Advancing Data Resilience: An In-Depth Study of Pyramid Codes for Fault Tolerance and Repair Cost Optimization

*Yanqiu Chen**

School of Science, Shandong Jiaotong University, 250357, Jinan, China

Abstract. In the digital era, distributed storage systems are essential for managing vast volumes of data generated by cloud computing, big data, and the Internet of Things. These systems face challenges such as scalability, fault tolerance, and cost control. Pyramid Codes have emerged as an effective solution, enhancing fault tolerance and reducing repair costs. This paper presents a thorough analysis of Pyramid Codes, examining their structure, performance, and advantages over traditional Reed-Solomon Codes. The study delves into fault tolerance mechanisms, demonstrating Pyramid Codes' ability to handle single, double, and triple disk failures efficiently. Additionally, strategies are proposed for optimizing reliability, such as increasing parity length and creating additional parity layers, which enhance repair capabilities without substantially increasing costs. The potential applications of Pyramid Codes in scenarios requiring high fault tolerance and low repair costs are discussed, emphasizing their role in advancing distributed storage resilience. Future directions include exploring adaptive frameworks for real-time data repair and optimizing parity distribution to balance security and cost-efficiency, paving the way for robust and efficient storage solutions in distributed systems.

1 Introduction

In the digital age, the exponential growth of data driven by advancements in cloud computing, big data, and the Internet of Things (IoT) has amplified the demand for efficient and resilient data storage solutions. Distributed storage systems have become crucial in modern data infrastructure, offering a scalable means to store, access, and process vast volumes of data across numerous network nodes [1]. By utilizing distributed computing principles, these systems enhance resource utilization and processing efficiency. As IoT continues to expand, connecting billions of devices and generating immense data flows, the need for distributed storage solutions that balance scalability, fault tolerance, and cost-efficiency has become more pressing than ever [2, 3].

Currently, distributed storage systems face several limitations, particularly concerning fault tolerance and repair costs. Traditional data replication, while providing reliability, becomes increasingly inefficient due to high redundancy and storage overhead in large-scale

*Corresponding author: 210151123@stu.sdjtu.edu.cn

systems. Erasure coding methods, such as Reed-Solomon (RS) codes, are widely implemented for data protection, offering robust error correction and data recovery in distributed environments [4]. However, despite their strengths, RS codes come with high repair costs and implementation complexities, making them less suitable for scenarios where cost-efficiency and repair optimization are critical [5]. Recently, Pyramid Codes have gained recognition as a promising alternative, featuring a hierarchical structure that reduces data volume requirements for recovery operations, thereby enhancing fault tolerance and minimizing repair costs.

This paper provides a comprehensive study of Pyramid Codes, focusing on their structural advantages, fault tolerance mechanisms, and repair cost optimization in comparison to RS codes. Key contributions include an analysis of Pyramid Codes' ability to handle single, double, and triple disk failures efficiently and strategies for improving reliability by extending parity length and adding additional parity layers. The paper also explores potential application scenarios where Pyramid Codes can be most effective, particularly in high-resilience and low-cost distributed storage settings. Additionally, future research directions are discussed, including the development of adaptive frameworks for real-time data repair and optimized parity distribution techniques to further enhance cost-efficiency and fault tolerance [6, 7]. This work aims to support the ongoing advancement of distributed storage systems, establishing Pyramid Codes as a valuable tool in achieving robust data resilience in today's rapidly evolving digital landscape.

2 Relevant theories

2.1 Definition of pyramid codes

Pyramid Code usually refers to programming techniques used to build pyramid structures. Pyramid (n, k) first proposed the idea of hierarchical coding. The Fig 1 below shows Microsoft Azure Storage (16,12) Pyramid Codes [6].

Fig. 1. (16,12) Pyramid code (Photo credit: Original).

As show in the fig.1. Start with 3 global parities like in Reed-Solomon code.

As show in the fig.2. Divide Parity 3 into two non-overlapping components: one involving data nodes from Date 1 to Date 6, and the other involving data nodes from Date 7 to Date 12. These are parities 3 and 4.

