Muntz and Lui 1990], in which data plus parity groups are distributed over a larger number of disks, was introduced to reduce the increased load per disk.

To improve recovery parallelism, the Disk-Oriented Reconstruction (DOR) [Holland et al. 1994] algorithm executes reconstruction processes associated with disks, which keeps every surviving disk busy with reconstruction reads at all time. In another work by Lee and Lui [2002], a pipelined recovery algorithm is proposed to take advantage of the sequential property of track retrievals to pipeline the reading and writing processes.

In online recovery mode, the foreground requests will interfere with the recovery process [Merchant and Yu 1996; Thomasian and Menon 1997]. To relieve the interference of foreground workload, Wu et al. [2009] proposed the WorkOut scheme which temporarily redirects all write requests and popular read requests to a surrogate RAID set to let the degraded RAID set be devoted to the recovery process. A Popularity-based multi-threaded reconstruction optimization (PRO) algorithm [Tian et al. 2007] was proposed to accelerate the users' accesses and avoid performance degradation by recovering the high-popularity data units prior to other units.

In contrast to the above recovery mechanisms which focus on reorganizing data layout or optimizing recovery workflow, there are some other researches focused on designing specific recovery algorithms for erasure codes according to the coding characteristics. By using generator check matrix of erasure codes, Hafner et al. [2005] proposed a code-specific hybrid reconstruction algorithm which can reduce XOR operations during recovery. Due to the reduction of XOR operations, the performance of decoding was improved. Later, a revised version of this algorithm was proposed in [Cassidy and Hafner 2007] to further improve space and time efficiency during recovery.

9. CONCLUSION

We propose a hybrid recovery approach for single disk failure of RAID-6 codes. The hybrid approach uses both parities for data reconstruction and reduces the number of disk reads for recovery by pre-storing overlapping symbols in memory. We derive the lower bounds of disk reads for the recovery of RDP and EVENODD codes and design optimal recovery schemes, RDOR-RDP and RDOR-EVENODD, which match the lower bound of disk reads, as well as having a load balancing property on all surviving disks.

Compared with the conventional scheme, RDOR-RDP and RDOR-EVENODD disrupt the sequentiality of disk reads for recovery in an off-line mode. The off-line experimental results show that our recovery schemes RDOR-RDP and RDOR-EVENODD outperform the conventional recovery schemes of RDP and EVENODD codes in terms of total recovery time and recovery workload on individual surviving disk. However, the improvements are less than the theoretical value. Our future works include evaluating the performance of hybrid approach in online recovery mode, and extending the hybrid approach to parity array codes that can tolerate triple failures, such as STAR, Extended RDP code, and so on.

APPENDIX

A. PROOF OF THE CORRECTNESS OF THEOREM 4.7 AND THEOREM 5.4

In here, we will prove that only when all the surviving symbols are read directly from disk, the number of disk reads can achieve the lower bounds in Theorem 4.7 and Theorem 5.4. For brevity, in the following, we only prove that of RDP code, the analysis of EVENODD code is similar and omitted.

Let the erasure column be column k $(0 \leq k \leq p-1)$. We need to recover all the $p-1$ symbols $d_{i,k}(0 \leq i \leq p-2)$ in column k. Erasure symbol $d_{i,k}$ can only be recovered by the parity sets to which it belongs. Surviving symbols for recovery are either read directly from disk or further generated from other parity sets.

Assume that in the recovery process, s surviving symbols are generated from other parity sets instead of being read directly from disks. There are $p-3$ additional symbols needed to be read for each of the s symbols. Therefore, $p-1+s$ parity sets are needed in total and consist of $(p-1)^2 + s(p-3)$ symbols (*Note: There are some symbols counted twice which will be subtracted later*).

Among all the $p-1+s$ parity sets, suppose there are t row parity sets and $p-1+s-t$ diagonal parity sets. According to Lemma 4.6, there is one overlapping symbol between each pair of row and diagonal parity sets. So there are $t(p-1+s-t)-2s$ overlapping symbols (excluding the s surviving symbols which are generated by parity sets and the s symbols which overlap at the failed disk).

