Enhancing Fault Tolerance in Distributed Storage Systems through an Extended EVENODD Code

Kejian Lu[1*], *Liyao* Song[2] and *Yuying* Xie[3]

[1]College of International, Beijing University of Posts and Telecommunications, Beijing, 100876, China
[2]College of Electrical Engineering, Jilin Engineering Normal University, Changchun, 130052, China
[3]School of Accounting & Finance, Xiamen University Tan Kah Kee College, Xiamen, 363123, China

Abstract. With the rapid development of technology, we have entered the era of big data. Therefore, the safe and effective handling of abundant amounts of data is essential. This paper analyzes and lists some commonly used data storage systems and encoding techniques and elaborates on the encoding and decoding process of one of them, namely EVENODD Code, which is widely used in distributed storage systems. At the same time, this article briefly discusses the possibility of improving the EVENODD Code. The original traditional EVENODD Code encoding can recover data with up to two block failures, but the computation process is more complex than that of multi-replica technology. As a result, it is less computationally efficient than multi-replica technology. The text briefly mentions the potential for enhancing the efficiency of data operations with the EVENODD Code, which could make it the optimal choice for large-scale, highly reliable applications such as cloud storage and media streaming systems.

1 Introduction

In the era of big data, the exponential growth in data generation from personal interactions, online activities, and numerous fields like weather forecasting, healthcare, and finance has led to an unprecedented demand for efficient and secure data storage solutions [1]. According to projections by the IDC (International Data Corporation), the global data volume is expected to reach 250 ZB (Zettabytes) by 2030 [2]. As data volume scales into PB (Petabyte) and EB (Exabyte) levels, traditional storage systems encounter significant challenges, such as single points of failure, limited scalability, and inefficiencies in resource utilization. Distributed storage systems have emerged as a preferred solution due to their inherent advantages in scalability, reliability, and cost-effectiveness, making them suitable for handling the vast datasets generated in this data-driven era [3].

Current research on distributed storage systems focuses primarily on fault-tolerance technologies to ensure data reliability and availability despite hardware failures or data losses.

* Corresponding author: lukejian@bupt.edu.cn

Traditional methods like replication, where multiple copies of data are stored across nodes, provide resilience but suffer from poor storage efficiency due to high redundancy [4]. Maximum Distance Separable erasure codes, including the widely adopted Reed-Solomon code and EVENODD Code, have become mainstream alternatives, offering optimal storage efficiency and fault tolerance [5, 6]. These codes enable data recovery even in the presence of failed disks or data blocks, reducing the need for excessive replication while maintaining high reliability. However, each method has limitations. Reed-Solomon codes, while effective, have high computational complexity during recovery, while the original EVENODD Code can only tolerate up to two simultaneous failures, which limits its applicability in highly reliable, large-scale storage systems [7, 8].

To address these challenges, this paper introduces an enhanced MDS (Maximum Distance Separable) array code, referred to as EVENODD*, which extends the capabilities of the traditional EVENODD Code. EVENODD* modifies the array structure to support a greater number of data blocks, improving system flexibility and fault tolerance. Our study examines the theoretical foundation of EVENODD*, analyzes its fault-tolerance performance, and evaluates its computational complexity. By leveraging XOR operations, the EVENODD* code achieves faster recovery times and reduced computational overhead compared to conventional MDS codes like Reed-Solomon. This makes EVENODD* particularly suitable for high-reliability applications, including cloud storage and streaming media, where efficient data recovery and optimal storage utilization are essential [9, 10].

2 Relevant techniques

This chapter mainly introduces the relevant knowledge of erasure codes in distributed storage systems, including fault-tolerant technology, and commonly used MDS codes.

With the development of technology, a large amount of data is generated on the network every day. In this case, it is inevitable that data will fail or be lost during storage and transmission [4]. In order to ensure the reliability of data, distributed storage systems have introduced storage redundancy and fault-tolerant technology.

The current mainstream storage redundancy and error tolerance technologies include multi replica technology and MDS erasure codes. This section will provide a brief introduction to these two fault-tolerant technologies.

