Triple-Disk Recovery RDP Coding: A Robust Solution for Data Storage in RAID Configurations

Chuhan Wang *

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China.

* Corresponding Author Email: 20211235@bjtu.edu.cn

Abstract. Data-intensive technologies require reliable and efficient storage solutions to ensure the integrity and availability of vast amounts of generated data. Redundant Arrays of Independent Disks (RAID) technology has emerged as a popular choice, offering fault tolerance and improved performance. Among RAID configurations, RAID-6 stands out for its ability to tolerate up to two disk failures. This is achieved through algorithms like Row-Diagonal Parity (RDP) coding, which introduces redundancy and enables data recovery. In this paper, we propose an innovative approach, namely Triple-Disk Recovery RDP (TDR-RDP), to enhance reliability based on RDP coding by accommodating three disk failures. We review the RDP algorithm, introduce the TDR-RDP coding scheme, elaborate on encoding and decoding processes, and address diverse types of triple disk failures along with their recovery strategies. Furthermore, we evaluate and compare the computing efficiency of TDR-RDP with the RDP algorithm, showcasing its practical feasibility. Overall, the TDR-RDP algorithm offers a compelling solution to bolster data storage reliability in RAID configurations, surpassing traditional RDP coding by ensuring resilience against triple disk failures and augmenting data-intensive technologies.

Keywords: RDP coding; Triple-Disk Recovery RDP coding; Computing efficiency.

1. Introduction

In the era of data-intensive technologies, the demand for high-quality data storage has become increasingly critical. As organizations and individuals generate massive amounts of data, ensuring the integrity and availability of this data is of utmost importance, bringing about new challenges in terms of data storage reliability and effectiveness.

RAID technology has proven to be a viable solution in addressing these challenges. RAID configurations allow for the distribution of data across multiple disks, providing advantages such as improved performance, fault tolerance, and cost-effectiveness [1]. By employing specific algorithms that introduce redundancy, RAID systems can tolerate disk failures and ensure data recovery. Among various RAID configurations, RAID-6 has gained widespread adoption due to its enhanced fault tolerance capabilities [2]. Unlike RAID-5, which can only tolerate one disk failure, RAID-6 can withstand the failure of up to two disks by utilizing additional parity information. In the event of two disk failures, the parity information enables the reconstruction of data, ensuring the integrity and availability of stored information.

One of the most commonly used algorithms in RAID-6 is EVENODD coding [3]. It is an efficient encoding procedure that employs exclusive-OR operations and independent parity to provide fault tolerance. By distributing parity information across the disks, EVENODD allows for two disk failures. This algorithm has been widely adopted due to its simplicity and effectiveness in maintaining data integrity.

However, researchers have continuously sought to enhance the reliability and efficiency of RAID-6 systems [4-6]. Building upon the foundations of EVENODD coding, the Row-Diagonal Parity (RDP) algorithm has emerged as an optimization approach, which introduces improvements in both reliability and efficiency [7]. By leveraging the properties of EVENODD coding, RDP offers comparable reliability while reducing encoding and decoding complexity, resulting in enhanced efficiency.

In light of the advantages of the RDP algorithm, this paper proposes a novel approach known as Triple-Disk Recovery RDP (TDR-RDP). The algorithm aims to address the limitations of the RDP algorithm by allowing the tolerance of three disk failures. This further strengthens data security and availability, aligning with the increasing requirements for reliable and effective data storage.

The remainder of this paper is structured as follows:

1. We provide a comprehensive review of the RDP coding scheme, along with its performance evaluation.

2. We introduce our TDR-RDP algorithm, presenting the encoding and decoding processes of the coding scheme, and offering a step-by-step breakdown of recovery strategies for different situations of triple disk failure.