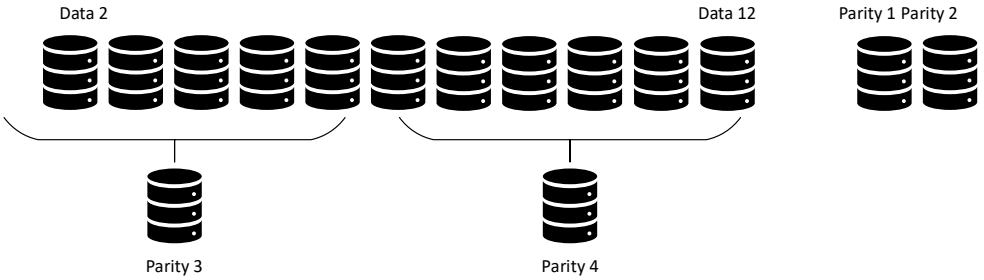

Fig. 2. Divide Parity 3 into two non-overlapping components (Photo credit: Original).

Parity 1:

First stripe of Parity 1: $p_{11}=d_{11}+d_{21}+...+d_{12,1}$
Second stripe of Parity 1:$p_{12}=d_{12}+d_{22}+...+d_{12,2}$
…
Last stripe of Parity 1:
$p_{1k}=d_{1k}+d_{2k}+d_{3k}+d_{4k}+d_{5k}+d_{6k}+d_{7k}+d_{8k}+d_{9k}+d_{10k}+d_{11k}+d_{12,k}$
Parity 2:
First stripe of Parity 2:$p_{21}=1*d_{11}+2*d_{21}+2^2*d_{31}...+2^{11}*d_{12,1}$
Second stripe of Parity 2:$p_{22}=1*d_{12}+2*d_{22}+2^2*d_{32}...+2^{11}*d_{12,2}$
…
Last stripe of Parity 2:
$p_{2k}=1*d_{1k}+2*d_{2k}+2^2*d_{3k}+2^2*d_{4k}+2^2*d_{5k}+2^2*d_{6k}+2^2*d_{7k}+2^2*d_{8k}+2^2*d_{9k}+2^2*d_{10k}+2^2*d_{11k}+2^{11}*d_{12,k}$
Parity 3:
First stripe of New Parity 3: $p_{31}=1*d_{11}+4*d_{21}+4^2*d_{31}...+4^5*d_{61}$
Second stripe of New Parity 3: $p_{32}=1*d_{12}+4*d_{22}+4^2*d_{32}...+4^5*d_{62}$
…
Last stripe of New Parity 3: $p_{3k}=1*d_{1k}+4*d_{2k}+4^2*d_{3k}+4^3*d_{4k}+4^4*d_{5k}+4^5*d_{6k}$
Parity 4:
First stripe of New Parity 4: $p_{41}=4^6*d_{71}+4^7*d_{81}+4^8*d_{91}...+4^{11}*d_{12,1}$
Second stripe of New Parity 4:$p_{42}=4^6*d_{72}+4^7*d_{82}+4^8*d_{92}...+4^{11}*d_{12,2}$
…
Last stripe of New Parity 4:$p_{4k}=4^6*d_{7k}+4^7*d_{8k}+4^8*d_{9k}+4^9*d_{10k}+4^{10}*d_{11k}+4^{11}*d_{12,k}$

2.2 Introduction to supporting codes

2.2.1 Generalized pyramid code

Galois Field (GF) is an important concept in mathematics, widely used in areas such as coding theory, cryptography, combinatorial design, and number theory. Galois Field (GF) is an important concept in mathematics, widely used in areas such as coding theory, cryptography, combinatorial design, and number theory. A field that has a finite number of elements is what it is [7]. The Galois Field's elements are capable of performing addition, subtraction, multiplication, and division (except for zero). They are in accordance with the fundamental properties of a field (such as commutativity, associativity, distributivity, etc.). It is usually denoted by GF (p^t). Field GF (p^t) is the basis for every codeword entry, with p being a prime and a positive integer.

Operations in Galois Field (addition and multiplication) are usually performed modulo a certain polynomial:

Addition: Perform addition of polynomial coefficients modulo p [8].

Multiplication: After multiplying the polynomials, reduce the result modulo some irreducible polynomial.

As show in the table 1 and table 2. Provide addition and multiplication tables for GF (7).

Here are the calculations for addition.

(1+6) mod 7 = 0; (4+6) mod 7 = 3;
(2+6) mod 7 = 1; (5+6) mod 7 = 4;
(3+6) mod 7 = 2; (6+6) mod 7 = 5;

Table 1. Addition Table

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Here are the calculations for multiplication.