2.1 Multi replica technology

Copy fault tolerance technology is a method of storing data in the form of multiple backups [5]. The principle is to replicate the original data into multiple copies and store these copies separately on different nodes. This allows each node to store a copy of the original data, so that it can still be read in the event of data failure or error during storage or transmission at other nodes.

This technology is simple to implement, easy to understand and operate [6]. When a node fails, a copy of the data can be obtained from the nearest available node, saving time while ensuring data availability.

But this technology also has obvious drawbacks. When the reliability of nodes in a storage system is relatively low, it is necessary to replicate a large number of replicas, resulting in low storage utilization and wastage of storage space. For example, in the case of replicating two copies, the storage utilization rate is 1/3.

2.2 MDS erasure codes

2.2.1 The properties of MDS erasure codes

MDS coding technology is an important data storage and fault-tolerant technique. It typically has optimal storage efficiency and can provide performance equivalent to system reliability at lower storage overhead.

MDS coding technology divides data into multiple data blocks and generates redundant check blocks through specific encoding algorithms [7]. These verification blocks are stored in different locations along with the original data blocks to ensure that even if some information is lost during transmission, the receiving end can reconstruct the original data by some special algorithms.

MDS code is a linear code with redundancy r and a minimum distance equal to r+1. This means that in a system consisting of n = k + m disks (or data blocks), as long as the number of failed disks (or data blocks) does not exceed m, the system can select any k disks (or data blocks) from the remaining k + m disks (or data blocks) and successfully recover the original data. In other words, this encoding mechanism can tolerate up to m = n - k disk (or data block) failures without losing data. Among them, k represents the number of original data disks, and m represents the number of verification disks.

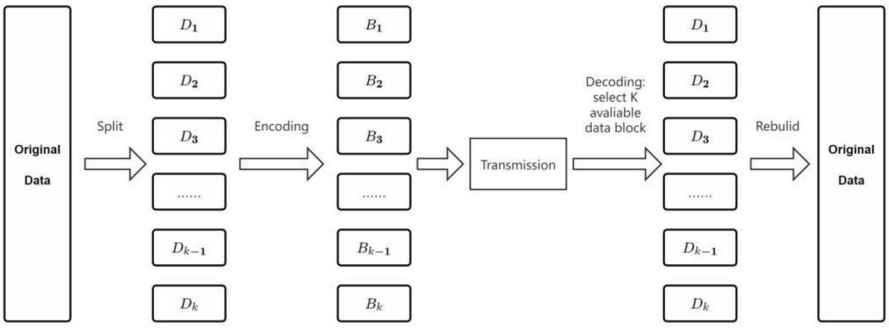

Fig. 1. Working principle diagram of MSD code (Photo credit: Original).

As show in the fig.1. MDS code technology is widely used in distributed storage systems, cloud storage, and RAID (Redundant Arrays of Independent Disk) storage systems. In these systems, MDS code technology ensures data security and availability by providing high reliability and fault tolerance. The commonly used MDS coding techniques include Reed-Solomon Code and EVENODD Code.

2.2.2 Reed-Solomon code

Reed-Solomon Code, also known as RS code, is an error correction coding technique widely used in data storage and communication fields. RS code is a forward error correction channel coding based on algebraic geometry and finite field theory.

RS code achieves error correction by representing data as polynomials over a finite field and generating parity disks. RS code can correct a certain number of errors, depending on the number of parity disks added during encoding [8]. RS code has a certain degree of flexibility, as it do not change the format and content of the original data during encoding and decoding, and can choose different n and k values as needed to adapt to different application scenarios.

RS code is typically represented as RS (n, k), where n is the total number of disks and k is the number of data symbols. This means that during the encoding process, the original data is divided into k data disks and n - k parity disks are added to generate a sequence of encoded disks of length n. The encoding process is generating a k * n matrix in the finite field GF ($\boldsymbol{p^t}$).

