

Storage and Data Processing Technology in the Era of Information Explosion: from Traditional to Edge Computing

Longhan Lin *

Advanced Manufacturing Engineering School, Chongqing University of Posts and Telecommunications, Chongqing, China

* Corresponding Author Email: 1227846919@qq.com

Abstract. Data-intensive IoT gradually occupying the market, and efficient and reliable storage algorithms of signals and information are becoming increasingly important. The old-era storage algorithms no longer meet the needs of digital and intelligent cloud storage due to large space usage and low operating efficiency. One of the important topics in IoT cloud computing is to find a turning point from traditional storage methods to develop data-intensive and efficient storage methods. This research analyzed the principles, features, advantages, and disadvantages of many classic storage algorithms, including RS code, EVENODD code, Pyramid code, etc. It is to find storage algorithms suitable with higher storage efficiency in the IoT. Finally, Pyramid code is leading in data recovery efficiency and scalability and is suitable for higher and faster storage needs of the IoT. However, the shortcoming of Pyramid code also has challenges in the large number of failures that occur, due to its slower recovery speed.

Keywords: Pyramid code, Internet of Things, Reed-Solomon code, IoT, Erasure Code, Edge computing.

1. Introduction

With the popularization of the Internet and the acceleration of digital transformation, the information age is facing an unprecedented information explosion. Diverse data needs processing and storage like enterprise data, social media data, sensor data, and personal data. Human society appeared information flood during the Big Data Era. Beyond doubt, the data wrapped in the flood has become to big treasure. They can be used in AI training, user experience improvement, smart city building, and so on. However, trillions of data need to be stored in huge volumes of storage units, so that the data can be visited. Without these storage units, data cannot be transformed into a resource for human development through various data processing technologies. The spirit of these storage units is efficient and reliable storage algorithms. The storage algorithms are the research theme of this article.

Currently, the mainstream storage algorithms on the market are divided into three categories: network storage, P2P storage, and cloud storage. Network storage solution requires building large servers as storage devices, and users upload various files as data for centralized storage and access services [1]. Network storage technology is a typically centralized storage proposal. The advantages are centralization management and fewer user burden. This model provides convenience for remote collaboration and data sharing, but it also brings problems with network latency and data security. The basic architecture of P2P storage (Peer-to-Peer Storage) is a distributed storage proposal. In this distribution system, each peer node can store and share data [2]. Instead of relying on centralized servers, distributed networks are the choice for P2P to store data. This solution approach can improve data availability and redundancy but also needs to address data consistency and security challenges. Cloud storage solutions are where users rent storage services provided by third parties. Users store data on the cloud service provider's servers, allowing users to access and manage their data through the Internet [3]. Cloud storage is more like a user requiring service from a third-party storage service provider. Compared with network storage, users have more control over data; compared with P2P storage, data security and controllability are better. Cloud storage provides a convenient way for data backup, sharing, and access, but data privacy and dependence on cloud vendors need to be considered.

The above storage solutions all have different advantages and disadvantages. To make sure the detailed core algorithm of these storage solutions is the first job of deciding which solution is most suitable for IoT that is the storage algorithm. This article focuses on commonly used fault tolerance techniques. Including traditional fault-tolerant technology, and MDS code pyramid code calculation. Afterward, the research focus shifted to the storage requirements in IoT technology and explored how edge computing changes the storage and processing of IoT. The next section will start with 3 kinds of traditional storage algorithms.

2. Traditional Fault Tolerance Techniques

2.1. Full backup technology

Full back up technology, also known as Comprehensive Backup technology. The specific operation is to directly copy all the data of the file system as a backup. So, when the disk fails or the file is damaged, there will be a complete backup for replacement [4]. This kind of backup directly guarantees the backup in the simplest way integrity. So, it can repair all errors in data files. The advantage of a full backup is that it does not require complex calculations. However, due to full copy, the capacity needs of full backup will be double that of the source. Second, Chervenak et al. came up with slow reading and writing speeds cause big time cost of full backup [4]. Due to the big cost of space and time, full back up technology is hard to use in massive data storage.

2.2. Erasure code technology

Erasure Code expects recovery of complete data from incomplete data. First, the data source will be separated into m data blocks. Then, the encoding function will create c data m data blocks from m source data blocks. The decoding function will allow any C data block to restore the m source data blocks. In general, C must be larger than m [5]. The formula (1) shows the encoding process of erasure code. The " G " in formula (1) is called "Generator matrix", which is used to process the " m "-data to store. The " c " is called codeword, which is stored in storage drives. Formula (2) shows the parity process of the erasure code. The " H^T " is called "Parity Check Matrix", which is used to check the damage of message from " c '", which is codeword after damage. Usually, there will be a method to recover data. Specific principles and functions are depended on.

