Optimal Recovery of Single Disk Failure in RDP Code Storage Systems

Liping Xiang[†] Yinlong Xu[†] John C.S. Lui[*] Qian Chang[†]

[†]School of Computer Science and Technology
University of Science and Technology of China
ylxu@ustc.edu.cn
{xlping, qchang}@mail.ustc.edu.cn

[*]Department of Computer Science and Engineering
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

ABSTRACT

Modern storage systems use thousands of inexpensive disks to meet the storage requirement of applications. To enhance the data availability, some form of redundancy is used. For example, conventional RAID-5 systems provide data availability for single disk failure only, while recent advanced coding techniques such as row-diagonal parity (RDP) can provide data availability with up to two disk failures. To reduce the probability of data unavailability, whenever a single disk fails, disk recovery (or rebuild) will be carried out. We show that conventional recovery scheme of RDP code for a single disk failure is inefficient and suboptimal. In this paper, we propose an optimal and efficient disk recovery scheme, Row-Diagonal Optimal Recovery (RDOR), for single disk failure of RDP code that has the following properties: (1) it is read optimal in the sense that it issues the smallest number of disk reads to recover the failed disk; (2) it has the load balancing property that all surviving disks will be subjected to the same amount of additional workload in rebuilding the failed disk. We carefully explore the design state space and theoretically show the optimality of RDOR. We carry out performance evaluation to quantify the merits of RDOR on some widely used disks.

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

Keywords

RAID recovery, RDP code, disk failure, recovery algorithm

1. INTRODUCTION

The tremendous growth of information makes designing a massive data storage system more challenging nowadays. As pointed out by authors in [20], there were around 5 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 huge demand for large and cost-effective storage systems, which use thousands or even hundreds of thousands of inexpensive disks to preserve large volumes of data [17, 16, 6].

The continuing increase of system size and the usage of inexpensive but less reliable components for economical consideration make component failures (e.g., disk failure) more common, and this has a huge impact on the reliability and data availability of large scale storage systems [27, 25]. However, a fundamental requirement of building large storage systems is to make sure information is reliable and available even under the presence of component failure. High reliability and data availability can be achieved by maintaining a sufficient amount of redundancy in the storage system. For example, in recent years, several erasure codes [26, 15, 7, 14, 8], such as row-diagonal parity (RDP) [5], EVENODD [2], and X-code [36] were proposed to protect data against more than one disk failure and still can maintain higher level reliability as compared to the conventional RAID-5 systems.

When the number of failed disks is more than the erasure capability, information will be lost and data will be unavailable. Thus, to maintain a high system reliability level, a key approach is to repair and recover a failed component as quickly as possible [1]. 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 [5]. 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 translates to lengthening the *window of vulnerability* (WOV) and increases the probability of data loss [34]. Thus, reducing the disk reads will accelerate the recovery process and improve system reliability.

Another fact which should be taken into consideration is that most of the storage systems have an online failure recovery mode [10], in which the system still provides service to the users during the recovery period. However, multiple reads from disks for recovery consume I/O bandwidth of the system and influence the system service performance

[12]. Therefore, by reducing the number of disk reads, one can also improve the performance to users during the online failure recovery mode. Additional benefits for reducing disk reads include conserving disk energy and relieving the communication load of the entire storage system.

Systems based on double fault tolerant array 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). Therefore, one should focus on designing an efficient recovery scheme of a single failed disk for storage systems that can tolerate multiple disk failures. However, most commonly adopted disk recovery strategies only use single parity for single failure recovery [2, 5, 36], and there is no research which focuses on designing a fast and efficient recovery scheme for single failure recovery of double fault tolerant array codes.

Recent studies show that memory reads are about 100 times faster than disk reads. 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 [32]. 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 work is on designing an optimal recovery scheme for single disk failure in a storage system that uses double fault tolerant array codes. We consider the single failure recovery problem for storage systems which use the RDP codes. We show that one can exploit both parity disks to reduce the number of disk reads for the recovery of single disk failure and propose a hybrid recovery scheme RDOR. Using RDOR, the disk reads for recovery can be reduced by approximately 25% compared with the conventional recovery strategies. The main contributions of this paper are:

1. We propose an optimal recovery scheme for single disk failure which can reduce disk reads and therefore improves system recovery performance.

2. We derive the lower bounds on the total amount of disk reads needed for any single failure recovery of RDP code storage systems.

3. RDOR has the load-balancing property that all surviving disks will experience the same amount of workload to recover the failed disk, and the recovery scheme matches the lower bound on the number of disk reads.

