Dean&Francis

ISSN 2959-6157

Advanced Parity Calculation Techniques in EVENODD Coding: A Comparative Analysis of Fault Tolerance and Computational Efficiency

Zijun Wang

Sydney Smart Technology College, Northeastern University, Qinghuangdao, China

*Corresponding author: zijun@ldy.edu.rs

Abstract:

In today's data-driven era, the reliability and security of data storage have attracted much attention. The rapid development of big data, cloud computing and Internet of Things technology has put forward higher requirements for the fault tolerance of data storage. Especially in critical areas such as finance and healthcare, where data integrity and recoverability are critical, data loss or corruption can have catastrophic consequences.Although traditional RAID technology improves data redundancy, its scalability, cost effectiveness and fault tolerance efficiency are limited in the face of big data challenges. Especially in large-scale data centers, RAID is difficult to balance high performance with high fault tolerance.EVENODD coding , as a new fault-tolerant technology, improves recovery efficiency through double parity check. However, the computational complexity of the second parity column (Parity2) is high, which affects the coding efficiency and resource utilization. This research focuses on optimizing the computational method of Parity2 in EVENODD encoding, aiming at improving fault tolerance and reducing complexity, providing technical support for building efficient and reliable data storage system, and ensuring data security and integrity.

Keywords: EVENODD Coding , Fault Tolerance Optimization , Data Storage Security.

1. Introduction

Error correction codes (ECC) are critical to improving the reliability of digital communications and data storage systems. The earliest error-correcting codes, such as Hamming codes [1], laid the foundation for error detection and correction. Hamming codes [1], proposed by Richard Hamming in 1950, are capable of detecting up to two errors and correcting one error, which led to their widespread use in early digital

systems. Subsequently, cyclic redundancy check (CRC) became another important technique, which generates checksums by polynomial division for detecting errors in blocks of data. Due to its high efficiency and reliability, this technology is widely used in network communication protocols and data storage systems.

With the continuous development of technology, more complex error correction codes have been introduced to meet the needs of modern communication systems for high reliability and high efficiency. Low-density parity check (LDPC) codes [3] are one of the important advances. LDPC codes were first proposed by Robert Gallagher in the 1960s and rediscovered in the 1990s. LDPC codes represent parity matrix by sparse bipartitic graph structure and support efficient iterative decoding algorithm, which makes it widely used in satellite communication, wireless network and data storage. In addition to LDPC codes, Reed-Solomon Code [4] is also a widely used error-correcting code. It is a block code based on finite field operation, which can correct multiple continuous errors in data blocks, so it is widely used in optical disc storage, digital television broadcasting and deep space communication to provide powerful error correction ability.

EVENODD coding [5,7] is another important development in error correction technology in RAID-6 systems. RAID-6 is a storage scheme that can cope with the failure of two disks at the same time, and EVENODD coding [5,7] is an efficient parity check method designed for this purpose. The EVENODD encoding, proposed by Blaum, Bruck, and Vardy in 1995, generates two parity columns (Parity1 and Parity2) by performing an XOR operation on a block of data, thereby maintaining data integrity in the event of a two-disk failure. EVENODD encoding not only provides reliable failure protection, but also reduces the computational overhead of parity, making it an effective solution in large-scale storage systems.

2. Literature review

The existing EVENODD encoding method is mainly based on the XOR operation, which is simple and efficient and easy to implement in a variety of hardware and software environments. In EVENODD encoding, data blocks are organized into a matrix, which first generates Parity1 for each row, that is, performs an XOR operation on all data blocks in that row. Next, Parity2 is generated by performing an XOR operation on the diagonal data blocks in the matrix. These two sets of parity blocks provide data recovery capability for the RAID-6 system when two disks fail.

The advantage of EVENODD encoding is its low computational complexity and low storage overhead. However, this encoding approach can encounter challenges when faced with multiple disk failures. The current parity check architecture is designed to deal with the failure of two disks, and the EVENODD encoding may not be sufficiently resilient when three or more disks fail simultaneously. While existing decoding algorithms can handle some cases of multi-disk failure, their computational complexity increases significantly, especially in large-scale storage systems, which can put more pressure on real-time data recovery. As a result, researchers are exploring more flexible parity structures and more efficient decoding algorithms to address the challenges in these complex scenarios. While these improvements are often accompanied by higher computational complexity and additional storage overhead, they have significant implications in terms of improving data reliability.

3. Basis of methods and technical models

3.1 Technical basis of EVENODD encoding

EVENODD coding [5,7] is an advanced fault-tolerant mechanism designed for RAID-6 systems. Its core idea is to process data blocks by clever use of XOR operations to generate two key parity columns - Parity1 and Parity2. These two columns of unique check data are seamlessly added to the end of the data matrix as a double guarantee of data integrity. Even in the extreme case that two disks fail at the same time, the two parity columns can effectively recover the lost data to ensure data reliability and availability.