(2*4) mod 7 = 1; (3*5) mod 7 = 1; (5*5) mod 7 = 4;
(2*5) mod 7 = 3; (3*6) mod 7 = 4; (5*6) mod 7 = 2;
(2*6) mod 7 = 5; (4*4) mod 7 = 2; (6*6) mod 7 = 1;
(3*3) mod 7 = 2; (4*5) mod 7 = 6;
(3*4) mod 7 = 5; (4*6) mod 7 = 3;

Table 2. Multiplication Table

*	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

2.2.2 Reed-solomon code

Reed-Solomon Code (RS Code) is an important erasure coding technology, which has the nature of MDS encoding, usually encoded and decoded on GF (2^m). Like Repetition code and Hamming code, Reed-Solomon is also a linear block code. RS decoders are also developed to correct errors (as opposed to erasures). The use of it is widespread in data transmission and storage [9]. RS is generally expressed in the form of RS (n, k), where n is the code length, k is the message length, and k/n is the code rate.

$$G=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ a_1^{k-1} & a_2^{k-1} & \cdots & a_n^{k-1} \end{bmatrix} \quad (1)$$

Here $a_1,a_2,...,a_n$ are distinct elements of GF (p^t). Each codeword entry is from field GF(p^t), where p is a prime and t is a positive integer.

Fig. 3. (14,10) RS Code (Photo credit: Original).

As show in the fig.3. Certain entries in the damaged codeword c are unavailable when erasures are present.

Make the assumption that this can be fixed with a maximum of (n-k) erasures. Consequently, c always has (at least) k entries accessible.

$\underline{c}$ can be the subvector of c and has these k available positions.

Certain entries in the damaged codeword c are unavailable when erasures are present.

Up to (n-k) erroneous entries are corrected by the code, therefore there are always k entries in c that are accessible.

Assume c is the subvector of c that has the available positions.

Then, $\underline{G}$=m*$\underline{c}$

The RS Code formula that is frequently employed is $\underline{G}$=m*$\underline{c}$, in which $\underline{G}$ is the kxk submatrix of $\underline{G}$ that has the same number of columns.

Construct a RS code over GF (7) for k=3 and n=5, and specify the generator matrix.

$$G=\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 2 & 3 & 4 & 5\\ 1 & 2^2 & 3^2 & 4^2 & 5^2\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 2 & 3 & 4 & 5\\ 1 & 4 & 9 & 16 & 25\end{bmatrix} \quad (2)$$

Because GF (7), the numbers in the matrix cannot exceed 7, numbers greater than 7 should be taken modulo 7.

9 mod 7=2 ;16 mod 7=2 ;25 mod 7=4

Therefore,

$$G=\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 2 & 3 & 4 & 5\\ 1 & 4 & 2 & 2 & 4\end{bmatrix} \quad (3)$$

Perform encoding of the input message [1 2 3], and specify the associated codeword.

According to the formula mentioned ($\underline{G}$=m*$\underline{c}$) above, it can be calculated that

$$[1\quad 2\quad 3]\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 2 & 3 & 4 & 5\\ 1 & 4 & 2 & 2 & 4\end{bmatrix}=[6\quad 3\quad 6\quad 1\quad 2] \quad (4)$$

1*1+2*1+3*1=6

1*1+2*2+3*4=17 mod 7=3

1*1+2*3+3*2=13 mod 7=6

1*1+2*4+3*2=15 mod 7=1

1*1+2*5+3*4=23 mod 7=2

Suppose the first and second coded entry failed. Perform complete decoding and verify correctness.

According to the above description, it can be seen $[?\quad ?\quad 6\quad 1\quad 2]$

$$[6\quad 1\quad 2]*\begin{bmatrix}1 & 1 & 1\\ 3 & 4 & 5\\ 2 & 2 & 4\end{bmatrix}^{-1}=[m_1\quad m_2\quad m_3] \quad (5)$$

According to the addition and multiplication rules of GF (7) mentioned above, it can be calculated that

$$\left[\begin{array}{ccc|ccc}1 & 1 & 1 & 1 & 0 & 0\\ 3 & 4 & 5 & 0 & 1 & 0\\ 2 & 2 & 4 & 0 & 0 & 1\end{array}\right]\rightarrow\left[\begin{array}{ccc|ccc}1 & 0 & 0 & 3 & 6 & 4\\ 0 & 1 & 0 & 6 & 7 & 6\\ 0 & 0 & 1 & 6 & 0 & 4\end{array}\right] \quad (6)$$

Therefore,

$$[6\quad 1\quad 2]\begin{bmatrix}3 & 6 & 4\\ 6 & 7 & 6\\ 6 & 0 & 4\end{bmatrix}=[1\quad 2\quad 3] \quad (7)$$

2.2.3 Parity

Through the following EVENODD table 3, you can clearly understand parity1 and parity2.