This paper is organized as follows. We will briefly introduce the concept of storage system and the construction mechanisms of RDP code in Section 2. Section 3 presents the main idea of our recovery scheme using a simple example. In Section 4, we theoretically analyze the lower bounds of the disk reads and propose the RDOR scheme which is read-optimal and load balanced. In Section 5, we present the experimental results to show the advantage of RDOR. Section 6 introduces the related works on disk recovery. Conclusion and discussion of future work are presented in Section 7.

2. BACKGROUND ON RDP STORAGE SYSTEMS

RDP code storage system can be realized by multiple inexpensive disks. Disks of the same size which are arranged in an array constitute a disk array storage system. Among these disks, some of them are information disks and the rest are parity disks.

Erasure codes are defined by parity symbol construction mechanisms to protect information from disk failure. When a disk fails in the system, 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.

RDP code is one of the most important double fault tolerant erasure codes which achieves optimality both in computation efficiency and I/O efficiency. It is also named as RAID-DP in NetApp's commercial products [19]. 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 [26], so all the disks will get the same workload.

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, 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 1 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. Symbols $\{d_{0,4}$, $d_{1,3}$, $d_{2,2}$, $d_{3,1}\}$ consist of the missing diagonal. Refer to [5] for more details.

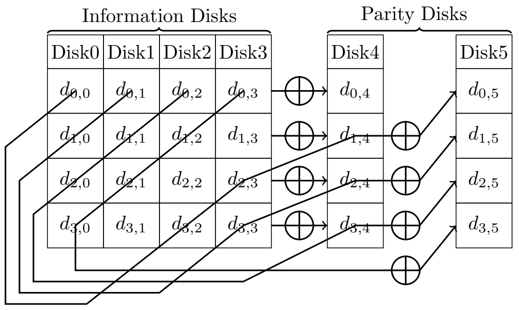
Information Disks
Parity Disks
Disk0
Disk1
Disk2
Disk3
Disk4
Disk5
$d_{0,0}$ $d_{0,1}$ $d_{0,2}$ $d_{0,3}$ $d_{0,4}$ $d_{0,5}$
$d_{1,0}$ $d_{1,1}$ $d_{1,2}$ $d_{1,3}$ $d_{1,4}$ $d_{1,5}$
$d_{2,0}$ $d_{2,1}$ $d_{2,2}$ $d_{2,3}$ $d_{2,4}$ $d_{2,5}$
$d_{3,0}$ $d_{3,1}$ $d_{3,2}$ $d_{3,3}$ $d_{3,4}$ $d_{3,5}$

Figure 1: Storage system with RDP code of $p = 5$.

2.1 Conventional Recovery Scheme for RDP Storage Systems

In this subsection, we will discuss the "*conventional*" approach of recovering a failed disk in the RDP storage systems.

For RDP, 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. For example, if Disk 0 fails, one can read $\{d_{0,1}, d_{0,2}, d_{0,3}, d_{0,4}\}$ to recover $d_{0,0}$.

If the failed disk is a parity disk, recovery is equivalent to the corresponding encoding. For example, if Disk 4 fails, one can XOR-sum $\{d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}\}$ to reconstruct $d_{0,4}$. If Disk 5 fails, one can XOR-sum $\{d_{0,3}, d_{1,2}, d_{2,1}, d_{3,0}\}$ to reconstruct $d_{3,5}$.

It is important to note that the conventional strategy for single failure recovery of RDP code only uses a single parity column. However, all data blocks are protected by two different parity groups. In the following, we will present the RDOR scheme which uses both parities. The interesting properties of RDOR are that (1) it reduces the number of disk read operations; (2) it maintains load balancing property among all surviving disks.

3. HYBRID RECOVERY APPROACH FOR SINGLE FAILURE

In this section, we will first introduce an example to illustrate the idea of RDOR for single failure recovery, and then we formally present the recovery process.

Example 1: Let us consider a storage system in Figure 2 which uses an RDP code with $p = 7$. Assume that Disk 0 (column 0) failed. The six information symbols $d_{i,0}$ $(0 \le i \le 5)$ are to be recovered. With the conventional recovery strategy, $d_{i,0}$ $(0 \le i \le 5)$ are all recovered from row parity (or Disk 6). Therefore, 36 symbols need to be read from disks for the recovery. As illustrated in Figure 2, the erasure symbols are labeled with "×", and the symbols used for recovery are labeled with "◯".

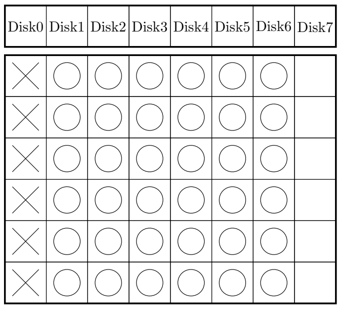

Figure 2: Conventional scheme to recover Disk 0.