ISSN 2959-6157

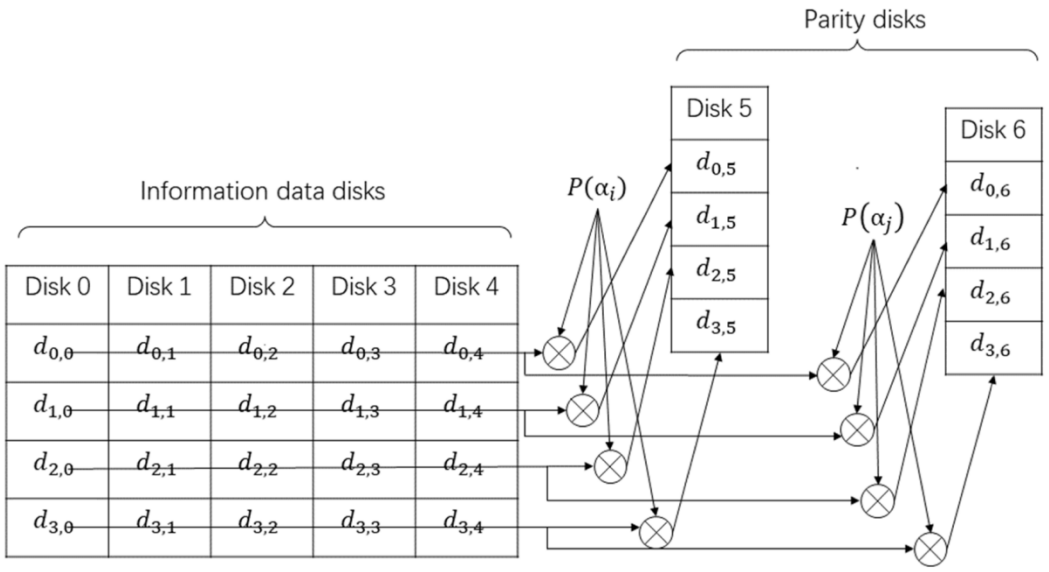

Fig.1 Method of calculating Parity1 and Parity2 based on EVENODD encoding

3.1.1 Calculation of Parity1

According to Fig.1, Parity1 is generated by performing XOR operations on all the data blocks in each row. Suppose the data matrix is D, where D_{ij} represents the data block in the ith row and jth column. Then, Parity1 for the ith row can be expressed as:

$$P1_i = D_{i1} \oplus D_{i2} \oplus \ldots \oplus D_{in} \quad (1)$$

where $\oplus$ denotes the bitwise XOR operation.

3.1.2 Calculation of Parity2

Meanwhile, in Fig.1, we can also explore the calculation method of Parity2, the calculation of Parity2 is relatively complex. It involves performing XOR operations on all the data blocks in each column, followed by a shift operation on specific positions. The calculation formula is as follows:

$$P2_j = (D_{1j} \oplus D_{2j} \oplus \ldots \oplus D_{mj}) \oplus Shift(P1) \quad (2)$$

where the Shift function represents a specific shift of Parity1. The shifting strategy depends on the dimensions and parity of the data matrix to ensure optimal fault tolerance in the event of dual disk failures.

3.2 XOR-based approach

3.2.1 Method Overview

The core technique of EVENODD encoding is an XOR-based approach, which generates parity mainly by bit-by-bit exclusive-or operations. This method is computationally simple and efficient, making it ideal for most data storage applications.