Odd Parity: The parity bit is set to 1 if there are even 1s in the data, and it is set to 0 if there are odd ones.

Table 3. Parity Calculation Table

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

1	0	1	1	1	1	1	0
2	1	1	1	0	1	1	0
3	0	1	0	1	1	1	1

Parity 1: Disk 0 :0+1+0+0+0=0 Disk 1:0+1+1+1+1=1

Disk 2 :1+1+1+0+1=1 Disk 3 :0+1+0+1+1=1

Parity 2: sum S first:$\sum_{t=1}^{m-1} a_{m-1-t,t}$ m=5

t=1 m-1-t=3 First entry in the summation is $a_{3,1}$

t=2 m-1-t=2 In addition to indices (3,1), in the summation are also indices (2, 2), (1, 3), and (0, 4).

S=1+1+1+0=1

First row l =0 $a_{0,6}=S+\sum_{t=0,t\neq1}^{4} a_{<0-t>5,t}$

t=0 <0,0>5=0 t=2 <0-2>5 means -2 mod 5=3

t=3 <0-3>5 means -3 mod 5=2

t=4 <0-4>5 means -4 mod 5=1

This is $a_{0,6}=S+a_{0,0}+a_{3,2}+a_{2,3}+a_{1,4}=1+0+0+0+1=0$

Second row l =1 $a_{1,6}=S+\sum_{t=0,t\neq2}^{4} a_{<1-t>5,t}$

t=0 <1-0>5 means 1 mod 5=1 t=1 <1-1>5 =0

t=3 <1-3>5 means -2 mod 5=3

t=4 <1-4>5 means -3 mod 5=2

This is $a_{1,6}=S+a_{1,0}+a_{0,1}+a_{3,3}+a_{2,4}=1+0+1+1+1=0$

Third row l =2 $a_{2,6}=S+\sum_{t=0,t\neq3}^{4} a_{<2-t>5,t}$

t=0 <2-0>5 means 2 mod 5=2

t=1 <2-1>5 means 1 mod 5=1

t=2 <2-2>5=0 t=4 <2-4>5 means -2 mod 5=3

This is $a_{2,6}=S+a_{2,0}+a_{1,1}+a_{0,2}+a_{3,4}=1+1+1+0+1=0$

Fourth row l =3 $a_{3,6}=S+\sum_{t=0,t\neq4}^{4} a_{<3-t>5,t}$

t=0 <3-0>5 means 3 mod 5=3

t=1 <3-1>5 means 2 mod 5=2

t=2 <3-2>5 means 1 mod 5=1 t=3 <3-3>5=0

This is $a_{3,6}=S+a_{3,0}+a_{2,1}+a_{1,2}+a_{0,3}=1+0+1+1+0=1$

3 System analysis and application research

3.1 Comparison with RS codes

The basic concepts of Pyramid Code and RS Code are fundamentally different. Pyramid Code is mainly used for image processing, particularly for multi-resolution image representation and compression [10]. Efficient image analysis and processing is made possible by creating images with multiple resolution levels.RS Code is an error correction coding method widely used in digital communication and storage. By using redundant data, it is able to recover lost or corrupted information, making it particularly suitable for correcting multiple errors.

Actually, there is no direct connection between them; they can only be used together in specific applications. For example, in digital image transmission, the image is first compressed using pyramid coding, and then RS Code is applied to ensure that even if some data is lost during transmission, the complete image can still be restored.

3.2 Fault tolerance

Fig. 4. (16,12) Pyramid code (Photo credit: Original).

In the case of the (16,12) Pyramid code described in Fig. 4, we can analyze the fault tolerance and repair mechanisms for data disks and parity disks as follows: For Data Disk 1 and Data Disk 2, they can be restored by accessing Parity 1 (which requires reading all data disks) and Parity 3, along with Data Disks 3, 4, 5, and 6. This results in a total of 6 disk accesses. Similarly, when repairing Data Disks 1, 2, and 3, access to both Parity 1 and Parity 2 (both of which need all data disks), Parity 3, and Data Disks 4, 5, and 6 is necessary, totaling 6 disk accesses as well. However, for the combination of Data Disks 7, 8, 11, and 12, the Pyramid codes cannot correct this level of failure since it surpasses the limit of handling 3 simultaneous disk failures.