3. We evaluate the computing efficiency of TDR-RDP coding, comparing it to RDP coding to demonstrate its feasibility.

2. RDP Coding

2.1. RDP Encoding Process

The RDP algorithm utilizes a simple parity encoding scheme that exclusively employs exclusive-or (XOR) operations. In the RDP coding scheme, 2 redundancies - Parity 1 and Parity 2 - are added to allow for any double disk failures. Parity 1 serves as the first redundancy and stores the row parity checksum by performing XOR operations on the source data within the same row. The second column, Parity 2, stores the diagonal parity checksum data. In Fig 1, data blocks sharing the same operation in Parity 2's computation are depicted with identical colors. Notably, the diagonal in white is not included in diagonal parity checksum computations.

Disk/ Block	1	2	3	4	Parity 1	Parity 2
0	1	0	0	1	0	
1	0	0	1	0	1	
2	1	1	0	1	0	
3	1	0	0	1	1	

Fig 1. The operation method of diagonal parity checksum data in RDP coding

The RDP coding scheme and the EVENODD scheme share similar algorithms [8]. However, the diagonal checksum operation in RDP is simplified. To be more specific, in RDP, there is not an additional diagonal parity checksum participating in Parity 2's computation which is named 'S' in EVENODD; instead, Parity 1 is included in the diagonal calculation. In RDP, the number of stored source disks is one less than that in EVENODD.

2.2. RDP Decoding Process

During the decoding process, RDP coding employs a consistent logical chain to deduce all the data. Specifically, when any two source disks fail, there will always be two diagonals with only one missing data, which can be recovered using diagonal parity. Subsequently, the entire disk can be recovered step by step by utilizing row parity and diagonal parity. A clear example of the recovery process for all erasures is depicted in Fig 2, with arrows indicating each step.

Disk/ Block	1	2	3	4	Parity 1	Parity 2
0	1		0		0	1
1	0		1		1	1
2	1		0		0	0
3	1		0		1	0

Fig 2. An example of the decoding process in RDP coding

2.3. Assessment of RDP Coding Performance

RDP coding, like EVENODD coding, ensures high data reliability by allowing any two disk failures. However, RDP coding offers several advantages over EVENODD coding.

Firstly, RDP eliminates the need to calculate 'S' in Parity 2's computation and thus reduces the access frequency of Parity 2, which helps decrease the likelihood of disk failure and enhance the robustness of RAID-6. Secondly, RDP demonstrates better efficiency in terms of I/O operations. This is because RDP stores data in clear and avoids reading and writing to Parity 2 frequently when adjusting data in blocks within the 'S' sequence. Additionally, RDP exhibits better computing efficiency due to fewer XOR operation times per block during encoding and decoding [9].

In summary, RDP coding outperforms EVENODD coding in terms of reliability and efficiency. Therefore, extending RDP to a Triple-Disk Recovery RDP coding scheme (TDR-RDP) holds significant promise.

3. Algorithm of Triple-Disk Recovery RDP Coding

3.1. Triple-Disk Recovery RDP Encoding Process

RDP coding is an extension of RDP coding which adds an additional redundancy to allow for any triple disk failures. The first redundancy (Parity 1) is achieved through row parity operations; the second redundancy (Parity 2) is achieved through diagonal parity operations with blocks in the same color as shown in Fig 1. Finally, the third redundancy (Parity 3) is achieved through back-diagonal parity operations as shown in Fig 3. Particularly, the operations in Parity 3 are symmetrical to that of Parity 2.

Disk/ Block	1	2	3	4	Parity 1	Parity 2	Parity 3
0	1	0	0	1	0		
1	0	0	1	0	1		
2	1	1	0	1	0		
3	1	0	0	1	1		

Fig 3. The operation method of back-diagonal parity checksum data in Triple-Disk Recovery RDP coding

3.2. Triple-Disk Recovery RDP Decoding Process

The situation of triple disk failures in a RAID-6 system can be categorized into 4 cases:
Case 1: 3 source disk failures.
Case 2: 2 source disk failures and 1 parity disk failure.
Case 3: 1 source disk failure and 2 parity disk failures.
Case 4: 3 parity disk failures.
The blocks within each disk are named as shown in Table 1.