$$mG = c \quad (1)$$

$$c'H^T = 0 \quad (2)$$

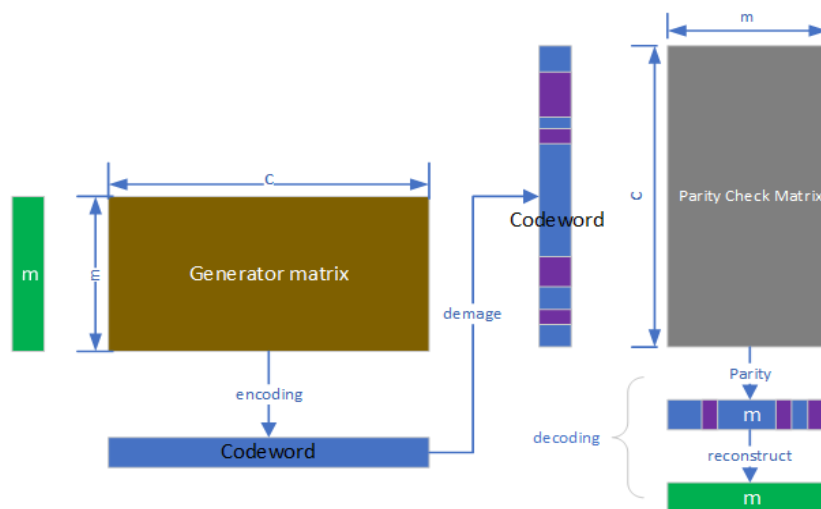


Fig 1. Erasure Code encode and decode principle.

Fig. 1 shows the work of the erasure code clearly. The green block, length m . It is the message to be stored or sent. Generator matrix is an $m \times c$ matrix, used to encode to get codeword. Purple blocks

on codeword mean damage like physical damage, noise pollution, stack overflow leading to packet loss, etc. The parity check matrix will process the damaged codeword and restore the message. Usually, the result of the parity check matrix times codeword shows the location of the error. If there is not any error, the result will be 0, like formula (2). According to the point of parity check matrix, there will be a reconstruction function to get the original data. However, erasure code cannot recover from too many failures, and usually the length of the intact part is shorter than the message length. [5].

Indeed, the invention of erasure code brings another way to store data. However, as message length grows, the generator matrix will grow to an incredible size. Moreover, storage needs more cost to maintain encoding and decoding system work. So, the erasure code is suitable for single communication. It is not a good solution for IoT, which requires huge storage volume and throughput requirements.

2.3. Classic Reed-Solomon Code

Around 1960, the Reed-Solomon code was invented by Irving S. Reed and Gustave Solomon, who were still studying at MIT [6]. The (k, n) Reed-Solomon code uses a special generate matrix as shown in formula (2) as a method to process data.

$$\begin{array}{cccccc}
 1 & 1 & 1 & \dots & 1 \\
 1 & 2 & 3 & \dots & n \\
 1^2 & 2^2 & 3^2 & \dots & n^2 \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & n^{k-1}
 \end{array} \quad (3)$$

The diagram illustrates the Reed-Solomon code encoding and decoding process.
Encoding: A message vector $[0, 2, 1]$ is multiplied by a 3×5 Generator Matrix to produce a 3×5 codeword $[3, 1, 1, 3, 0]$.
Decoding: A received codeword $[3, e, 1, e, 0]$ (where e represents errors) is processed. First, the columns corresponding to the errors are removed, leaving $[3, 1, 0]$. Then, the remaining columns of the generator matrix are used to form a submatrix, which is inverted. Finally, the remaining codeword is multiplied by the inverse of the submatrix to recover the original message $[0, 2, 1]$.

Fig 2. Reed-Solomon code encoding and decoding.

As shown in **Fig. 2**, message “021” times generator matrix to get codeword. To show the restoring process, the codeword lost the second and fourth bits. First, the generator matrix will be removed corresponding columns to fit the remaining codeword. Inverting the generated submatrix will be the next step. Then just multiply the remaining codeword to get the original message.

It should be noted that RS code needs to follow the operation rules of Galois Field instead of ordinary bit operations when encoding, decoding, data repair, and forming the generating matrix [7]. Galois Field is a kind of finite field. Addition and multiplication tables instead of four decimal arithmetic operations to become basic rules of Galois Field. When RS code needs a result of matrix operation, just check the table of Galois Field. The feature increases RS code calculation speed greatly.

RS code can efficiently handle burst errors [8]. The coding efficiency of RS code will grow as code length increases [8]. Because RS code is a symbolic calculation based on the Galois Field finite field algorithm, which is different from the digital calculation of other coding methods [8]. This advantage makes RS code widely used in information transmission. As a message processing

algorithm, RS code has time and space advantages when applied to FPGA and MCU [9]. This is also due to the flexibility of the RS code, which allows it to meet various usage scenarios by adjusting the code rate. However, traditional RS code is only used to process a small amount of one-dimensional information, cannot handle complex two-dimensional data structures, and cannot be applied to large-scale data storage because of the generation matrix.