When n = 5 and k = 3, the encoding process steps in the finite field GF (7) are as follows:

$$\text{Generate matrix G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2^2 & 3^2 & 4^2 & 5^2 \end{bmatrix} \tag{1}$$

$$\text{Perform calculations using mod 7:G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2^2 & 3^2 & 4^2 & 5^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 2 & 2 & 4 \end{bmatrix} \tag{2}$$

$$\text{Encoding: m*G = c: } [a \quad b \quad c]\text{*G} = [U \quad V \quad W \quad X \quad Y] \tag{3}$$

Assuming V and X are lost, the decoding process is as follows:

$$\text{Extract G corresponding to U, W, and Y: G} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 5 \\ 1 & 2 & 4 \end{bmatrix} \tag{4}$$

Find the inverse matrix of G (G^{-1})

$$\text{m} = \text{c} * G^{-1} : [U \quad W \quad Y] * G^{-1} = \text{m} \tag{5}$$

2.2.3 EVENODD Code

EVENODD Code is a fault-tolerant technology based on error correction codes, which has a wide range of applications in storage systems, especially in RAID technology. It can tolerate errors in two data blocks simultaneously and adopts a specific layout strategy to ensure system stability [9]. The fundamental operation of this encoding algorithm involves calculating data using XOR operations according to specific rules.

EVENODD Code has m data disks and 2 parity disks. Here m needs to be prime. It is a (m-1)*(m+2) matrix.For ease of expression,use $\boldsymbol{a_{i,j}}$ to represent the i row and j column,where $0 \le i \le m-1, 0 \le j \le m+2$

The encoding rules for its parity disk are as follows:

$$\begin{aligned} a_{i,m} &= \sum_{j=0}^{m} a_{i,j} \\ \text{S} &= \sum_{j=1}^{m-1} a_{m-1-j,j} \\ a_{i,m+1} &= \text{S} + \sum_{j=0, j\neq i+1}^{m-1} a_{<i-j>m,j} \end{aligned} \tag{6}$$

Among them, (1,1) is the operation rule of the parity 1 disk, and (1,2) and (1,3) are the operation rules of parity 2 disk. The entry $a_{i,j}$ in the table and sum is always calculated by modulo 2 (XOR sum).

As show in the table 1. Assume m = 5 (that means it has 5 data disks), the EVENODD Code matrix looks like table 1. Among this, S = $a_{3,1} + a_{2,2} + a_{1,3} + a_{0,4}$.

Table 1. EVENODD Code encoding.

isk/Block	0	1	2	3	4	Parity 1	Parity 2
0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,6} = a_{0,0} + a_{0,1} + a_{0,3} + a_{0,4}$	$a_{0,7} = S + a_{0,0} + a_{3,2} + a_{2,3} + a_{1,4}$
1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,6} = a_{1,0} + a_{1,1} + a_{1,3} + a_{1,4}$	$a_{1,7} = S + a_{1,0} + a_{0,1} + a_{3,3} + a_{2,4}$
2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,6} = a_{2,0} + a_{2,1} + a_{2,3} + a_{2,4}$	$a_{2,7} = S + a_{2,0} + a_{1,1} + a_{0,2} + a_{3,4}$
3	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,6} = a_{3,0} + a_{3,1} + a_{3,3} + a_{3,4}$	$a_{3,7} = S + a_{3,0} + a_{2,1} + a_{1,2} + a_{0,3}$

3 Expanding the EVENODD Code

A new MDS array code EVENODD* code is proposed based on the construction method of EVENODD Code. The EVENODD Code is a $(p-1) \times (k+2)$ array with fault tolerance of 2. Its original array size is $(p-1) \times k$. The last two columns are coded check columns, where p is an odd prime number and k is a positive integer smaller than p.

The EVENODD* code proposed in this chapter is a $(p-1)\ \tau \times (k+2)$ array with fault tolerance of 2, the size of which is $(p-1)\ \tau \times k$. The last two columns are also coded check columns. p is an odd prime number, and τ is a positive integer.

The difference is that each node can store more data blocks, including $p-1$ and other positive integer blocks. The value of k is expanded from p to $p\tau$, and the optional parameter is more flexible. The EVENODD* code degenerates to EVENODD when $\tau = 1$[10].

In addition, the MDS attribute of EVENODD* code is proved and an effective decoding method is given.