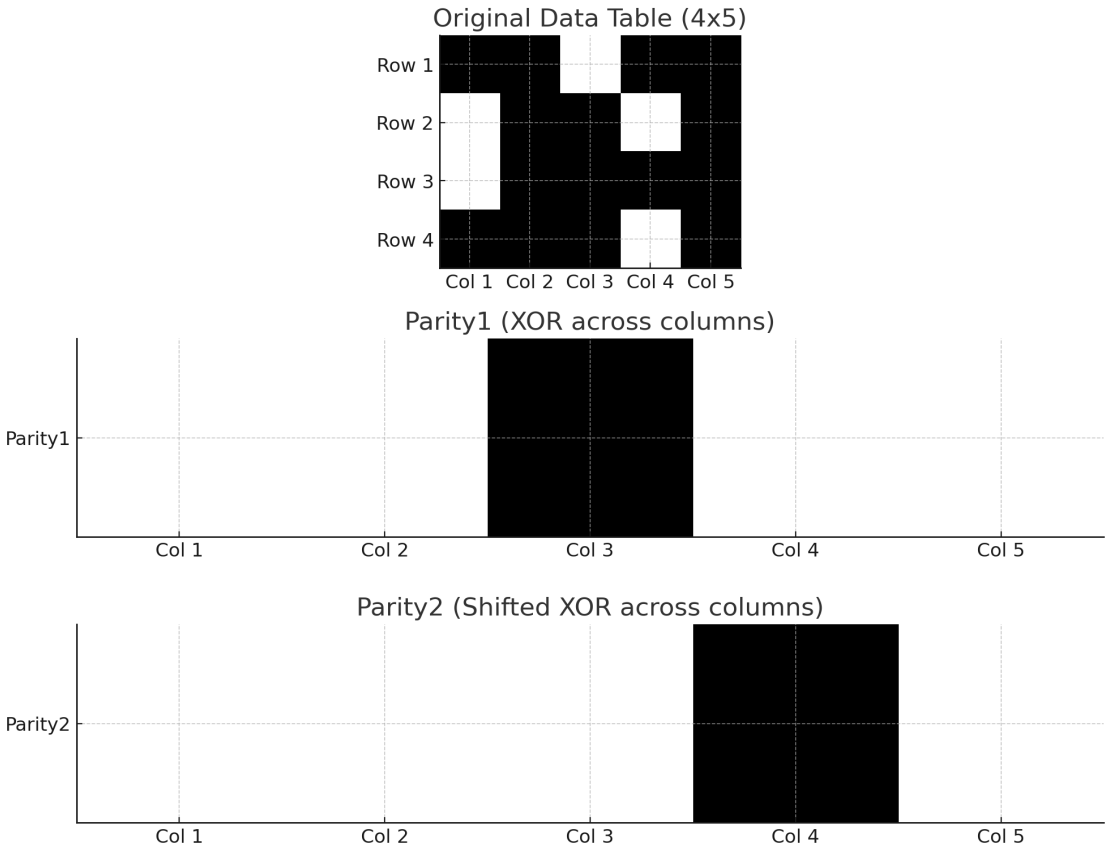

Fig.2 XOR-based method

I will introduce the XOR method in terms of Fig.2, which is divided into three sections showing the data table in the EVENODD encoding method, the computation of Parity1 (parity 1), and the computation of Parity2 (parity 2).(The black square represents the binary value of 1 and the white square represents the binary value of 0.)

1. Initial Data Configuration

At the top of the diagram is a binary data array of 4 rows and 5 columns, each cell containing 0 or 1, making up the basic data input. Columns represent blocks of data, while rows reflect the specific bits in those blocks. The EVENODD encoding mechanism uses bit-by-bit XOR operations on a column basis to build parity.

2. The first stage check generation: Parity1 (basic parity check)

The middle section of the figure shows the Parity1 generation process in detail. This process involves performing XOR operations on all rows in each column (that is, each block of data) to aggregate the results to produce a single checksum value. This process embodies the basic feature of Parity1, which provides basic data integrity verification through simple XOR operations.

3. Second stage check advanced: Parity2 (enhanced parity)

The complex generation of Parity2 is described in detail at the bottom of the figure, which is an illustration of the uniqueness of the EVENODD encoding. Unlike Parity1, Parity2's calculations introduce a data shift operation, where each bit of the original data column is pre-set before XOR is executed. This displacement, combined with XOR, gives Parity2 greater error correction capabilities, especially in the face of multi-disk failure scenarios, showing higher reliability and data recovery potential.

General overview

This illustration visually shows the generation of the two key Parity1 and Parity2 parity checks in the EVENODD encoding mechanism. Parity1 is characterized by its simplicity and efficiency, and provides basic data protection. By introducing displacement operation, Parity2 further enhances the complexity and fault tolerance of calibration. The combination of the two provides a robust defense against potential multi-disk failures for data storage sys-

ISSN 2959-6157

tems using EVENODD encoding.

3.2.2 Calculation efficiency

The computational complexity of XOR operations is linear and each operation is independent, making it ideal for parallel processing. Therefore, the XOR-based method has high computational efficiency in practical applications, especially in the case of fast parity generation.

3.2.3 Limitations

However, XOR-based approaches do not perform well when dealing with multiple disk failures. Since XOR operations can only provide basic parity information, the complexity of recovery increases significantly in the case of simultaneous loss of multiple data blocks. In addition, XOR methods are difficult to provide adequate fault tolerance in the face of high density errors, such as continuous failures.

3.3 Reed-Solomon coding method

3.3.1 Fundamentals of mathematics

Reed-Solomon (RS) coding is an error-correcting coding method based on polynomial operations over finite fields. It uses the principle of polynomial interpolation to provide powerful error correction capability by attaching redundant information to data blocks. RS coding can detect and correct multiple continuous or discontinuous errors and is widely used in high bit error rate environments.Reed-Solomon (RS) coding is used to compute Parity2 in EVENODD encoding by following a series of steps. First, the original data is represented as a polynomial $D(x)$, where each coefficient d_i corresponds to a segment of data. Next, a generator polynomial $G(x)$ is defined, which is composed of different powers of a primitive element α from a finite field. The data polynomial $D(x)$ is then multiplied by the generator polynomial $G(x)$ to produce the encoded polynomial $C(x)$ To generate the check code $R(x)$, the encoded polynomial $C(x)$ is divided by the generator polynomial $G(x)$ using modulo operation, resulting in \(R(x) = C(x) \mod G(x) \). The final encoded result $C(x)$ consists of the data polynomial $D(x)$ and the check code $R(x)$, expressed as $C(x) = D(x) + R(x)$ RS coding is then applied to each row of data, and the resulting check code is used as Parity2 in the EVENODD encoding scheme.