This coding scheme is designed to accommodate up to three disk failures by leveraging redundancy from the remaining disks. The repair strategy for a single data disk involves accessing another parity disk, whereas the recovery of two data disks involves accessing additional parity disks. Any single parity disk can be reconstructed from all data disks. In the event of failures among all parity disks, recovery is possible by re-encoding from all data disks.

The design of the pyramid code ensures that the number of equations available matches the number of unknowns needed for solving during disk recovery, thus facilitating the recovery of all single, double, and triple fault scenarios. The specific recovery equations and required accesses depend on the nature and location of the disk failures, indicating whether they fall within the repairable scenarios outlined above..

3.3 Reliability and repair capabilities

According to the above text, it will encounter some errors that cannot be fixed, so the following solutions have been implemented.

Increase the length of Parity 3 and Parity 4.

As show in the fig.5. As mentioned above, no more than 4 errors can appear simultaneously in a parity (whether it is parity3 or parity4), so the length of parity3 and parity4 can be extended to ensure there are enough elements for equation calculations, thereby enabling repairs.

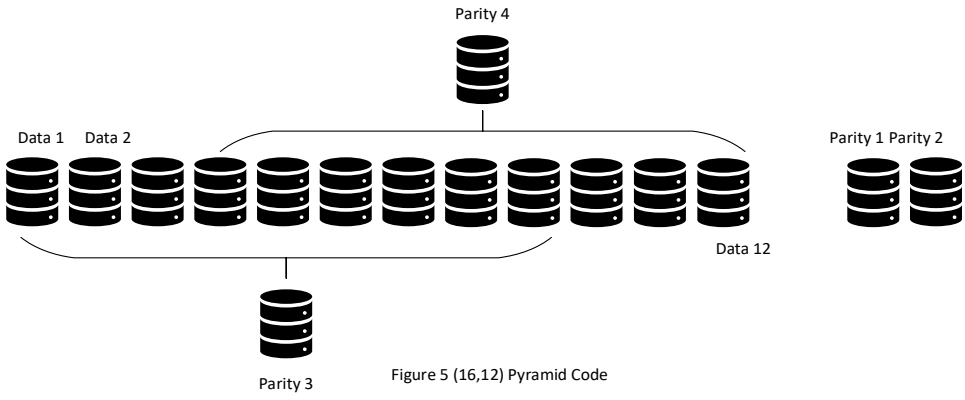

Fig. 5. (16,12) Pyramid code (Photo credit: Original).

Create a new parity.

As show in the fig.6. As previously stated, it is not possible for four errors to appear simultaneously in a single parity (whether it is parity3 or parity4). Therefore, by adding a parity5 (i.e., splitting parity3 into parity3, parity4, and parity5), I can perform equation calculations to avoid any parity being unable to compete due to too many errors.

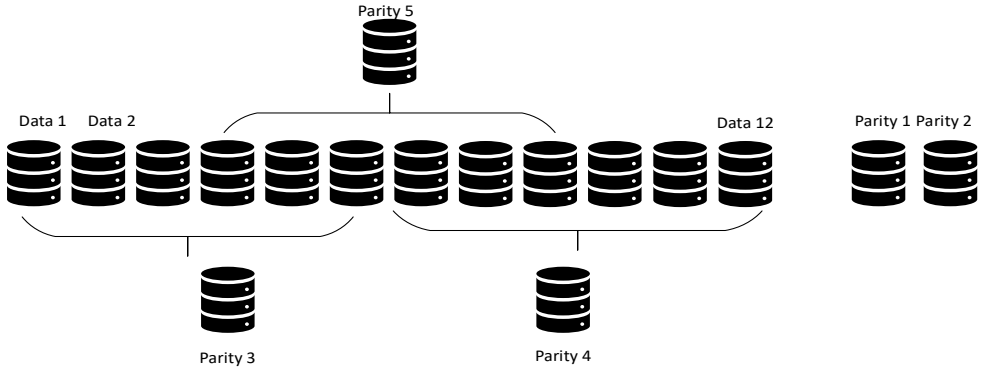

Fig. 6. (16,12) Pyramid code (Photo credit: Original).