Minimizing Disk Reads:[1] Because an information symbol can be recovered from either row parity or diagonal parity, the conventional recovery strategy is only one of many possible schemes. Suppose that in Example 1, if the $d_{0,0}$, $d_{1,0}$, $d_{2,0}$ 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 3 illustrates this case, where the symbols labeled with "◯" are read for the recovery through row parity and the symbols labeled with "□" are read for the recovery through diagonal parity. If an overlapping symbol is stored

[1]The main idea of using double parities for single failure recovery to reduce the number of disk reads can be generalized to other double fault tolerant array codes such as EVENODD and X-code.

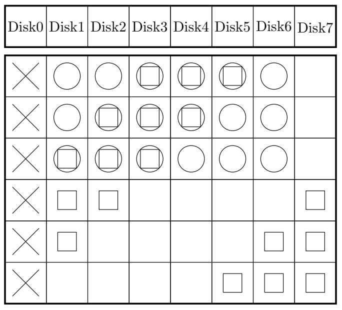

Figure 3: Reducing the number of disk reads for recovering Disk 0.

in memory after it is read from a disk at the first time, it can be read from memory for the 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. Since this recovery scheme reduces disk reads by using memory read instead of disk read, the recovery process will be sped up and the communication load of the storage system will be reduced. The main idea of using previously read symbols to recover data so as to reduce I/O operation is appealing. Theoretically, we also like to answer the following questions:

- What is the lower bound of disk reads for single 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 3 is carried out in each stripe, nearly all the data from Disk 1 need to be read for the recovery of the failed Disk 0, while only half from Disk 4.

Consider another hybrid recovery scheme in Example 1, 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 4 depicts the symbols being read for the recovery with this scheme. In Figure 4, 27 symbols are read from disks for the recovery, the same as that in Figure 3. However, in Figure 4, 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 are *perfectly balanced* among all surviving disks. One interesting question we like to address is:

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

In the following sections, We will analyze the minimum disk reads and balanced disk reads for the recovery. We will

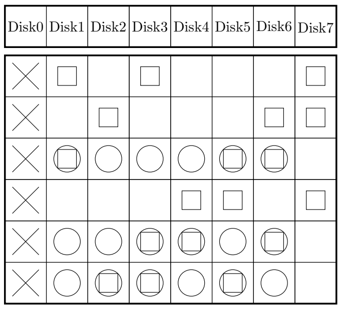

Figure 4: Load balancing in recovering Disk 0.

also design recovery schemes for single disk failure recovery of RDP code which match the minimum disk reads and balanced disk read simultaneously.

4. THE RECOVERY OF SINGLE DISK FAILURE

4.1 Lower Bounds of Disk Reads for Single Failure Recovery

In this section, we derive the lower bound of the number of symbols to be read from the disks for any single column erasure recovery under the RDP code. Because an erasure symbol can be recovered either from the row parity or the 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 1.

1) Define $R_i = \{d_{i,r} | 0 \leq r \leq p-1\}$ as the i-th row parity set, $0 \leq i \leq p-2$;

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

From Definition 1, $|R_i| = p$ 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 1.

LEMMA 1. *Given a symbol d in an RDP disk array,*

1) if $d \in R_i$, d can be recovered by XORing 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 by 1) and/or 2)

Remark: Erasure symbols can only be recovered from their parity sets, but the surviving symbols 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 bound of disk reads in the following Theorem 1 can not be further reduced by generating surviving symbols from other parity sets.

To illustrate, consider the example in Figure 1. 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 2.

LEMMA 2. *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 $d_{i,k} \in D_{<i+k>_p}$. Symbol $d_{i,k}$ can be recovered from either R_i or $D_{<i+k>_p}$. [2]

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 $d_{i,p} \in D_i$, $d_{i,p}$ can only be recovered by D_i.

Remark: Lemma 2 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 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 5 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 5, 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

[2]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

Figure 5: RDP coding with R_i, D_j representation.

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 all together $2 \times 2 \times 2 \times 1 = 8$ recovery combinations to reconstruct symbols in Disk 1. The conventional recovery scheme, (R_0, R_1, R_2, R_3), is just one such possible combination. Since Disk p (or column p) is the diagonal parity column, which can only be recovered by diagonal parity set, 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 which has the highest number of overlapping symbols. We have the following Lemma 3 stating the property of overlapping symbols between two parity sets.

Lemma 3.

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 strategy for single disk failure only uses row parity sets for the recovery. Lemma 3 shows that there is no overlapping symbol during the recovery process with the conventional recovery strategy. Therefore, one cannot reduce the read operations using the conventional recovery scheme. Also according to Lemma 3, 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 1 gives a lower bound of disk reads for any single erasure column recovery.

Theorem 1.