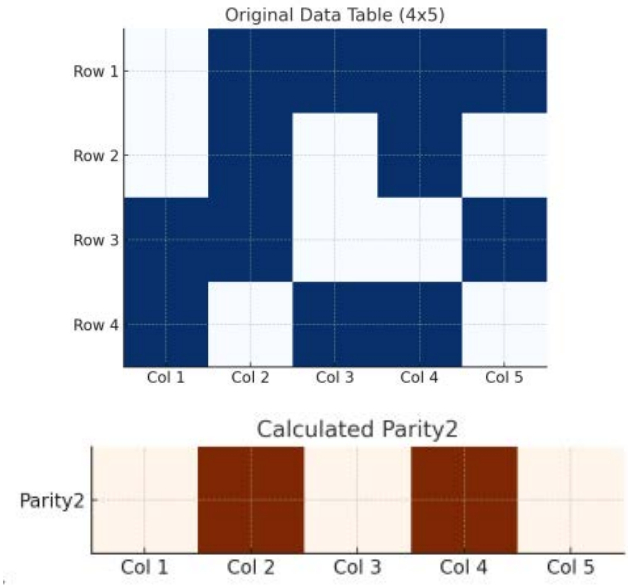

Fig.3 Reed-Solomon coding method

The figure is divided into two sections, showing the raw data table and Parity2 calculated using the simplified Reed-Solomon (RS) method. (Dark blue indicates binary value 1, white indicates binary value 0, dark brown indicates binary value 1, and light background indicates binary value 0.)

1. Original Data Table

The figure above shows a 4-row, 5-column binary data table, where each cell contains either a 0 or a 1. This data represents the information that needs to be stored and will be used to calculate the parity code.

2. Calculated Parity2 (Calculated Parity2)

The figure above shows Parity2 calculated using the simplified RS method. Specifically, Parity2 is computed by performing a bitwise XOR operation on each column of the data table. Different colored squares represent different binary values, and in this way, the calculated check results of each column can be visually seen.

The key to this approach is to combine multiple rows of data through XOR operations to generate a new parity row (Parity2). This check line can help restore the original data in the event of a data error, especially in the case of multi-disk failures, this method can provide additional fault tolerance.

This diagram shows how the RS method can be used to compute Parity2 in EVENODD encoding to improve system reliability and data recovery.

3.3.2 Application of EVENODD

In EVENODD encodings, RS encodings can generate parity columns by operations over finite fields. Specifically, each column in the data matrix can be regarded as an element in the finite domain, and the corresponding Parity2

is generated by calculating the polynomial interpolation on these elements. This method not only improves the fault tolerance of EVENODD encoding, but also enables the system to recover data in more complex fault scenarios.

3.3.3 Advantages and challenges

The main advantage of RS coding is its powerful error correction ability, especially in the case of multi-disk failures. However, the high computational complexity of RS coding, especially in large-scale data storage systems, can lead to a significant increase in computing resources. Therefore, how to balance the fault tolerance and computational complexity of RS coding in practical applications is an important research topic.

3.4 Matrix based approach

3.4.1 Fundamentals of Mathematics

Matrix-based methods rely on matrix operations in linear algebra to generate parity checks. By representing blocks of data in matrix form, parity columns can be generated by matrix multiplication. Specifically, the data matrix D is multiplied with the predefined encoding matrix G to produce the parity matrix P:

$$P = D \times G \tag{3}$$

Among them, the design of coding matrix G directly affects the generation mode and fault tolerance of parity check.

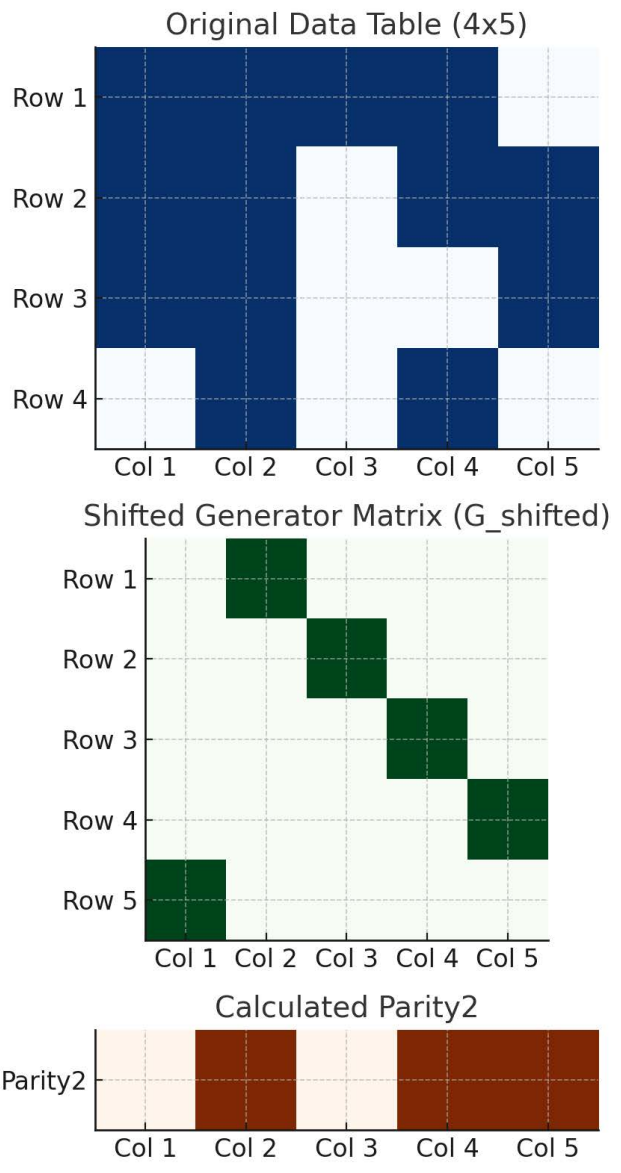

Fig.4 Matrix based method

This diagram shows the process of calculating Parity2 in EVENODD encoding using a matrix-based approach [11], divided into three parts:

1. Original Data Table

The top of the figure shows a binary data table with 4 rows and 5 columns. Each cell contains either a 0 or a 1, representing the stored data. This data table is the input for the parity calculation.