So the number of disk reads for the recovery is

$$
\begin{aligned}
&(p-1)^2 + s(p-3) - (t(p-1+s-t) - 2s) \\
=&(t-(p-1+s)/2)^2 + 3(p-1)^2/4 + s(2p-2-s)/4.
\end{aligned}
$$

When $t = (p-1+s)/2$, the number of disk reads is minimized, which is equal to $3(p-1)^2/4 + s(2p-2-s)/4$. As RDP can only tolerate double disk failures, $0 \le s < 2p-2$, only when $s = 0$ and $t = (p-1)/2$,

$$3(p-1)^2/4 + s(2p-2-s)/4 = 3(p-1)^2/4,$$

which achieves the lower bound in Theorem 4.7. Therefore, no matter how we get surviving symbols for recovery, the lower bound of disk reads is $3(p-1)^2/4$.

B. BALANCING DISK READS OF EVENODD CODE

For EVENODD code, the minimum number of disk reads for the recovery of column $k\,(0 \le k \le p-1)$ is $(p-1)(3p+1)/4$. Since any read optimal recovery combination contains $(p-1)/2$ row and $(p-1)/2$ diagonal parity sets, $(p-1)/2$ symbols will always be read from Disk p and Disk $p+1$ respectively. Then the average number of symbols to be read from Disk j $(0 \le j \le p-1, j \ne k)$ is

$$\frac{(p-1)(3p+1)/4 - (p-1)}{p-1} = 3(p-1)/4.$$

As $(p-1)/2$ symbols will always be read from Disk p and Disk $p+1$, now we only consider how to balance disk reads on Disk $j\,(0 \le j \le p-1, j \ne k)$ in the following. Given a prime number $p > 2$, either $p \equiv 3 \pmod 4$ or $p \equiv 1 \pmod 4$.

Case 1. $p \equiv 1 \pmod 4$, then $3(p-1)$ is divisible by 4. A read optimal and balanced recovery combination should read $3(p-1)/4$ symbols from each of the surviving disks (*except for Disk* p *and Disk* $p+1$).

Case 2. $p \equiv 3 \pmod 4$, then $3(p-1)$ isn't divisible by 4. A read optimal and balanced recovery combination should read $\lceil 3(p-1)/4 \rceil$ symbols from some disks and $\lfloor 3(p-1)/4 \rfloor$ symbols from the rest.

In the following, we will only give the detailed analysis of the case $p \equiv 1 \pmod 4$. The analysis of the case $p \equiv 3 \pmod 4$ is similar and omitted.

Similar to the analysis of RDP code, we also assume that there is an imaginary row $p-1$ with all symbols $d_{p-1,j} = 0$ $(0 \le j \le p+1)$ to the EVENODD encoding array. With the additional row, the encoding array is now a $p \times (p+2)$ array. Since all symbols in row $p-1$ are 0, $d_{p-1,k}$ can be regarded as being recovered from its row parity set. And for simplicity, we consider that the symbol $d_{p-1-k,k} \in H$ is recovered from its row parity set.

Now the recovery sequence of EVENODD code is $x_0, x_1, \ldots, x_{p-2}, x_{p-1}$ with $x_{p-1-k} = 0$ and $x_{p-1} = 0$, where $x_i = 0$ means that $d_{i,k}$ is recovered from its row parity set, and $x_i = 1$ means that $d_{i,k}$ is recovered from its diagonal parity set. The following Theorem B.1 provides a sufficient condition for a recovery sequence to be both read optimal and balanced of EVENODD code.

THEOREM B.1. *If a recovery sequence* $\{x_i\}_{0 \le i \le p-1}$ *of Disk* k $(0 \le k \le p-1)$ *satisfies that:*

(*1*) $x_0 + x_1 + \cdots + x_{p-2} + x_{p-1} = (p-1)/2$*;*
(*2*) $x_{p-1-k} = x_{p-1} = 0$*;*

(3) $\forall j$, $0 \leq j \leq p-1$, $j \neq k$, $\sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} + x_{p-1-j} = (p-1)/4$*;*

$\{x_i\}_{0 \leq i \leq p-1}$ *is a read optimal and balanced recovery sequence for the recovery of Disk* $k\,(0 \leq k \leq p-1)$.