1) *For any erasure column k ($k \neq p$), the minimum number of disk reads for single disk failure recovery 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 $k\,(0 \le k \le p-1)$. We need to recover all the $p-1$ symbols $d_{i,k} (0 \le i \le p-2)$ in column k. Erasure symbol $d_{i,k}$ can only be recovered by the parity sets to which it belongs. From Lemma 3, we know 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 is equal 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 late 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 1 concludes. $\square$

From Theorem 1, 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 read operations on individual disk of some choices are the same (or almost the same), while that of other choices varies greatly. In the next subsection, we will study how to balance the disk reads.

4.2 Balancing Number of Disk Reads

Let the erasure column be column $k\,(0 \le k \le p-1)$. 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 balanced read 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 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.

In the following, we will only present the analysis for the case of $p \equiv 3 \pmod 4$. The analysis of the case of $p \equiv 1 \pmod 4$ is similar. So we will only list the conclusions of the case of $p \equiv 1 \pmod 4$ 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. Given a recovery sequence $\{x_i\}_{0 \le i \le p-1}$, the following Lemma 4 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 2.

LEMMA 4. *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 $c_{i,j} = 1\,(0 \le i, j \le p-1)$.

If $x_i = 0$, symbol $d_{i,k}$ is recovered by row parity set R_i, $d_{i,j}$ is read to recovery $d_{i,k}$ 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 $j\,(0 \le j \le p-1, j \ne k)$ for the recovery of Disk k is

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

Therefore, Lemma 4 concludes. □

The following Theorem 2 provides a necessary and sufficient condition for a recovery sequence to be both read-optimal and balanced.

THEOREM 2. *$\{x_i\}_{0 \le i \le p-1}$ is a read-optimal and balanced recovery sequence for Disk $k\,(0 \le k \le p-1)$, if the following three conditions hold:*

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

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 1, that condition 1) and condition 2) hold is equivalent to that $\{x_i\}_{0 \le i \le p-1}$ is a read-optimal sequence.

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

As any read-optimal recovery will averagely read $(3p-5)/4$ symbols 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, The number of symbols which don't need to be read from Disk $j\,(0 \le j \le p-1, j \ne k)$ is

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

Therefore

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

Because the recovery is read optimal, from Theorem 1, $(p-1)/2$ symbols in column k are recovered from their diagonal parity sets. So

$$\sum_{i=0}^{p-1} x_i = (p-1)/2.$$

From Equation (1) and (2),

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

Let $t = <j-k>_p$, then $<i+j-k>_p = <i+t>_p$. Since $0 \le j \le p-1$, $j \ne k$ and $0 \le k \le p-1$, $1 \le t \le p-1$. Equation (3) means that condition 3) holds.

One can prove in the similar way 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 2 holds. □

In the following of this subsection, we use difference set [31] to present the read-optimal and balanced recovery scheme. Firstly, we introduce some definitions and properties of multiset and difference set.

Definition 2. A multiset M is a generalization of a set, in which there maybe multiple instances of an element. M is denoted as

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

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

Definition 3. Given a recovery sequence $\{x_i\}_{0 \leq i \leq p-1}$. Define $A_x = \{i | x_i = 1,\ 0 \leq i \leq p-1\}$ and the multiset $M_{A_x} = \{a_1 - a_2 | a_1, a_2 \in A_x, a_1 \neq a_2\}$.

LEMMA 5. *Given a recovery sequence $\{x_i\}_{0 \leq i \leq p-1}$. Then*

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

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

PROOF. Given t, $1 \leq t \leq p-1$. 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 $0 \leq i_{t_u} \leq p-1$ and $i_{t_u} \neq i_{t_v}$ for $t_u \neq t_v$, such that

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

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

$$(i_{t_u} + t) - i_{t_u} = t \in M_{A_x}.$$

So for any t, $1 \leq t \leq p-1$, the multiplicity of t in M_{A_x} is $(p-3)/4$. Lemma 5 concludes. $\square$

Consider the example in Figure 4, (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 $\{x_i\}_{0 \leq i \leq 6} = \{1101000\}$, and $A_x = \{0, 1, 3\}$. $M_{A_x} = \{a_1 - a_2 | a_1, a_2 \in A_x, a_1 \neq a_2\} = \{1, 2, 3, 4, 5, 6\}$, which satisfies that $\forall t$ $(1 \leq t \leq 6)$, t has a multiplicity of $(p-3)/4 = 1$ in the multiset M_{A_x}.

Definition 4. 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 $M_S = \{s_1 - s_2 \,|\, s_1,\ s_2 \in S,\ s_1 \neq s_2\}$.

Definition 5. 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 3. *[31] 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 $S_p = \{i^2 \bmod p | 1 \leq i \leq p-1\} = \{i^2 \bmod p | 1 \leq i \leq (p-1)/2\}$. 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. *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 in F_p, then*

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

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 2, we need to prove that $\{x_i\}_{0 \leq i \leq p-1}$ satisfies the three conditions of Theorem 2.