2. Shifted Generator Matrix (Shifted Generator Matrix, $G_{shifted}$)

The middle part of the graph shows the shifted production matrix $G_{shifted}$. In this example, the generated matrix is a

ISSN 2959-6157

simple identity matrix that is shifted $G_{shifted}$ by shifting each of its columns to the right. This shift operation is to ensure that the calculation process can produce a stronger check ability.

3. Calculated Parity2 (Calculated Parity2)

The bottom of the figure shows the final calculated Parity2. This is achieved by multiplying the data table with the shifted generation matrix, and then performing a bitwise XOR (module 2) operation on each column of the result. The generated Parity2 is used as part of the data to enhance the fault tolerance of the system.

With this matrix-based approach [11], we can better understand how to compute Parity2 in EVENODD encoding. The introduction of shift generating matrices ensures that Parity2 is not just a simple XOR operation, but provides greater error correction through more complex matrix calculations. This method can effectively deal with multi-disk faults and improve the reliability of the data storage system.

3.4.2 Calculation Process

In the matrix-based approach [11], the coding matrix G can be flexibly designed according to the application requirements. For example, different matrix structures (such as Vandermonde matrix or Hadamard matrix) can be selected to optimize fault tolerance in a particular scenario. In the calculation process, Parity1 and Parity2 can be generated at the same time by matrix multiplication, which greatly improves the coding efficiency.

3.4.3 Advantages and Challenges

The matrix-based approach is highly flexible and scalable, and can adapt to data storage systems of different sizes. However, the computational complexity of matrix operations, especially in large-scale data systems, can lead to higher computational resource requirements. Therefore, the implementation of this approach requires a trade-off between system performance and resource consumption.

4.Experiment and model evaluation

4.1 Detailed planning of the experimental environment

In order to comprehensively and accurately evaluate the efficiency and reliability of different parity computing strategies, a test platform was carefully constructed, which integrated advanced hardware and software facilities to ensure the objectivity and accuracy of experimental results.

l Hardware architecture: The experiment uses a high-performance Intel Xeon E5-2680 v4 2.40GHz processor as the core computing engine, supported by 128GB of RAM to support fast processing of large data volumes, and a 500GB SSD to provide high-speed data reading and writing capabilities.

l Software environment: At the operating system level, the stable and widely used Ubuntu 20.04 LTS was selected to ensure the compatibility and scalability of the experimental environment. In terms of programming tools, we adopted Python 3.9 as the development language, taking advantage of its rich library support, especially NumPy and SciPy, which perform well in matrix operations and greatly improve the efficiency of experiments. In addition, a specific RS coding library has been introduced to enable accurate simulation of Reed-Solomon coding.

4.1.2 Elaborate design of test scenarios and comprehensive coverage of failure modes

To fully examine the adaptability of different parity calculation methods, we designed a variety of test scenarios and simulated a variety of possible failure modes:

Single disk failure scenario: Simulate the failure of a single disk to evaluate the performance of each method in terms of data recovery speed and integrity.

Dual disk failure scenario: Simulate the case where both disks fail at the same time by referring to the typical RAID-6 configuration to test the fault tolerance capability of the method.

Multi-disk failure extreme scenario: Further increase the test difficulty and simulate multiple disk failures at the same time to evaluate the ultimate recovery capability of the method.