Table 1. Names of blocks within Disk 1-4 and Parity 1-3

Disk/ Block	1	2	3	4	Parity 1	Parity 2	Parity 3
0	d0,1	d0,2	d0,3	d0,4	p0,1	p0,2	p0,3
1	d1,1	d1,2	d1,3	d1,4	p1,1	p1,2	p1,3
2	d2,1	d2,2	d2,3	d2,4	p2,1	p2,2	p2,3
3	d3,1	d3,2	d3,3	d3,4	p3,1	p3,2	p3,3

The paper will discuss each case in detail and specify the decoding process mathematically.

3.2.1 Decoding Process for Case 1

In Case 1, there is only one intact source disk; the primary aim is to recover from one of the failed source disks by employing the intact source disk and 3 parity disks. This contributes to the transformation of triple disk failures into double disk failures. Then, by using the RDP coding scheme, the remaining 2 failed disks can be recovered, leading to a Triple-Disk Recovery success.

We make use of the mathematical relationships between blocks in different rows and diagonals, trying to figure out some of the erased blocks, as shown in Fig 4.

(1) Suppose that Disk 2 is the only intact source disk.

By summing up the parity values from the six sets of rows and diagonals, we can derive the following equations, as shown in Fig 4 (1).

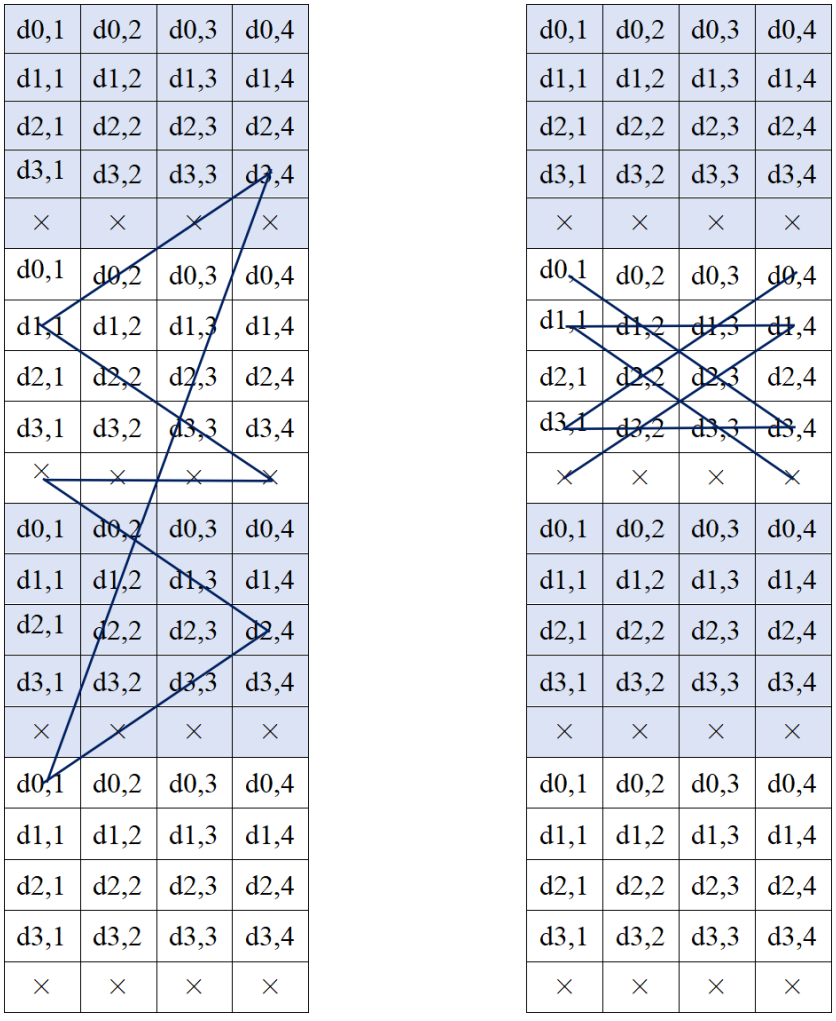

Fig 4 (1). The operations to figure out the mathematical relationships between Disk 2 and Disk 3