(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 2.

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

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

$$\begin{aligned} p \notin S'_p &\Rightarrow p - (k+1) \notin S'_p - (k+1) \\ &\Rightarrow p - (k+1) \notin A \Rightarrow x_{p-1-k} = 0 \end{aligned}$$

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

(3) According to Theorem 3, 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 $\forall t$ $(1 \leq t \leq p-1)$, t has a multiplicity of $(p-3)/4$ in the multiset $M_{S'_p} = \{s_1 - s_2 | s_1, s_2 \in S'_p,\ s_1 \neq s_2\}$.

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

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

for any t, $1 \leq t \leq p-1$. Thus, $\{x_i\}_{0 \leq i \leq p-1}$ satisfies condition 3) of Theorem 2.

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

From Theorem 4, 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 in F_p.

2) If $k \in S_p$, let A be $S'_p - (k+1)$; if $k \in S'_p$, let A be $S_p - (k+1)$.

3) For the failed Disk k, the read-optimal and balanced recovery scheme is 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 $k\,(k \neq p)$, when $p \equiv 3 \pmod 4$. Assume that $p = 7$, then $S_7 = \{i^2 \bmod 7 | 1 \leq$

$i \le (7-1)/2\} = \{1^2 \bmod 7, 2^2 \bmod 7, 3^2 \bmod 7\} = \{1,2,4\}$, 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, $A = S_p' - (k+1) = S_p' - (1+1) = \{(s-2) \bmod 7 | s \in S_p'\} = \{1,3,4\}$ corresponds to a recovery sequence $\{x_i\}_{0\le i\le 6} = \{0101100\}$. Thus, the read-optimal and balanced recovery scheme is 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, $A = S_p - (k+1) = S_p - (3+1) = \{(s-4) \bmod 7 | s \in S_p\} = \{0,4,5\}$ corresponds to a recovery sequence $\{x_i\}_{0\le i\le 6} = \{1000110\}$. Thus, the read-optimal and balanced recovery scheme is 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 5 to Theorem 7 give a read-optimal and balanced recovery scheme for any single disk failure with $p \equiv 1 \pmod 4$. They are similar to Theorem 2 to Theorem 4 of the case of $p \equiv 3 \pmod 4$ respectively. Therefore they can be proved in the similar way and we omit them for brevity.

THEOREM 5. *Let p be a prime number with $p \equiv 1 \pmod 4$. $\{x_i\}_{0\le i\le p-1}$ 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, if the following three conditions hold:*

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 $\sum_{i=0}^{p-1} x_i x_{<i+t>_p} = (p-5)/4$.

Definition 6. 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 $\{s_1 - s_2 | s_1, s_2 \in S, s_1 \neq s_2\}$.

THEOREM 6. *[21] 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 in 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 7. *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' in 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 $k\,(k \neq p)$ in an RDP coded disk array.

4.3 The RDOR Recovery Scheme for Single Disk Failure

In the last subsection, we presented a recovery combination which is read-optimal and balanced for any erasure column $k\,(k \neq p)$. In the following, we propose the RDOR scheme, which is a read-optimal, balanced recovery scheme 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 and Theorem 7. For $0 \le i \le p-1$, 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 intersection of the selected row parity sets and diagonal parity sets for the recovery.

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

In summary, the following is the RDOR scheme:

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 0.

2. Recover the $(p-1)/2$ symbols $d_{j,k}$ $(0 \le j \le p-1,\ j \notin A)$.
   (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.

We compare RDOR with the conventional recovery scheme in the following three aspects.

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

Table 1: 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

5. PERFORMANCE EVALUATION

To evaluate the performance of RDOR, we conduct experiments which compare RDOR scheme with conventional scheme in terms of total recovery time and individual disk access time.

5.1 Methodology

We conduct the experiments by using a widely accepted disk simulator, DiskSim [3]. The simulated disk array subsystem is composed of $p + 1$ disks including two redundant disks. The logical layout is rotated between different parity stripes as in common RAID-6 implementation. 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 scheme with conventional scheme under the same circumstances.

For the recovery of a single failed disk, RDOR and the conventional scheme are implemented into the simulation environment. In both schemes, during recovering each stripe, 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. 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.

Note that, the strip size and the disk array size affect the system recovery performance. Our experiments are conducted with different number of disks and different strip sizes. The simulated disk's parameters are extracted from a modern enterprise hard disk [28], and Table 1 lists all the parameters used in this section.

5.2 Performance with Different Strip Sizes

To evaluate the recovery performance under different strip sizes, we vary the strip size from 32KB to 1024KB with fixed number of disks, $p + 1 = 8$, 14, and 20 respectively.