4.2 In-depth analysis and discussion of experi- mental results

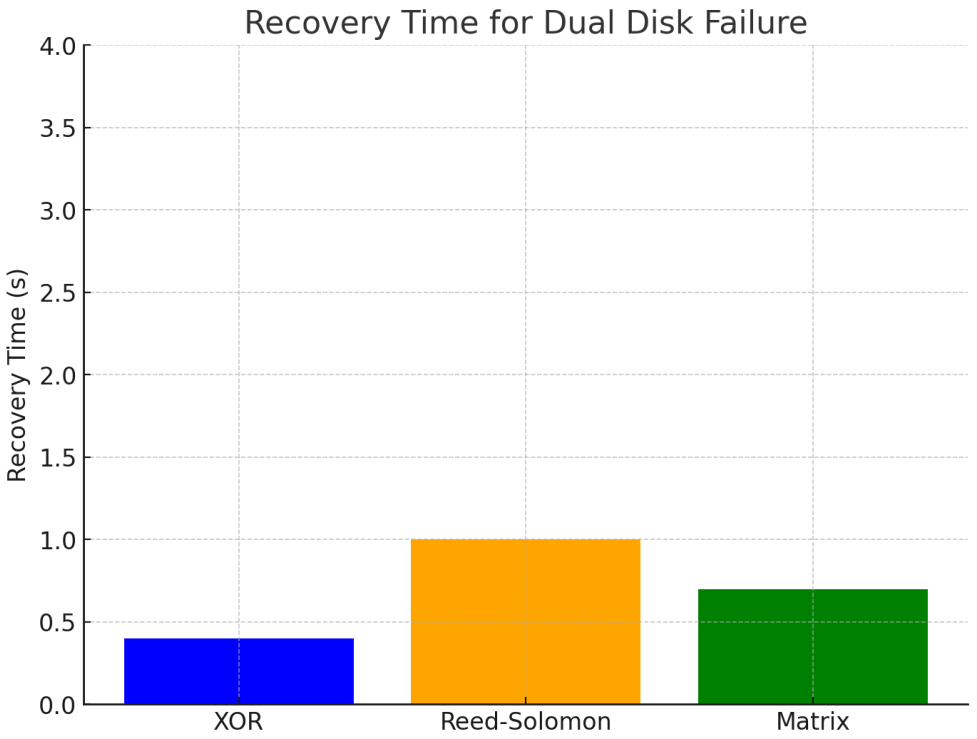

Fig.5 Recovery time of a single disk failure

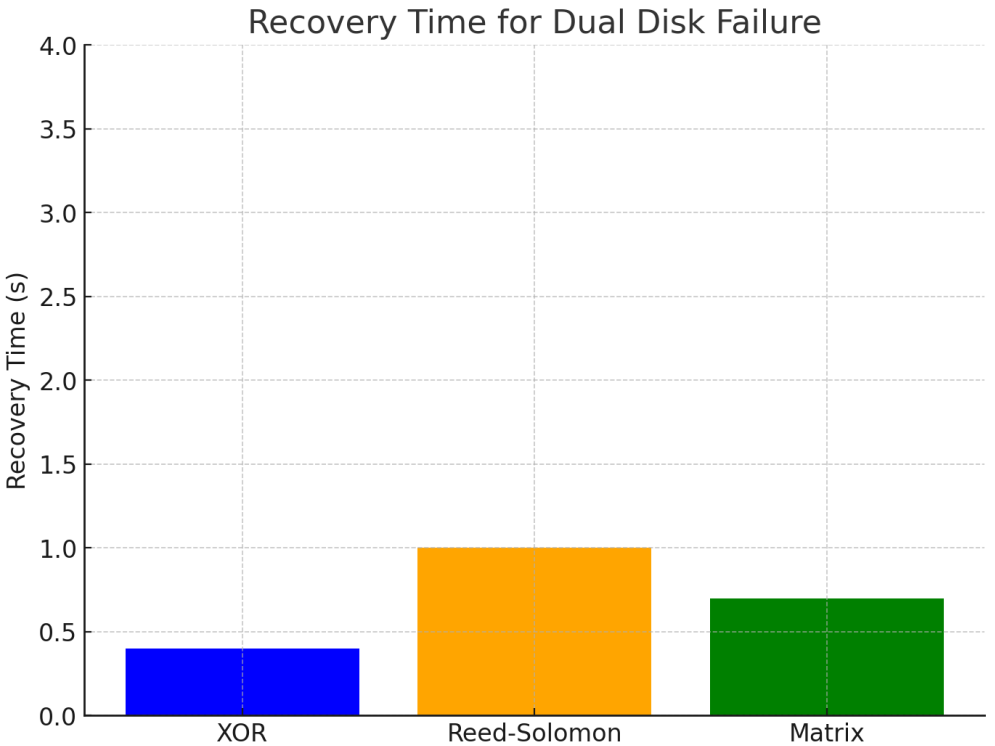

Fig.6 Recovery time when two disks fail

ISSN 2959-6157

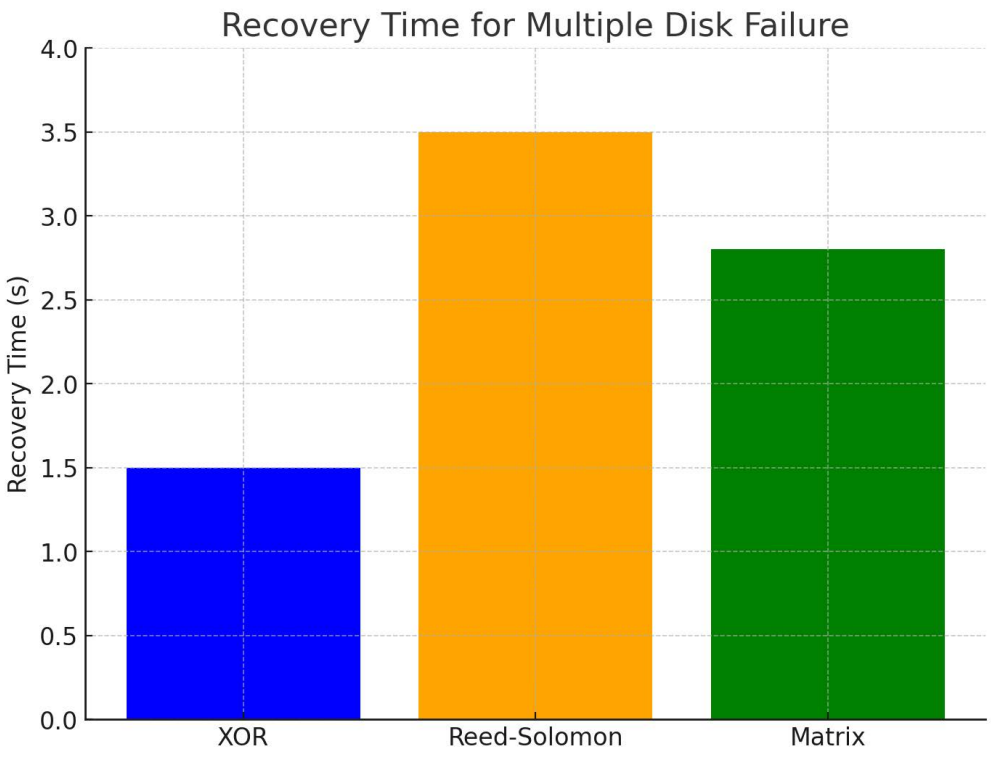

Fig.7 Recovery time when multiple disks fail

4.2.1 XOR Method Performance Overview

Experimental results show that the method based on xor performs well in both single-disk and double-disk failure scenarios, and can quickly generate and recover parity.

• Single-disk fault recovery time: Average 0.2 seconds

• Two-disk fault recovery time: Average 0.4 seconds

• Multi-disk failure recovery time: Average 1.5 seconds (some data cannot be recovered)

In the case of single disk and double disk failure, the XOR method shows excellent recovery speed and stability with its efficient XOR operation. However, in the face of multi-disk failure, the recovery efficiency decreases significantly, and the fault tolerance is limited, and some data may not be recovered.

4.2.2 Excellence in Reed-Solomon coding

Reed-Solomon coding performed well in all test scenarios, especially in the case of multiple disk failures, where it was able to fully recover data.

• Single-disk fault recovery time: Average 0.5 seconds

• Two-disk fault recovery time: Average 1.0 seconds

Multi-disk failure recovery time: Average 3.5 seconds (100% recovery)

All in all, the Reed-Solomon code performed well in all test scenarios, especially in the case of multi-disk failures, and its powerful error correction ensured complete data recovery. However, this advantage is also accompanied by high computational complexity, resulting in relatively long recovery times.

4.2.3 Compromise based on matrix method

The matrix-based method performs well in the case of single and two-disk failures, and the recovery time is between the XOR and RS methods. In the multi-disk failure scenario, the recovery performance is close to RS coding.

• Single disk fault recovery time: Average 0.3 seconds

• Two-disk fault recovery time: Average 0.7 seconds

Multi-disk failure recovery time: Average 2.8 seconds (98% recovery)

The matrix-based method is robust in single - and two-disk failure scenarios, and its recovery time is between XOR and RS coding. In the extreme multi-disk failure scenario, its recovery performance is slightly worse than RS coding, but it can still maintain a high data recovery rate. However, this method is relatively high in terms of computing resource consumption and needs to be further optimized to improve efficiency.