PROOF. Condition (1) and condition (2) hold is equivalent to that $\{x_i\}_{0 \leq i \leq p-1}$ is a read optimal recovery sequence. In the following of the proof, we concentrate on condition (3).

Define $c_{i,j}$ to identify whether symbol $d_{i,j}$ is read for recovery. If $d_{i,j}$ is read for recovery, $c_{i,j} = 0$; otherwise $c_{i,j} = 1$ $(0 \leq i, j \leq p-1, j \neq k)$.

If $x_i = 0$, symbol $d_{i,k}$ is recovered by row parity set R_i, $d_{i,j}$ is read to recover $d_{i,k}$ because $d_{i,j} \in R_i$. If $x_{<i+j-k>_p} = 1$, symbol $d_{<i+j-k>_p,k}$ is recovered by diagonal parity set $D_{<i+j>_p}$, $d_{i,j}$ is read to recover $d_{<i+j-k>_p,k}$ because $d_{i,j} \in D_{<i+j>_p}$. Given $d_{i,j}$ ($d_{i,j} \notin H$), either $d_{i,j} \in R_i$ or $d_{i,j} \in D_{<i+j>_p}$. So only when $x_i = 0$ or $x_{<i+j-k>_p} = 1$, $d_{i,j}$ is read for recovery, and $c_{i,j} = 0$. Therefore, $c_{i,j} = x_i(1 - x_{<i+j-k>_p})$.

Because all symbols $d_{p-1-j,j} \in H$ are always need for recovery, $c_{p-1-j,j} = 0$ for $0 \leq j \leq p-1$ $(j \neq k)$. Therefore,

$$c_{i,j} = \begin{cases} x_i(1 - x_{<i+j-k>_p}) & <i+j>_p \neq p-1 \\ 0 & <i+j>_p = p-1 \end{cases} \tag{4}$$

As $c_{p-1-j,j} = 0$ for $0 \leq j \leq p-1$ $(j \neq k)$, the number of symbols which do not need to be read from Disk $j (0 \leq j \leq p-1, j \neq k)$ for the recovery of Disk k is

$$\begin{aligned}\sum_{i=0}^{p-1} c_{i,j} = \sum_{i=0, i \neq p-1-j}^{p-1} c_{i,j} &= \sum_{i=0, i \neq p-1-j}^{p-1} x_i(1 - x_{<i+j-k>_p}) \\ &= \sum_{i=0}^{p-1} x_i(1 - x_{<i+j-k>_p}) - x_{p-1-j}(1 - x_{<(p-1-j)+j-k>_p}) \\ &= \sum_{i=0}^{p-1} x_i(1 - x_{<i+j-k>_p}) - x_{p-1-j}(1 - x_{p-1-k}).\end{aligned}$$

According to Condition (2), $x_{p-1-k} = 0$, so

$$\begin{aligned}\sum_{i=0}^{p-1} c_{i,j} &= \sum_{i=0}^{p-1} x_i(1 - x_{<i+j-k>_p}) - x_{p-1-j} \\ &= \sum_{i=0}^{p-1} x_i - \sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} - x_{p-1-j}\end{aligned}$$

As $\sum_{i=0}^{p-1} x_i = (p-1)/2$, and according to condition (3), $\sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} + x_{p-1-j} = (p-1)/4$, we have

$$\sum_{i=0}^{p-1} c_{i,j} = \sum_{i=0}^{p-1} x_i - (\sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} + x_{p-1-j}) = (p-1)/4.$$

Then the number of symbols need to be read from Disk $j(0 \leq j \leq p-1, j \neq k)$ is $(p-1) - (p-1)/4 = 3(p-1)/4$. Therefore, condition (3) holds is equivalent to that $\{x_i\}_{0 \leq i \leq p-1}$ is a balanced recovery sequence for Disk $k\,(0 \leq k \leq p-1)$.