Disk access time. RDOR improves the read access efficiency by reducing the number of disk reads in each surviving disk. Figure 6 shows that the average disk read access time of RDOR is reduced by 15.16% $\sim$ 22.60% compared with the conventional scheme. However, the ratio of disk access time doesn't show a stable tendency with strip size varying from 32KB to 1024KB.

RDOR reduces disk reads but changes 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

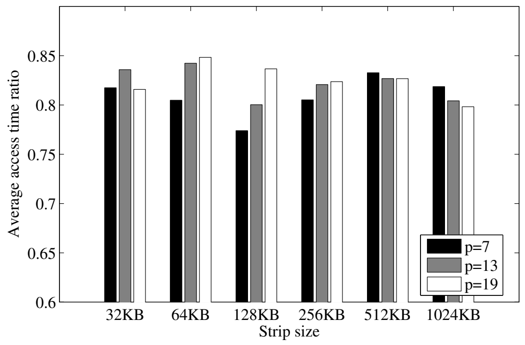

Figure 6: Average disk access time ratio of RDOR to conventional scheme with different strip sizes when $p = 7$, 13, and 19.

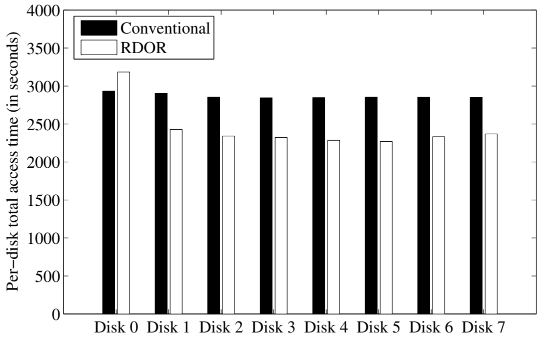

Figure 7: Disk access time of RDOR and conventional scheme of each disk when $p = 7$ and strip size is 32KB. Disk 0 is the failed and thereafter replaced disk.

reason that the access time improvements shown in Figure 6 are less than the theoretical value, approximately 25%.

Moreover, the sequentiality of reads on individual surviving disk in RDOR is not the same. As in Figure 4, 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 7.

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

Total recovery time. Figure 8 shows the total recovery time ratio of RDOR to conventional scheme, from which we can see that RDOR constantly outperforms conventional scheme as strip size varies from 32KB to 1024KB. For example, when $p = 7$, the total recovery time of RDOR is reduced by 5.72% $\sim$ 12.60%. As shown in Figure 8, the improvements of recovery time decrease as strip size varies

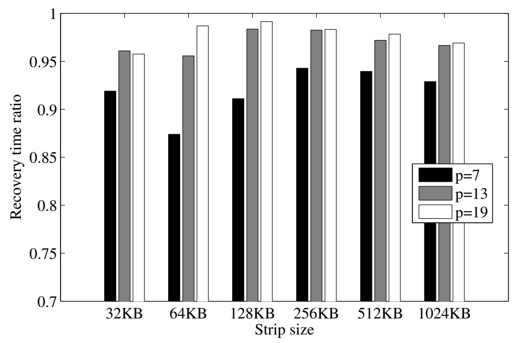

Figure 8: Recovery time ratio of RDOR to conventional scheme with different strip sizes when $p = 7$, 13, and 19.

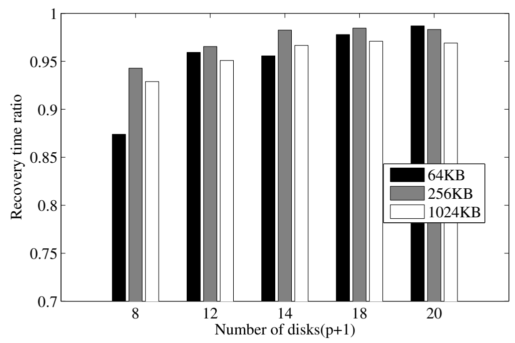

Figure 10: Recovery time ratio of RDOR to conventional scheme with different number of disks when strip size is 64KB, 256KB, and 1024KB.

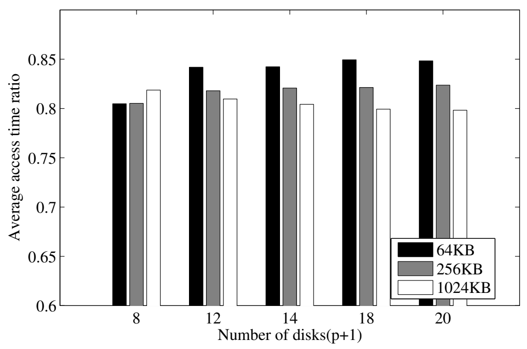

Figure 9: Average disk access time ratio of RDOR to conventional scheme with different number of disks when strip size is 64KB, 256KB, and 1024KB.

from 32KB to 128KB first and then increase as strip size varies from 128KB to 1024KB.