The extended EVENODD (EEOD) code is a binary maximum distance separable (MDS) array code that can correct any three-column failure. However, the update complexity of EEOD code is not optimal. The update complexity is asymptotically optimal. In terms of encoding/decoding/updating, EEOD+ codes have lower computational complexity than EEOD codes.

Finally, the fault tolerance of EVENODD* code is improved under certain conditions, and the slope selection range is more flexible during construction.

3.1 Concepts

Before constructing EVENODD* code, we introduce the related concepts of modern algebra involved:

Mutual primes: If the greatest common divisor of two integers is 1, the two integers are mutual primes.

Irreducible polynomials: Cannot be decomposed into two polynomials of lower degree the product of the polynomial.

Mapping: There is a one-to-one correspondence between two sets, which means that for any element in one set, there is a unique determined element corresponding to it in the other set.

Identity mapping: The mapping of a set is itself, that is, any element in the set corresponds to the element itself.

Inverse mapping: The result of a composite mapping of two mappings that are inverse to each other is an identity mapping.

Isomorphism: represents a bijection that preserves structure in the field of modern algebra. structurally

In other words, two objects of isomorphism are completely equivalent, i.e., a relationship defined in one structure holds in the other structure.

Information polynomial: the coefficient of the term in each information polynomial represents the information data in the column node.

As show in the table 2. Check polynomial: the coefficient of the item in each check polynomial represents the check data in the column node.

Table 2. The meaning and interpretation of symbols.

symbol	explain
k	The number of columns in the information column
r	Number of virtual rows
p	An odd prime number $p\tau \geqslant k$

aij	For the data in i and $j, 0 \le j \le k-1$ indicates the information bit
Rpt	$j = k, k+1$ indicates the parity bit
Cpt	Ring $\mathbb{F}2[x]/(1+xp\tau)$
Rpt	A subset of the ring $Rp\tau$ whose elements are multipliers of $1+x\tau$
Mpr (x)	The polynomial $1+x\tau+\cdots+x(p-1)\tau$

3.2 New construction method

Recently, based on the EVENODD code, a new array code with the MDS attribute, EVENODD* code, is proposed. The EVENODD * code is an array of $(\mathrm{p}-1)\tau$ x $(k+2)$ encoded from $(\mathrm{p}-1)\ \tau$ x k information bits, where p is an odd prime, τ is a positive integer, and $k \le p\tau$. The elements in the i-th row and j-th column of the array are represented by the symbol $\mathrm{a_{i,j}}$. It indicates the information bit when $0 \le \mathrm{j} \le \mathrm{k}-1$. When j is equal to k or $k+1$, it indicate the parity bit. To facilitate the description, this article assumes that each node has τ virtual bits, namely $\mathrm{a_{(p-1)\tau+\mu,j}} = 0,\ 0 \le \mu \le\ \tau\ -1$, $0 \le \mathrm{j} \le \mathrm{k}+1$. Just as the EVENODD Code, these virtual does not exist in the actual storage.

The encoding formula for the k-th column of the horizontal check bit is:

$$a_{i,k} = \sum_{j=0}^{k-1} a_{i,j}, 0 \le i \le (p-1)\tau - 1 \tag{7}$$

The encoding formula for the skew check column K+1 is:

$$a_{i,k+1} = \sum_{j=0}^{k-1} a_{(i-j)modp\tau,j} + S_{imod\tau}, 0 \le i \le (p-1)\tau - 1 \tag{8}$$

Where Si is a common factor of τ, the solution is as follows:

$$S_i = \sum_{i=i+1}^{k-1} a_{(p-1)\tau-j,j}, 0 \le i \le \tau - 1 \tag{9}$$

As show in the table 3. The following shows an example of an encoding rule for EVENODD+(7,3,3), where the common cause