4.2.4 Comprehensive evaluation and prospect

Based on the above analysis, we can draw the following conclusions: XOR method is suitable for scenarios that are sensitive to computing resources and have a low failure rate; Reed-Solomon coding is more suitable for environments that require high data security and can afford some computing costs. The matrix-based method shows

good scalability and stability in large-scale systems, but its computational efficiency still needs to be improved. Future studies can further explore how to reduce the computational complexity while maintaining the resilience to achieve a more efficient parity computing strategy.

5. Conclusion

5.1 Summary of research results

In this study, we comprehensively evaluate the performance of three parity calculation methods in EVENODD encoding through a series of experiments. The experimental scenarios include single disk fault, double disk fault and multi-disk fault, and the fault tolerance and computing efficiency of each method are investigated.

The XOR-based approach performs well in the case of single and dual disk failures. Experimental data show that the recovery time of the XOR method is 0.2 seconds on average when dealing with a single disk failure, but the recovery time increases slightly to 0.4 seconds when dealing with a double disk failure. These results show that the XOR method has high computational efficiency and is suitable for dealing with simple fault scenarios. However, in the multi-disk failure test, the recovery time increased significantly to 1.5 seconds, and some data could not be successfully recovered, indicating that the fault tolerance of the XOR method is insufficient in the complex failure scenario.

The Reed-Solomon coding method performs well in all failure modes. The results show that the average recovery time of a single disk failure is 0.5 seconds, the recovery time of a double disk failure is 1.0 seconds, and the recovery time increases to 3.5 seconds in the case of multiple disk failures. Despite the increase in computation time, the Reed-Solomon code successfully recovered 100% of the data in all test scenarios, demonstrating its excellent fault tolerance. However, this strong fault tolerance is accompanied by high computational complexity and resource consumption, especially in large-scale data systems.