However, in Figure 8 the total recovery time ratios are mostly larger than 0.9. Although RDOR 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 is slower than read on surviving disk, which becomes another bottleneck. Thus, the reduction on disk reads cannot be translated into equivalent saving of recovery time.

In the case of an RDP 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 of RDOR is expected to be better in online recovery than off-line.

5.3 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 disks from 8 to 20 with fixed strip sizes of 64KB, 256KB, and 1024KB respectively.

Disk access time. Figure 9 shows the disk access time ratios are almost the same with different number of disks when the strip size is fixed to 64KB, 256KB, and 1024KB respectively, which implies RDOR can be implemented in fairly large storage systems to improve the read access efficiency. With strip size of 1024KB, the disk access time of RDOR is reduced by 18.13% $\sim$ 20.17%.

Total recovery time. Figure 10 shows the total recovery time ratio increases as the number of disks increases. With strip 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. Since the access time of reading in a stripe on individual surviving disk with RDOR may be different and the write requests to the spare disk should wait until the data has been reconstructed, the recovery process of RDOR may cost more time in waiting and synchronizing with large number of disks. Thus, RDOR provides comparatively smaller improvements in recovery time with large number of disks.

The experimental results can be generalized as follows.

1) By using RDOR, the access time of an individual disk for recovery is reduced by 15.16% $\sim$ 22.60% compared with conventional scheme. Less disk access time indicates that the disk array system can further accelerate the recovery process in online recovery scenario.

2) RDOR can constantly reduce the total recovery time compared with the conventional scheme with strip size varies from 32KB to 1024KB, which indicates that RDOR scales well as strip size increases.

6. 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 [11, 13, 35] address the problem by trading storage capacity for improved failure recovery performance. Menon and Mattson [22] showed that instead of using a dedicated spare disk, there are many advantages to distribute the spare space across the disk array. In such an array, the recovery writes are distributed as well as the recovery reads. Clustered RAID [24], 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 the recovery parallelism, the Disk-Oriented Reconstruction (DOR)[13] 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 [18], 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 [23, 29]. To relieve the interference of foreground workload, Wu et.al [33] 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 [30] 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 [9] 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 [4] to further improve space and time efficiency during recovery.

7. CONCLUSION

This paper proposes an optimal recovery scheme RDOR for single disk failure of RDP code storage systems. The RDOR scheme uses both row parity and diagonal parity for data reconstruction, and also minimizes the number of disk reads from surviving disks to reduce the recovery time. We derive the lower bound of disk reads for the recovery and design recovery schemes which match the lower bound of disk reads, as well as have a load balancing property on all surviving disks. Experimental results show that our scheme outperforms the conventional recovery scheme in terms of total recovery time and recovery workload on individual surviving disk. It is important to point out that the main idea of our approach can be generalized to other double fault tolerant array codes such as EVENODD or X-code. Our future work includes evaluating the performance on storage systems which are working in online recovery mode.

8. ACKNOWLEDGMENTS

We would like to thank our shepherd, Bianca Schroeder, and the anonymous reviewers for their valuable comments. We would also like to thank Kaiqian Ou, Haotian Zhao, Yubiao Pan and Runhui Li for the helpful discussions during this work. This work is supported by the National Science Foundation of China under Grant No. 60773036.

9. REFERENCES

[1] M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T. Giuli, and P. Bungale. A fresh look at the reliability of long-term digital storage. In *Proceedings of the 2006 EuroSys Conference*, pages 221–234, 2006.

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44(2):192–202, 1995.

[3] J. Bucy, J. Schindler, S. Schlosser, and G. Ganger. The DiskSim simulation environment (v4.0). http://www.pdl.cmu.edu/DiskSim/.

[4] B. Cassidy and J. L. Hafner. Space efficient matrix methods for lost data reconstruction in erasure codes. Technical Report RJ10415, IBM Research, 2007.

[5] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. Row-diagonal parity for double disk failure correction. In *FAST '04: Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, pages 1–14, 2004.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *SOSP '03: Proceedings of the 19th ACM Symposium on Operating systems Principles*, pages 29–43, 2003.

[7] J. L. Hafner. WEAVER codes: highly fault tolerant erasure codes for storage systems. In *FAST '05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 211–224, 2005.

[8] J. L. Hafner. HoVer erasure codes for disk arrays. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 217–226, 2006.

[9] J. L. Hafner, V. Deenadhayalan, K. K. Rao, and J. A. Tomlin. Matrix methods for lost data reconstruction in erasure codes. In *FAST '05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 15–30, 2005.