Theorem B.1 holds. □

According to the condition (3) of Theorem B.1, $\forall j$, $0 \le j \le p-1$, $j \ne k$,

$$\sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} = \begin{cases} (p-1)/4 & x_{p-1-j} = 0, \\ (p-5)/4 & x_{p-1-j} = 1. \end{cases} \tag{5}$$

Given a recovery sequence $\{x_i\}_{0\le i\le p-1}$, define $A_x = \{i|x_i = 1, 0 \le i \le p-1\}$ and the multiset $\mathcal{M}_{A_x} = \{a_1 - a_2|a_1, a_2 \in A_x, a_1 \ne a_2\}$. According to Lemma 4.12, Equation (5) is equivalent to that for $x_{p-1-j} = 0$, the multiplicity of $j-k$ in $\mathcal{M}_{A_x}$ is $(p-1)/4$, and for $x_{p-1-j} = 1$, the multiplicity of $j-k$ in $\mathcal{M}_{A_x}$ is $(p-5)/4$.

Consider an example of $p = 5$ and $k = 0$, Figure 9 shows a read optimal and balanced recovery sequence $\{x_i\}_{0\le i\le 4} = \{10010\}$. According to the recovery sequence, $x_0 = x_3 = 1$, $A_x = \{0, 3\}$, and $\mathcal{M}_{A_x} = \{a_1 - a_2|a_1, a_2 \in A_x,\ a_1 \ne a_2\} = \{2, 3\}$.

For $x_{p-1-j} = x_0 = x_3 = 1$, $p-1-j = 0$ and $p-1-j = 3$, so $j = 4$ and $j = 1$. According to Equation (5), $j-k = 4$ and $j-k = 1$ has a multiplicity of $(p-5)/4 = 0$ in the multiset $\mathcal{M}_{A_x} = \{2, 3\}$. For $x_{p-1-j} = x_1 = x_2 = 0$, $p-1-j = 1$ and $p-1-j = 2$, so $j = 3$ and $j = 2$. According to Equation (5), $j-k = 3$ and $j-k = 2$ has a multiplicity of $(p-1)/4 = 1$ in the multiset $\mathcal{M}_{A_x} = \{2, 3\}$.

The following theorem gives a read optimal and balanced recovery sequence $\{x_i\}_{0\le i\le p-1}$ for any failed Diks k $(0 \le k \le p-1)$ of EVENODD code with $p \equiv 1 \pmod 4$.

THEOREM B.2. *Given a prime number* p *with* $p \equiv 1 \pmod 4$. *For the nonzero square set* S_p *and the non-square set* S'_p *of* F_p,

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0\le i\le p-1}$ *for the recovery of Disk* k $(k \ne p, p+1)$ *in EVENODD code storage systems.*

PROOF. We only prove the case of $k \in S_p$. The proof of the case of $k \notin S_p$ is similar.

According to $A = S'_p - (k+1)$, define a recovery sequence $\{x_i\}_{0\le i\le p-1}$ as that if $i \in A$, $x_i = 1$, otherwise $x_i = 0$. We need to prove that $\{x_i\}_{0\le i\le p-1}$ satisfies the three conditions of Theorem B.1.

(1) $|A| = |S'_p| = (p-1)/2$, $x_0 + x_1 + \cdots + x_{p-2} + x_{p-1} = (p-1)/2$. So $\{x_i\}_{0\le i\le p-1}$ satisfies condition (1) of Theorem B.1.