Table 3. An example of an encoding

$d_{0,0}$	$d_{0,1}$	$d_{0,2}$	$P_{0,3} = d_{0,0} + d_{0,1} + d_{0,2}$	$P_{0,4} = d_{0,0} + d_{5,2} + S_1$	$P_{0,5} = d_{0,0} + d_{5,1} + d_{3,2} + S_2$
$d_{1,0}$	$d_{1,1}$	$d_{1,2}$	$P_{1,3} = d_{1,0} + d_{1,1} + d_{1,2}$	$P_{1,4} = d_{1,0} + d_{0,1} + S_1$	$P_{1,5} = d_{1,0} + d_{4,2} + S_2$
$d_{2,0}$	$d_{2,1}$	$d_{2,2}$	$P_{2,3} = d_{2,0} + d_{2,1} + d_{2,2}$	$P_{2,4} = d_{2,0} + d_{1,1} + d_{0,2}$	$P_{2,5} = d_{2,0} + d_{0,1} + d_{5,2} + s_2$
$d_{3,0}$	$d_{3,1}$	$d_{3,2}$	$P_{3,3} = d_{3,0} + d_{3,1} + d_{3,2}$	$P_{3,4} = d_{3,0} + d_{2,1} + d_{1,2}$	$P_{3,5} = d_{3,0} + d_{1,1} + S_2$
$d_{4,0}$	$d_{4,1}$	$d_{4,2}$	$P_{4,3} = d_{4,0} + d_{4,1} + d_{4,2}$	$P_{4,4} = d_{4,0} + d_{3,1} + d_{2,2}$	$P_{4,5} = d_{4,0} + d_{2,1} + d_{0,2}$
$d_{5,0}$	$d_{5,1}$	$d_{5,2}$	$P_{5,3} = d_{5,0} + d_{5,1} + d_{5,2}$	$P_{5,4} = d_{5,0} + d_{4,1} + d_{3,2}$	$P_{3,5} = d_{5,0} + d_{3,1} + d_{1,2}$

Sub$S$ 1 and $S2$ are XOR to the first two check bits in column 4 and the first four check bits in column 5, respectively. Where S1=d5, 1+d4, 2. S2 is equal to d4, 1 plus d2, 2.

An EVENODD* code takes the parameters (p = 3, k = 5, τ = 3). Given all the bits in columns 0 through 4, the code is encoded geometrically. The result is shown in Table 3, where $S0$ = 1, $S1$ =0, $S2$ = 1.

$$\begin{cases} a_k(x) = \sum_{i=0}^{4} a_i(x) = x^3 + x^5, \\ a_{k+1}(x) = \sum_{i=0}^{4} x^i a_i(x) mod M_{3\cdot3}(x) = x + x^3 \end{cases} \tag{10}$$

Given five information polynomials on the ring $\mathbb{F}2[x]/M_{3\cdot3}(x)$, the formula as shown in the following: $(\mathrm{a_0(x)},\ \mathrm{a_1(x)},\ \mathrm{a_2(x)},\ \mathrm{a_3(x)},\ \mathrm{a_4(x)}) = (1+\ \mathrm{x},\ \mathrm{x},\ \mathrm{x^2}+\mathrm{x^5},\ \mathrm{x^2}+\mathrm{x^3}\ +\mathrm{x^4},\ 1\ +\ \mathrm{x^4})$, while $M_{3\cdot3}(x) = 1+\ \mathrm{x^3}\ +\ \mathrm{x^6}$. Then this code can get two calibration polynomials, as shown in (10). The results are the same regardless of geometric coding or algebraic coding.

3.3 Decoding algorithm

In distributed storage systems, the EVENODD Code enhances fault tolerance and data recovery efficiency, especially in cases where one or two columns of data are corrupted. This document details the EVENODD Code decoding algorithm based on the type and number of failed columns, ensuring effective data recovery following data block loss. The decoding algorithm is divided into four main cases: single-column failure, two-column failure (parity columns only), two-column failure (one data and one parity column), and failure of two data columns.

Case 1: Single-Column Failure. When only one column is corrupted, the decoding process is relatively straightforward. First, if the failed column is a data column (assume it's the f column), it can be reconstructed row-by-row using the horizontal parity column. The horizontal parity column is defined such that each row's parity block, $a_{i,k}$,is the XOR sum of all data blocks. To recover the missing data block$a_{i,f}$ in row I, we use the formula $a_{i,f} = a_{i,k} \oplus \sum_{j=0,j\neq f}^{k-1} a_{i,j}$where "$\oplus$" represents the XOR operation. XOR operations in GF (2) allow lossless data reconstruction, providing a highly efficient recovery method without additional computational overhead. On the other hand, if the failed column is the horizontal parity column (column k), we can recalculate each parity block row-by-row using the other data columns, with the formula$a_{i,k} = \sum_{j=0}^{k-1} a_{i,j}$This straightforward process has minimal computational complexity.