Fig 4 (2). The operations to figure out the mathematical relationships between Disk 1 and Disk 4

$$d_{1,1} \oplus d_{0,2} \oplus d_{3,4} = p_{2,1} \oplus p_{1,2}, \quad (1)$$

$$d_{0,1} \oplus d_{3,3} \oplus d_{2,4} = p_{1,1} \oplus p_{0,2}, \quad (2)$$

$$d_{1,1} \oplus d_{2,2} \oplus d_{3,3} = p_{0,1} \oplus p_{1,3}, \quad (3)$$

$$d_{0,2} \oplus d_{1,3} \oplus d_{2,4} = s_2, \quad (4)$$

$$d_{0,1} \oplus d_{1,2} \oplus d_{2,3} \oplus d_{3,4} = p_{0,3}, \quad (5)$$

Where

$$s_2 = d_{0,2} \oplus d_{1,3} \oplus d_{2,4} \oplus p_{3,1} = p_{1,3} \oplus p_{2,3} \oplus p_{3,3} \oplus p_{0,3} \oplus p_{3,1}. \quad (6)$$

By making an XOR sum of equations (1-5), we can get that:

$$d_{1,2} \oplus d_{2,2} \oplus d_{1,3} \oplus d_{2,3} = C_1, \quad (7)$$

where C_1 is a constant, since all the blocks in Parity 1 - 3 are known.

Similarly, by translating the lines downward in Fig. 4 the distance of one block each time, we can get that:

$$d_{2,2} \oplus d_{3,2} \oplus d_{2,3} \oplus d_{3,3} = C_2, \quad (8)$$

$$d_{3,2} \oplus d_{3,3} = C_3, \quad (9)$$

$$d_{0,2} \oplus d_{0,3} = C_4, \quad (10)$$

where C_2, C_3, and C_4 are all constants.

Therefore, it can be calculated that:

$$d_{1,2} \oplus d_{1,3} = C_1 \oplus C_2 \oplus C_3, \quad (11)$$

$$d_{2,2} \oplus d_{2,3} = C_2 \oplus C_3, \quad (12)$$

showing that when Disk 2 is intact, Disk 3 can be instantly recovered. Symmetrically, if Disk 3 is intact, the algorithm allows for direct recovery of Disk 2.

(2) Suppose that Disk 1 is the only intact source disk.

Since the analytical process is the same as the former part, the paper only gives the outcome, according to Fig 4 (2):

$$d_{0,1} \oplus d_{0,4} = Q_1, \quad (13)$$

$$d_{1,1} \oplus d_{1,4} = Q_1 \oplus Q_4, \quad (14)$$

$$d_{2,1} \oplus d_{2,4} = Q_2 \oplus Q_3, \quad (15)$$

$$d_{3,1} \oplus d_{3,4} = Q_2, \quad (16)$$

where Q_1, Q_2, Q_3, and Q_4 are all XOR sums of the blocks in known disks, making them constants.

Therefore, Disk 1 and Disk 4 have the same relationship as Disk 2 and Disk 3.

Thus, any situation of Case 1 can be transformed into a double-disk-failure problem and thus be solved by RDP coding.

3.2.2 Decoding Process for Case 2

(1) If Parity 2 or Parity 3 fails, the situation is equivalent to a double-disk failure situation which can be solved through RDP coding. Every erasure can be finally recovered using a row parity disk (Parity 1) and a diagonal parity disk (Parity 2 or Parity 3).

(2) If Parity 1 fails, the situation is more complex. But the basic idea is still to sum up the parity values across multiple sets of rows and diagonals, thus turning the situation into a double-disk-failure one.

Suggest that Disk 1, Disk 4, and Parity 1 failed. As the blocks in the lines in Fig 5 show:

$$p_{1,1} \oplus p_{3,1} = K_1, \tag{17}$$

where $K_1 = p_{1,2} \oplus p_{3,2} \oplus p_{0,3} \oplus p_{3,3} \oplus d_{2,2} \oplus d_{2,3}$, making K_1 a constant.