[10] M. Holland. *On-Line Data Reconstruction in Redundant Disk Arrays*. PhD thesis, Carnegie Mellon University, 1994.

[11] M. Holland and G. A. Gibson. 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*, pages 23–35, 1992.

[12] M. Holland, G. A. Gibson, and D. P. Siewiorek. Fast, on-line failure recovery in redundant disk arrays. In *Proceedings of the 23rd Annual International Symposium on Fault-Tolerant Computing*, pages 422–431, 1993.

[13] M. Holland, G. A. Gibson, and D. P. Siewiorek. Architectures and algorithms for on-line failure recovery in redundant disk arrays. *Distributed and Parallel Databases*, 2(3):295–335, 1994.

[14] C. Huang, M. Chen, and J. Li. Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems. In *NCA '07: 6th IEEE International Symposium on Network Computing Applications*, pages 79–86, 2007.

[15] C. Huang and L. Xu. STAR: an efficient coding scheme for correcting triple storage node failures. In *FAST '05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, 2005.

[16] N. Joukov, A. M. Krishnakumar, C. Patti, A. Rai, S. Satnur, A. Traeger, and E. Zadok. RAIF: Redundant array of independent filesystems. In *MSST '07: Proceedings of 24th IEEE Conference on Mass Storage Systems and Technologies*, pages 199–212, 2007.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. In *ASPLOS '00: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.

[18] J. Y. B. Lee and J. C. S. Lui. Automatic recovery from disk failure in continuous-media servers. *IEEE Transactions on Parallel Distributed Systems*, 13(5):499–515, 2002.

[19] C. Lueth. RAID-DP: Network Appliance implementation of RAID double parity for data protection. Technical Report No. 3298, Network Appliance Inc, 2004.

[20] P. Lyman and H. R. Varian. How much information? http://www.sims.berkeley.edu/how-much-info-2003.

[21] S. L. Ma. A survey of partial difference sets. *Des. Codes Cryptography*, 4(3):221–261, 1994.

[22] J. Menon and D. Mattson. Comparison of sparing alternative for disk arrays. In *Proceedings of the International Symposium on Computer Architecture*, pages 318–329, 1992.

[23] A. Merchant and P. S. Yu. Analytic modeling of clustered raid with mapping based on nearly random permutation. *IEEE Transactions on Computers*, 45(3):367–373, 1996.

[24] R. R. Muntz and J. C. S. Lui. Performance analysis of disk arrays under failure. In *Proceedings of the sixteenth international conference on Very large databases*, pages 162–173, 1990.

[25] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *FAST '07: Proceedings of the 5th conference on USENIX Conference on File and Storage Technologies*, pages 17–28, 2007.

[26] J. S. Plank. The RAID-6 liberation codes. In *FAST '08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–14, 2008.

[27] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *FAST '07: Proceedings of the 5th conference on USENIX Conference on File and Storage Technologies*, pages 1–16, 2007.

[28] Seagate Inc. Cheetah® 15K.5 Fibre Channel 146-GB Hard Drive ST3146855FC Product Manual.

[29] A. Thomasian and J. Menon. RAID-5 performance with distributed sparing. *IEEE Transactions on Parallel Distributed Systems*, 8(6):640–657, 1997.

[30] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang, and Z. Song. PRO: a popularity-based multi-threaded reconstruction optimization for RAID-structured storage systems. In *FAST '07: Proceedings of the 5th USENIX conference on File and Storage Technologies*, 2007.

[31] J. van Lint, R. M. Wilson, and J. K. Hale. *A Course in Combinatorics*. Cambridge University Press, Cambridge, MA, 1993.

[32] Wikipedia. DDR2 SDRAM — Wikipedia, the free encyclopedia, 2010.

[33] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao. Workout: I/O workload outsourcing for boosting raid reconstruction performance. In *FAST '09: Proccedings of the 7th conference on File and storage technologies*, pages 239–252, 2009.

[34] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. Reliability mechanisms for very large storage systems. In *MSST '03: Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 146–156, 2003.

[35] Q. Xin, E. L. Miller, and T. J. E. Schwarz. Evaluation of distributed recovery in large-scale storage systems. In *HPDC '04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pages 172–181, 2004.

[36] L. Xu and J. Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, 45(1):272–276, 1999.

APPENDIX

In this section, we will prove that only when all the surviving symbols are read directly from disk, the number of disk reads can achieve the lower bound $3(p-1)^2/4$.

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. Each generated symbol reads $p-3$ additional 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 3, 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, $0 \leq s < 2p-2$, 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 1. Therefore, no matter how we get surviving symbols for recovery, the lower bound of disk reads is $3(p-1)^2/4$.