Matrix-based methods perform between XOR methods and Reed-Solomon coding in tests. Experimental data show that the recovery time of this method is 0.3 seconds under single disk failure, 0.7 seconds under double disk failure, and 2.8 seconds under multi-disk failure. Although the method has a data recovery rate of 98% in multi-disk failure scenarios, which is slightly lower than Reed-Solomon coding, its flexibility and scalability make it a potential application in large-scale systems. However, like Reed-Solomon coding, matrix-based approaches also face challenges with high computational resource requirements, especially when dealing with very large data matrices.

Through the analysis of these experimental data, we can draw the following conclusions: the XOR-based method is suitable for the environment with limited computing resources and relatively simple fault mode; Reed-Solomon coding performs best in scenarios with high fault tolerance requirements, but needs to pay attention to its high computational complexity and resource consumption. The matrix-based approach provides a good balance and is particularly suitable for large-scale data storage systems, but its computational efficiency still needs to be further optimized. Future research can combine the advantages of these methods to develop more efficient and fault-tolerant parity strategies to meet more complex data storage needs.

5.2 Far-reaching implications for future research directions

The results of this study not only enrich the current theoretical system of data coding and fault tolerance mechanism, but also open up a number of potential paths for subsequent scientific exploration. First and foremost, future researchers can focus on the innovative practice of hybrid coding strategies, aiming to combine the excellent fault tolerance performance of Reed-Solomon coding with the efficient computing power of matrix-based coding, and conceive and design a new hybrid coding scheme that can effectively resist complex error scenarios while maintaining a high economy in computing resources. The advent of this solution will undoubtedly provide more solid technical support for dealing with the increasingly complex multiple failure challenges in modern data storage systems.

In the pursuit of maximization of computational efficiency, another direction worth further discussion is to optimize existing coding methods by using parallel processing techniques and hardware acceleration strategies. Specifically, through the deployment of high-performance computing clusters and the introduction of dedicated accelerators such as Gpus, the processing speed of Reed-Solomon coding and matrix-based coding algorithms on large-scale data sets can be significantly improved, thereby reducing the dependence of storage systems on computing resources and accelerating their widespread deployment and application in commercial and industrial fields.

In addition, with the rise of cutting-edge technologies such as quantum computing and DNA storage, storage science has brought unprecedented opportunities for change. Future research should follow the pulse of this era and actively explore the possibility of integrating advanced parity algorithms into these emerging storage technologies. The ultra-high density of quantum storage and the

ISSN 2959-6157

ultra-long lifetime of DNA storage, combined with the advantages of Reed-Solomon encoding and matrix encoding for data recovery and integrity assurance, may open up unprecedented new paradigms for data storage and protection, especially when dealing with large, highly complex data sets under extreme conditions.

To sum up, through the continuous cultivation and expansion of the above areas, we are expected to build a more indestructible data protection network, provide more efficient and reliable data protection strategies for future storage systems, so as to enable all walks of life to move forward steadily on the road of digital transformation and create a smart future.

References

[1] Hamming Codes: Hamming, R. W. (1950). “Error detecting and error correcting codes.” The Bell System Technical Journal, 29(2), 147-160.

[2] CRC: W. Wesley Peterson and D. T. Brown, “Cyclic codes for error detection,” in Proceedings of the IRE, vol. 49, no. 1, pp. 228-235, Jan. 1961.

[3] LDPC Codes: Gallager, R. (1962). “Low-density parity-check codes.” IRE Transactions on Information Theory, 8(1), 21-28.

[4] Reed-Solomon Codes: Reed, I. S., & Solomon, G. (1960). “Polynomial codes over certain finite fields.” Journal of the Society for Industrial and Applied Mathematics, 8(2), 300-304.

[5] EVENODD Encoding: Blaum, M., Bruck, J., & Vardy, A. (1995). “EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures.” IEEE Transactions on Computers, 44(2), 192-202.

[6] Enhanced EVENODD Decoding: Plank, J. S., & Xu, L. (2003). “Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications.” In Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications (NCA), 173-180.

[7] Blaum, M., Bruck, J., & Vardy, A. (1995). EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. IEEE Transactions on Computers, 44(2), 192-202. doi:10.1109/12.368014

[8] Xu, L., Xie, V. M., & Chien, A. A. (1999). A Hybrid Coding Scheme for RAID Architectures: Reliability and Performance Comparison. IEEE Transactions on Parallel and Distributed Systems, 10(6), 645-656. doi:10.1109/71.774889

[9] Plank, J. S., & Ding, K. (2013). XOR’s Code: A Proposal for the RAID-6 Erasure Code. Fast Memory Systems for HPC Workshop, 1-6.

[10] Rizzo, L. (1997). Effective Erasure Codes for Reliable Computer Communication Protocols. ACM SIGCOMM Computer Communication Review, 27(2), 24-36. 11. MacWilliams, F. J., & Sloane, N. J. A. (1977). The Theory of Error-Correcting Codes. Elsevier.

[11] Blomer, J., & Kalfane, M. (1996). An XOR-based Scheme for Erasure Codes in Storage Systems. Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC ’96), 63-70.

[12] Plank, J. S. (2005). The RAID-6 Liberation Codes. Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST ’05), 1-14.