(2) Given $k \in S_p$ and $A = S'_p - (k+1)$,

$$k \in S_p \Rightarrow k \notin S'_p \Rightarrow k-(k+1) \notin S'_p - (k+1) \Rightarrow p-1 \notin A \Rightarrow x_{p-1} = 0$$
$$p \notin S'_p \Rightarrow p-(k+1) \notin S'_p - (k+1) \Rightarrow p-(k+1) \notin A \Rightarrow x_{p-1-k} = 0$$

So $\{x_i\}_{0\le i\le p-1}$ satisfies condition (2) of Theorem B.1. From (1) and (2), $A = S'_p-(k+1)$ corresponds to a read optimal sequence $\{x_i\}_{0\le i\le p-1}$.

(3) According to Definition 4.18 and Theorem 4.19, S'_p is a $(p, \frac{p-1}{2}, \frac{p-5}{4}, \frac{p-1}{4})$-partial difference set in the additive group of F_p when $p \equiv 1 \pmod 4$. So $\forall t \in S'_p$, t has a multiplicity of $(p-5)/4$ in the multiset $\mathcal{M}_{S'_p} = \{s_1 - s_2|s_1, s_2 \in S'_p, s_1 \ne s_2\}$, and $\forall t \notin S'_p$, t has a multiplicity of $(p-1)/4$ in the multiset $\mathcal{M}_{S'_p}$

Because $A = S'_p - (k+1)$, multiset $\mathcal{M}_A = \{a_1 - a_2|a_1, a_2 \in A,\ a_1 \ne a_2\} = \{s_1 - s_2|s_1, s_2 \in S'_p,\ s_1 \ne s_2\} = \mathcal{M}_{S'_p}$. So $\forall t \in S'_p$, t has a multiplicity of $(p-5)/4$ in $\mathcal{M}_A$, and $\forall t \notin S'_p$ $(t \ne 0)$, t has a multiplicity of $(p-1)/4$ in $\mathcal{M}_A$, which is equivalent to $\forall t \ne 0$,

$$\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = \begin{cases} (p-5)/4 & t \in S'_p \\ (p-1)/4 & t \notin S'_p \end{cases} \tag{6}$$

Since $p \equiv 1 \pmod 4$,

i) $t \in S'_p \Rightarrow p - t \in S'_p \Rightarrow p - 1 - <t+k>_p \in S'_p - (k+1) = A$;
ii) $t \notin S'_p \Rightarrow p - t \notin S'_p \Rightarrow p - 1 - <t+k>_p \notin S'_p - (k+1) = A$.

Therefore, $\forall t \neq 0$

$$x_{p-1-<t+k>_p} = \begin{cases} 1 & t \in S'_p \\ 0 & t \notin S'_p \end{cases} \tag{7}$$

Add (7) to (6), and substitute t for $<j-k>_p$, we have $\forall j,\ 0 \leq j \leq p-1,\ j \neq k$,

$$\sum_{i=0}^{p-1} x_i x_{<i+j-k>_p} + x_{p-1-j} = (p-1)/4$$

From (1) to (3), $\{x_i\}_{0 \leq i \leq p-1}$ is a read optimal and balanced recovery sequence for Disk k $(0 \leq k \leq p-1)$. □

To illustrate, consider $p = 5$ and $k = 0$ as an example, $S_p = \{1, 4\}$ and $S'_p = \{2, 3\}$. Because $k \notin S_p$, $A = S_p - (k+1) = S_p - 1 = \{0, 3\}$ corresponds to a read optimal and balanced recovery sequence $\{10010\}$. See Figure 9 for an illustration. The following Theorem 5.6 gives a read optimal and balanced recovery scheme for any single disk failure with $p \equiv 3 \pmod 4$. Its proof is similar to Theorem B.2 and therefore omitted.

THEOREM B.3. *Given a prime number p with $p \equiv 3 \pmod 4$. For the nonzero square set S_p and the non-square set S'_p of F_p,*

$$A = \begin{cases} S'_p - (k+1) & k \in S_p \\ S_p - (k+1) & k \notin S_p \end{cases}$$

corresponds to a read optimal and balanced recovery sequence $\{x_i\}_{0 \leq i \leq p-1}$ for the recovery of Disk $k (k \neq p, p+1)$ in EVENODD code storage systems.