Parity 1: $p_{1k}=d_{1k}+d_{2k}+d_{3k}+d_{4k}+d_{5k}+d_{6k}+d_{7k}+d_{8k}+d_{9k}+d_{10k}+d_{11k}+d_{12,k}$

Parity 2: $p_{2k}=1*d_{1k}+2*d_{2k}+2^2*d_{3k}+2^2*d_{4k}+2^2*d_{5k}+2^2*d_{6k}+2^2*d_{7k}+2^2*d_{8k}+2^2*d_{9k}+2^2*d_{10k}+2^2*d_{11k}+2^{11}*d_{12,k}$

Parity 3: $p_{3k}=1*d_{1k}+4*d_{2k}+4^2*d_{3k}+4^3*d_{4k}+4^4*d_{5k}+4^5*d_{6k}$

Parity 4: $p_{4k}=4^6*d_{7k}+4^7*d_{8k}+4^8*d_{9k}+4^9*d_{10k}+4^{10}*d_{11k}+4^{11}*d_{12,k}$

Parity 5: $p_{5k}=8^3*d_{4k}+8^4*d_{5k}+8^5*d_{6k}+8^6*d_{7k}+8^7*d_{8k}+8^8*d_{9k}$

3.4 Cost control

In Plan A, the greater the Parity overlap, the greater the chance of fixing errors, but this also leads to an increase in cost. Plan B is not an ideal choice for high-cost storage, but it is a great option for low-cost storage. However, in real life, achieving the dual goals of protecting user storage security without delaying transportation and maximizing cost reduction for profit has been an ongoing effort for researchers.

4 Conclusion

This study presents an in-depth exploration of Pyramid Codes, focusing on their fault tolerance capabilities and cost-effectiveness for distributed storage systems. Through a comprehensive analysis, the advantages of Pyramid Codes over traditional Reed-Solomon Codes have been highlighted, specifically in terms of fault-tolerant capacity and repair efficiency. The paper examines the structural features of Pyramid Codes, demonstrating their effectiveness in managing single, double, and triple disk failures without requiring excessive storage overhead. By optimizing the parity length and introducing additional parity layers, the research illustrates methods to enhance Pyramid Codes' repair capabilities, thereby achieving a balance between data resilience and storage cost. The practical applicability of Pyramid Codes in various high-resilience, low-cost scenarios is underscored, with an emphasis on their utility in modern distributed storage systems facing increasing data volumes and reliability demands.

Future research could investigate several key avenues to extend the capabilities and adaptability of Pyramid Codes. One potential direction is the development of adaptive

frameworks that enable real-time data repair, allowing the system to dynamically adjust repair strategies based on the specific nature and location of data loss. Additionally, optimizing parity distribution within Pyramid Codes could help achieve a more balanced trade-off between fault tolerance and cost-efficiency. By exploring these adaptive and optimization-focused enhancements, future studies could further establish Pyramid Codes as a robust and cost-effective solution for resilient data storage in large-scale distributed systems.

References

1. W. Wang, Z. Li, M. Lyu, L. Xu, Y. Xu, Toward efficient repair for wide-stripe erasure coding with high reliability. IEEE Trans. Reliab. (2024)
2. H. Guo, M. Xu, J. Zhang, C. Liu, R. Ranjan, D. Yu, Cheng, BFT-DSN: A Byzantine fault tolerant decentralized storage network. IEEE Trans. Comput. (2024)
3. H. Xu, X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, Research of Pipeline Leak Detection Technology and Application Prospect of Petrochemical Wharf, in 2020 IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC), Vol. **9**, pp. 263-271 (IEEE, 2020)
4. Z. Zhao, X. Zhu, X. Wei, X. Wang, J. Zuo, Application of Workflow Technology in the Integrated Management Platform of Smart Park, in 2021 IEEE 4th Adv. Inf. Manag., Communicates, Electron. Autom. Control Conf. (IMCEC), Vol. **4**, pp. 1433-1437 (IEEE, 2021)
5. V. Rajendran, R.K. Ramasamy, Real-time evaluation of the improved Eagle strategy model in the Internet of Things. Future Internet 16(11), 409 (2024)
6. 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, in 2020 IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC), Vol. 9, pp. 812-816 (IEEE, 2020)
7. J. Wu, the framework of cyber resilience engineering empowered by endogenous security and safety. In Cyber Resilience System Engineering Empowered by Endogenous Security and Safety, pp. 279-330. Springer Nature Singapore, Singapore (2024)
8. 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, in Proc. 2020 8th Int. Conf. Inf. Technol.: IoT Smart City, pp. 227-232 (2020)
9. S. MahmoudZadeh, A. Yazdani, Y. Kalantari, B. Ciftler, F. Aidarus, M.O. Al Kadri, Holistic review of UAV-centric situational awareness: Applications, limitations, and algorithmic challenges. Robotics **13**(8), 117 (2024)
10. 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