Case 2: Failure of Two Parity Columns. When two parity columns fail (assume the horizontal parity column k and the diagonal parity column k+1), data can be restored using complete data column information, allowing us to directly recalculate both parity columns. First, we rebuild the horizontal parity column by computing each row's parity block from the data columns, using $a_{i,k} = \sum_{j=0}^{k-1} a_{i,j}$Once the horizontal parity column is restored, we can recalculate the diagonal parity column with both the data and restored horizontal parity columns. The formula for the diagonal parity block is $a_{i,k+1} = \sum_{j=0}^{k-1} a_{(i-j)\bmod p,j} + S$ Where $S = \sum_{j=1}^{k-1} a_{p-1-j,j}$ is a shared factor that ensures the correct zeroing of virtual blocks in the diagonal parity column, in line with EVENODD encoding rules. This process fully restores both parity columns, maintaining data integrity across the entire parity set.

Case 3: Failure of One Data Column and One Parity Column. Suppose one data column (column f) and the diagonal parity column (column k+1) fail. In this case, decoding involves two steps: first recovering the data column, then using the restored data to recalculate the diagonal parity column. To reconstruct the data column, we use the horizontal parity column to recover each lost data block, row-by-row, as follows: $a_{i,f} = a_{i,k} \oplus \sum_{j=0,j\neq f}^{k-1} a_{i,j}$Once the data column is restored, we can recalculate the diagonal parity column by applying all data and horizontal parity values to the formula $a_{i,k+1} = \sum_{j=0}^{k-1} a_{(i-j)\bmod p,j} + S$,with $S = \sum_{j=1}^{k-1} a_{p-1-j,j}$remaining consistent as previously calculated.

Case 4: Failure of Two Data Columns. When two data columns fail (assuming columns f and g, where f<g<k), the decoding process is more complex due to the absence of additional original data. In such cases, both parity columns and intermediate sums are used to reconstruct the data. First, we calculate the shared factor$S = \sum_{i=0}^{p-2} (a_{i,k} + a_{i,k+1})$ to ensure the integrity of recovered data. Next, we define partial sums for the horizontal and diagonal parity columns to exclude the missing columns and set intermediate values. Specifically, we define the partial horizontal parity sum$S_i^{(0)} = \sum_{j=0,j\neq f,j\neq g}^{k} a_{i,j}$and the partial diagonal parity sum$S_i^{(1)} = S + a_{i,k+1} + \sum_{j=0,j\neq f,j\neq g}^{k-1} a_{(i-j)\bmod p,j}$.Using these intermediate sums, we can iteratively solve for the missing data in each of the failed columns, completing the recovery process.

3.4 Computational complexity analysis

The EVENODD decoding algorithm relies primarily on XOR operations, which provides a significant advantage in terms of computational complexity. For single-column recovery, whether for a data or parity column, the algorithm has a complexity of O(n), where n is the number of columns. In the case of two-column failure, restoring the horizontal and diagonal parity columns has a complexity of$O(n^2)$since each row recovery involves multiple XOR operations across columns. When both data columns fail, the complexity may increase to $O(n^2)$ due to the calculation of intermediate variables, such as partial sums in the horizontal and diagonal parities. Overall, EVENODD offers lower computational overhead compared to other MDS codes with $O(n^3)$complexity, such as Reed-Solomon codes, making it suitable for large-scale distributed storage systems.

Compared to traditional replication strategies, EVENODD achieves significantly higher storage efficiency. It introduces only two parity columns (horizontal and diagonal parity), so in a system with k data columns, the additional storage requirement is approximately 2/k. This storage overhead is about one-third of the conventional three-replica method. Moreover, as EVENODD uses linear encoding to generate and store parity data, it reduces additional storage needs and improves storage utilization.