REFERENCES

BAKER, M., SHAH, M., ROSENTHAL, D. S. H., ROUSSOPOULOS, M., MANIATIS, P., GIULI, T., AND BUNGALE, P. 2006. A fresh look at the reliability of long-term digital storage. In *Proceedings of the 2006 EuroSys Conference, (EuroSys'06)*. ACM, Leuven, Belgium, 221–234.

BLAUM, M., BRADY, J., BRUCK, J., AND MENON, J. 1995. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers 44,* 2, 192–202.

BUCY, J., SCHINDLER, J., SCHLOSSER, S., AND GANGER, G. 2008. The DiskSim simulation environment (v4.0). Tech. Rep. CMU_PDL_08_101, Carnegie Melon University.

CASSIDY, B. AND HAFNER, J. L. 2007. Space efficient matrix methods for lost data reconstruction in erasure codes. Tech. Rep. RJ10415, IBM Research.

CHEN, P. M., LEE, E. K., GIBSON, G. A., KATZ, R. H., AND PATTERSON, D. A. 1994. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys 26*, 145–185.

CORBETT, P., ENGLISH, B., GOEL, A., GRCANAC, T., KLEIMAN, S., LEONG, J., AND SANKAR, S. 2004. Row-diagonal parity for double disk failure correction. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST'04)*. USENIX Association, San Francisco, CA, 1–14.

DIMAKIS, A. G., GODFREY, P. B., WU, Y., WAINWRIGHT, M. J., AND RAMCHANDRAN, K. 2010. Network coding for distributed storage systems. *IEEE Transactions on Information Theory 56*, 4539–4551.

EMRICH, T., GRAF, F., KRIEGEL, H.-P., SCHUBERT, M., AND THOMA, M. 2010. On the impact of flash ssds on spatial indexing. In *Proceedings of the Sixth International Workshop on Data Management on New Hardware (DaMoN'10)*. ACM, New York, NY, USA, 3–8.

GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. 2003. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*. ACM, New York, NY, USA, 29–43.

GREENAN, K. M., LI, X., AND WYLIE, J. J. 2010. Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs. In *Procceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST'10)*. IEEE, Los Alamitos, CA, USA, 1–14.

HAFNER, J. L., DEENADHAYALAN, V., RAO, K. K., AND TOMLIN, J. A. 2005. Matrix methods for lost data reconstruction in erasure codes. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST'05)*. USENIX Association, Berkeley, CA, USA, 15–30.

HOLLAND, M. 1994. On-line data reconstruction in redundant disk arrays. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

HOLLAND, M. AND GIBSON, G. A. 1992. Parity declustering for continuous operation in redundant disk arrays. In *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V)*. ACM, New York, NY, USA, 23–35.

HOLLAND, M., GIBSON, G. A., AND SIEWIOREK, D. P. 1993. Fast, on-line failure recovery in redundant disk arrays. In *Proceedings of the 23rd Annual International Symposium on Fault-Tolerant Computing (FTCS'93)*. Toulouse, France, 422–431.

HOLLAND, M., GIBSON, G. A., AND SIEWIOREK, D. P. 1994. Architectures and algorithms for on-line failure recovery in redundant disk arrays. *Distributed and Parallel Databases 2,* 3, 295–335.

JOUKOV, N., KRISHNAKUMAR, A. M., PATTI, C., RAI, A., SATNUR, S., TRAEGER, A., AND ZADOK, E. 2007. RAIF: Redundant array of independent filesystems. In *Proceedings of 24th IEEE Conference on Mass Storage Systems and Technologies (MSST'07)*. IEEE, San Diego, CA, 199–212.

KUBIATOWICZ, J., BINDEL, D., CHEN, Y., CZERWINSKI, S., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. 2000. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*. New York, NY, USA, 190–201.

LEE, J. Y. B. AND LUI, J. C. S. 2002. Automatic recovery from disk failure in continuous-media servers. *IEEE Transactions on Parallel Distributed Systems 13,* 5, 499–515.