Translate the lines downward in Fig. 6 the distance of one block each time, and we can find that:

$$p_{1,1} \oplus p_{2,1} = K_2, \tag{18}$$

$$p_{2,1} \oplus p_{3,1} = K_3, \tag{19}$$

$$p_{3,1} = K_4, \tag{20}$$

where K_2, K_3, and K_4 are all XOR sums of data in known blocks, making them constants.

Therefore, $p_{0,1}$, $p_{1,1}$, $p_{2,1}$, and $p_{3,1}$ can be figured out according to equations (17-20), turning triple disk failures into double disk failures.

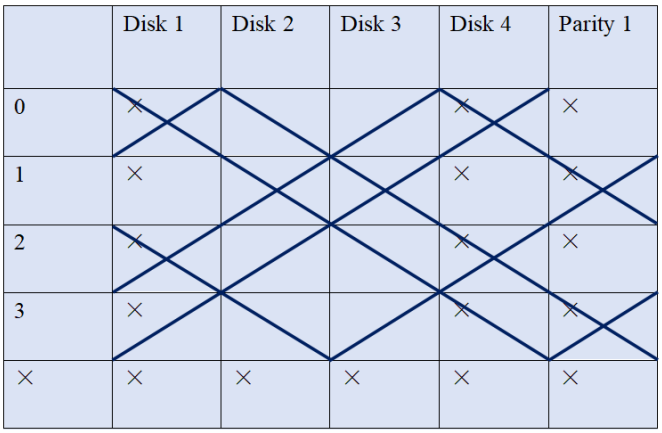

Fig 5. The operations to recover from one of the failed disks among Disk 1, Disk 4, and Parity 1

Similarly, translating the lines towards the right in Fig. 6 the distance of one block each time allows us to deal with each situation in this part.

3.2.3 Decoding Process for Case 3

(1) If Parity 2 and Parity 3 fail at the same time, the situation equals a one-disk-failure one in RAID-5 and can be easily solved.

(2) If one of Parity 2 and Parity 3 fails, the 2 situations are symmetrical.

Suggest that Parity 1, Parity 3, and data disk 1 fail at the same time. Considering the row and diagonal checksum, $d_{0,1}$, $d_{1,1}$, $d_{2,1}$, and $d_{3,1}$ can be easily figured out by making XOR operations between known numbers in intact disks - Disk 2, 3, 4, and Parity 2. Disk 1 recovered; Parity 1 can therefore be finally recovered.

Other situations in Case 3 follow the same process.

3.2.4 Decoding Process for Case 4

This situation is the simplest one since we can directly recover from Parity 1, 2, and 3 by making the row, diagonal, and back-diagonal XOR sums respectively.

4. Computing Efficiency Analysis

The computing efficiency of encoding and decoding is a crucial dimension in evaluating the performance of a coding scheme. Since redundancies in TDR-RDP are gained through XOR operations only, we determine the computational cost as the times of XOR operations to protect one source block on average in encoding or decoding.

4.1. Encoding Efficiency

According to the encoding algorithm introduced in Section 3.1, (n-1) XORs are needed to figure out each row parity checksum block (each block in Parity 1), so (n-1) n XORs are needed in total, where n is the number of source disks. Equally, generating the other 2 disks of redundancy - Parity 2 and 3 - requires (n-1) n XORs each. Thus, 3(n-1) n XORs are required to save n2 source blocks; (3-3/n) XORs are required to save each data block on average.

4.2. Decoding Efficiency

According to the decoding algorithm introduced in Section 3.2, the most complicated but also common triple-disk-failure situation is 3 source disk failures. So, we only consider this situation to simplify the assessment.

Following the decoding steps in Section 3.2.1, we get the total XOR operation times; due to the space limitation, the paper only shows the final result: (3.75n+1) n-8 [10]. So, $[(3.75+1/n)-8/n^2]$ XORs are required for each data block on average.

By comparing the computing efficiency of RDP and TDR-RDP, it is clear that the encoding efficiency of TDR-RDP is exactly 1.5 times that of RDP; in decoding, the XOR operation times for each source block in TDR-RDP stay stable between 3.5 - 4, namely about 2 more XORs than that of RDP, which is acceptable.

A Specific comparison can be seen in Table 2.