EVENODD supports the recovery of up to two failed columns without compromising data integrity, which is advantageous for handling hardware failures or node losses. In the event of a single column failure, EVENODD can directly recover using either the horizontal or diagonal parity column. When both parity columns or one data and one parity column fail, the algorithm can restore integrity by sequentially recovering each column. In the extreme case of two data columns failing, the algorithm can still recover the data using the parity columns by constructing intermediate variables, significantly enhancing system reliability in distributed storage environments.

Due to its advantages in computational complexity and storage efficiency, EVENODD is well-suited for large-scale, high-reliability applications, such as cloud storage, streaming media storage, and enterprise distributed file systems. Compared to MDS codes like Reed-Solomon, EVENODD demonstrates advantages in recovery speed and storage efficiency, making it particularly applicable to large-scale node systems and high-volume data storage scenarios.

4 Conclusion

This paper presented the EVENODD* code, an enhancement of the traditional EVENODD fault-tolerance code for distributed storage systems, addressing the limitations of current fault-tolerance methods in handling large-scale data reliably and efficiently. By expanding the array structure and leveraging Maximum Distance Separable properties, EVENODD* allows for more flexible system configurations and higher fault tolerance, supporting larger data blocks within nodes. The theoretical analysis and computational complexity evaluations demonstrate that EVENODD* significantly reduces recovery time and computational overhead compared to conventional methods such as replication and Reed-Solomon codes. These improvements position EVENODD* as a robust solution for high-reliability applications, including cloud storage and media streaming, where efficient data recovery and optimized storage utilization are paramount. Future research could explore further optimization of the EVENODD* code to enhance fault tolerance beyond two data block failures, potentially adapting it for triple failure correction in scenarios demanding extremely high resilience. Additionally, investigating adaptive encoding schemes that dynamically adjust fault tolerance based on system load and storage demands could provide even greater flexibility and efficiency. Research on integrating EVENODD* with emerging technologies

like distributed edge computing and hybrid cloud storage could also open up new avenues for scalable, high-reliability storage solutions in diverse and evolving data environments.

Authors contribution

All the authors contributed equally and their names were listed in surname order.

References

1. H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, et al., Big data and its technical challenges, Commun. ACM **57(7)**, 86-94 (2014)
2. V. Turner, D. Reinsel, J. F. Gantz, S. Minton, The Digital Universe of Opportunities: Rich Data and Increasing Value of the Internet of Things (2014)
3. C. Qu, A Comparative Study on Fault-Tolerant Coding Techniques in RAID6 Analyzing the Efficiency and Limitation of EVENODD and RDP Codes, Sci. Technol. Eng. Chem. Environ. Prot. **1(9)** (2024)
4. Z. Zhao, Y. Peng, X. Zhu, X. Wei, X. Wang, J. Zuo, Research on Prediction Of Electricity Consumption In Smart Parks Based On Multiple Linear Regression, 2020 IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC), Chongqing, China, pp. 812-816 (2020)
5. W. Tao, Application and Performance Analysis of EVENODD in RAID Architecture, Highlights Sci. Eng. Technol. **87,** 45-51 (2024)
6. H. Xu, X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, Research of Pipeline Leak Detection Technology and Application Prospect of Petrochemical Wharf, 2020 IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC), Chongqing, China, pp. 263-271 (2020)
7. C. Wang, Triple-Disk Recovery RDP Coding: A Robust Solution for Data Storage in RAID Configurations, Highlights Sci. Eng. Technol. **81,** 519-526 (2024)
8. X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, Action recognition method based on wavelet transform and neural network in wireless network, Proc. 2021 5th Int. Conf. Digit. Signal Process., pp. 60-65 (2021)
9. X. Zhu, Y. Zhang, Z. Zhao, J. Zuo, Radio frequency sensing based environmental monitoring technology, in Fourth Int. Workshop Pattern Recogn. (SPIE, 2019), Vol. **11198,** pp. 187-191
10. X. Wei, X. Zhu, X. Wang, Z. Zhao, J. Zuo, Fuzzy Fault Tree Analysis Method and Its Application in Fault Diagnosis of Denitration System in Thermal Power Plant, Proc. 2020 8th Int. Conf. Inf. Technol.: IoT Smart City, pp. 227-232 (2020)