LUETH, C. 2004. RAID-DP: Network Appliance implementation of RAID double parity for data protection. Tech. Rep. No. 3298, Network Appliance Inc.

LYMAN, P. AND VARIAN, H. R. 2003. How much information? http://www.sims.berkeley.edu/how-much-info-2003.

MA, S. L. 1994. A survey of partial difference sets. *Des. Codes Cryptography 4,* 3, 221–261.

MENON, J. AND MATTSON., D. 1992. Comparison of sparing alternative for disk arrays. In *Proceedings of the International Symposium on Computer Architecture (ISCA'92)*. New York, NY, USA, 318–329.

MERCHANT, A. AND YU, P. S. 1996. Analytic modeling of clustered RAID with mapping based on nearly random permutation. *IEEE Transactions on Computers 45,* 3, 367–373.

MUNTZ, R. R. AND LUI, J. C. S. 1990. Performance analysis of disk arrays under failure. In *Proceedings of the 16th International Conference on Very large Databases (VLDB'90)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 162–173.

PINHEIRO, E., WEBER, W.-D., AND BARROSO, L. A. 2007. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*. USENIX Association, Berkeley, CA, USA, 17–28.

PLANK, J. S. 2008. The RAID-6 liberation codes. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08)*. USENIX Association, Berkeley, CA, USA, 1–14.

PLANK, J. S., LUO, J., SCHUMAN, C. D., XU, L., AND WILCOX-O'HEARN, Z. 2009. A performance evaluation and examination of open-source erasure coding libraries for storage. In *Proccedings of the 7th conference on File and Storage Technologies (FAST'09)*. USENIX Association, Berkeley, CA, USA, 253–265.

PLESS, V. 1998. *Introduction to the Theory of Error-Correcting Codes*. Wiley Interscience.

SCHROEDER, B. AND GIBSON, G. A. 2007. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*. USENIX Association, Berkeley, CA, USA, 1–16.

SEAGATE. 2007. Cheetah® 15K.5 Fibre Channel 146-GB Hard Drive ST3146855FC Product Manual. Tech. Rep. Cheetah 15K.5 FC, Seagate Inc.

THOMASIAN, A. AND MENON, J. 1997. RAID-5 performance with distributed sparing. *IEEE Transactions on Parallel Distributed Systems 8,* 6, 640–657.

TIAN, L., FENG, D., JIANG, H., ZHOU, K., ZENG, L., CHEN, J., WANG, Z., AND SONG, Z. 2007. PRO: A popularity-based multi-threaded reconstruction optimization for RAID-structured storage systems. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'05)*. USENIX Association, Berkeley, CA, USA, 301–314.

VAN LINT, J., WILSON, R. M., AND HALE, J. K. 1993. *A Course in Combinatorics*. Cambridge University Press, Cambridge, MA.

WIKIPEDIA. 2010. DDR2 SDRAM. http://en.wikipedia.org/wiki/DDR2_SDRAM.

WU, S., JIANG, H., FENG, D., TIAN, L., AND MAO, B. 2009. Workout: I/O workload outsourcing for boosting RAID reconstruction performance. In *Proccedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09)*. USENIX Association, Berkeley, CA, USA, 239–252.

XIANG, L., XU, Y., LUI, J. C., AND CHANG, Q. 2010. Optimal recovery of single disk failure in RDP code storage systems. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'10)*. ACM, New York, NY, USA, 119–130.

XIN, Q., MILLER, E. L., SCHWARZ, T., LONG, D. D. E., BRANDT, S. A., AND LITWIN, W. 2003. Reliability mechanisms for very large storage systems. In *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'03)*. IEEE, Washington, DC, USA, 146–156.

XIN, Q., MILLER, E. L., AND SCHWARZ, T. J. E. 2004. Evaluation of distributed recovery in large-scale storage systems. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*. IEEE, Washington, DC, USA, 172–181.

XU, L. AND BRUCK, J. 1999. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory 45,* 1, 272–276.

Received October 2010; revised March 2011; accepted May 2011