Table 2. Comparison of computing efficiency between RDP and TDR - RDP

p=n+1	n	Encoding		Decoding	
		RDP 2-2/n	TDR - RDP 3-3/n	RDP 2-2/n	TDR - RDP $(3.75+1/n)-8/n^2$
5	4	1.5	2.25	1.5	3.5
7	6	1.66	2.5	1.66	3.694
11	10	1.8	2.7	1.8	3.77
13	12	1.83	2.75	1.83	3.778
17	16	1.875	2.8125	1.875	3.78125

5. Conclusion

This paper has presented the TDR-RDP algorithm as an innovative approach to enhance the reliability and efficiency of data storage in RAID configurations. TDR-RDP extends the fault tolerance capabilities to tolerate three disk failures, addressing the growing need for reliable data storage in data-intensive technologies.

The recovery strategies for triple disk failures have been thoroughly explored in this study. TDR-RDP employs advanced coding schemes to introduce redundancy and facilitate data recovery in the event of multiple disk failures. Through a comprehensive evaluation, the effectiveness of TDR-RDP in maintaining data integrity and availability has been demonstrated. Furthermore, the computing efficiency of TDR-RDP has been assessed and compared with the traditional RDP coding. The evaluation results indicate that TDR-RDP achieves comparable computing efficiency with RDP coding. The encoding process requires approximately 1.5 times the XOR operations per block compared to RDP, while the decoding process requires almost 2 additional XOR operations per block. These computational overheads are within acceptable limits and do not significantly impact the overall performance of TDR-RDP.

The practical feasibility of TDR-RDP in real-world data-intensive applications is evident. Its ability to accommodate three disk failures and provide reliable data storage makes it a compelling choice for various environments, including databases, cloud storage, and high-performance computing systems. The enhanced reliability offered by TDR-RDP ensures that critical data remains accessible even in the face of multiple disk failures.

In summary, the TDR-RDP algorithm presented in this paper offers a promising solution for improving the reliability and efficiency of RAID-based data storage. By extending fault tolerance capabilities to tolerate triple disk failures and demonstrating comparable computing efficiency with RDP coding, TDR-RDP proves to be an effective and practical approach. Future research can explore further optimizations and implementations of TDR-RDP in different data-intensive scenarios to maximize its potential benefits.

References

[1] Chen P M, Lee E K, Gibson G A, et al. RAID: High-performance, reliable secondary storage. ACM Computing Surveys (CSUR), 1994, 26(2): 145-185.

[2] Plank J S. The raid-6 liber8tion code. The International Journal of High-Performance Computing Applications, 2009, 23(3): 242-251.

[3] Blaum M, Brady J, Bruck J, et al. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. IEEE Transactions on computers, 1995, 44(2): 192-202.

[4] Zhang G, Li K, Wang J, et al. Accelerate rdp raid-6 scaling by reducing disk i/os and xor operations. IEEE Transactions on Computers, 2013, 64(1): 32-44.

[5] Hou H, Lee P P C. A new construction of EVENODD codes with lower computational complexity. IEEE Communications Letters, 2018, 22(6): 1120-1123.

[6] Fu Y, Shu J. D-Code: An efficient RAID-6 code to optimize I/O loads and read performance. 2015 IEEE International Parallel and Distributed Processing Symposium. IEEE, 2015: 603-612.

[7] Corbett P, English B, Goel A, et al. Row-diagonal parity for double disk failure correction. Proceedings of the 3rd USENIX Conference on File and Storage Technologies. 2004: 1-14.

[8] Hou H, Han Y S, Shum K W, et al. A unified form of EVENODD and RDP codes and their efficient decoding. IEEE Transactions on Communications, 2018, 66(11): 5053-5066.

[9] Chen Y. Efficiency Comparison of Row-Diagonal Parity and EVENODD Encoded Check Disk Repair Algorithms. 2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE). IEEE, 2022: 55-58.

[10] Wan W, Wu Z, Chen Y, et al. A data placement based on toleration on triple failures array codes in RAID. CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-, 2007, 30(10): 1721.