3. MDS Codes and Array Coding

3.1. Basic principles of MDS code

The codewords of MDS (Maximum Distance Separable) are linear combinations of a fixed set of basis codewords. MDS has parameters $[n, k, n-k+1]$ [10]. Specifically, it turns a symbol sequence of size k into a symbol sequence of size n . The symbols of both n and k come from the same alphabet q . A necessary condition for MDS codes is that the minimum code distance must be less than or equal to $n - k + 1$. The minimum code distance means the smallest non-zero codeword weight between any two different codewords. This singleton bond is the way to distinguish MDS from other erasure codes.

3.2. Array encoding method

EVENODD code is a double-fault tolerant encoding method [11]. As **Fig. 3** shows, the basic structure of EVENODD code is to add two parity disks as redundancy. The structure is like a two-dimensional array which has columns consisting of disks and blocks as elements stored in it.

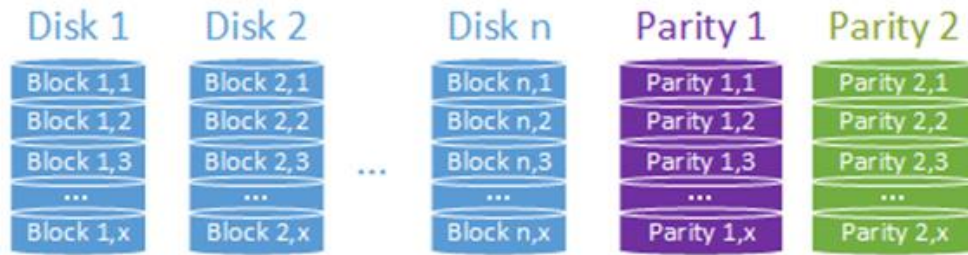


Fig 3. Two or more references.

The rule for Parity 1 disk is shown in formula (3), which is the sum of all blocks in the same row (XOR operation). The operation of Parity 2 is relatively complicated. First, get the diagonal-S use formula (4). Next, formula (5) shows the generate of each block of parity 2. “ m ” is the number of the data disk. “ $m + 1$ ” Means the location of parity 2. “ $\langle x-t \rangle m$ ” means that x is equal to the result number mod m , and the resulting number ranges from 0 to $m-1$.

$$Parity\ 1, x = "Block\ 1, x" XOR "Block\ 2, x" XOR ... "Block\ n, x" \quad (4)$$

$$S = \sum_{t=1}^{n-1} Block\ m - 1 - t, t \quad (5)$$

$$parity\ x, m + 1 = S + \sum_{t=0, t \neq x+1}^{m-1} block\ \langle x - t \rangle m, t \quad (6)$$

Data in parity disk 1 be used to solve one failure problem, according to formula (3). When two data disks fail at the same time, the solution is activating the parity 2 disk and parity 1 and listing the equations grouped according to formulas (5) and (3) to get the missing data.

According to Wang's experiments, program running time will increase as the number of data disks [11]. This is because when recovering the data of any disk, the data of all other disks must be obtained. For IoT, this shortcoming will cause huge read costs for each data recovery.

4. Pyramid Coding Technology

4.1. Introduction to Pyramid Code

Reed-Solomon code has a variant as RAID6. This RS code is a distributed improvement of the classic RS code to adapt to RAID6 mode. Pyramid code is a distributed RAID6 storage method invented based on this RS code.

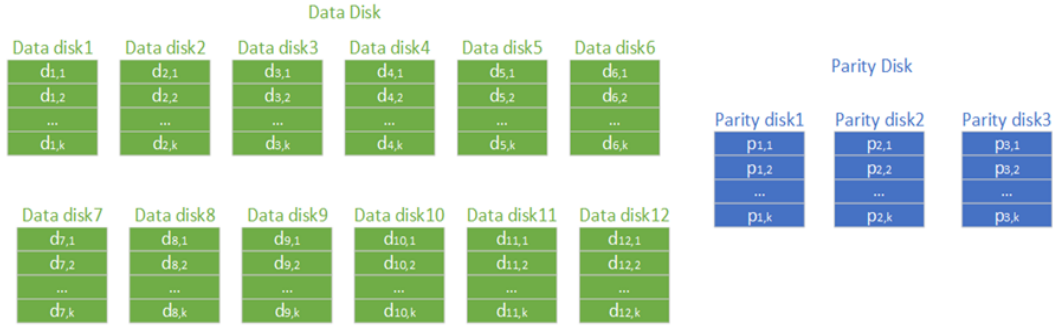


Fig 4. (12, 15) RS code.

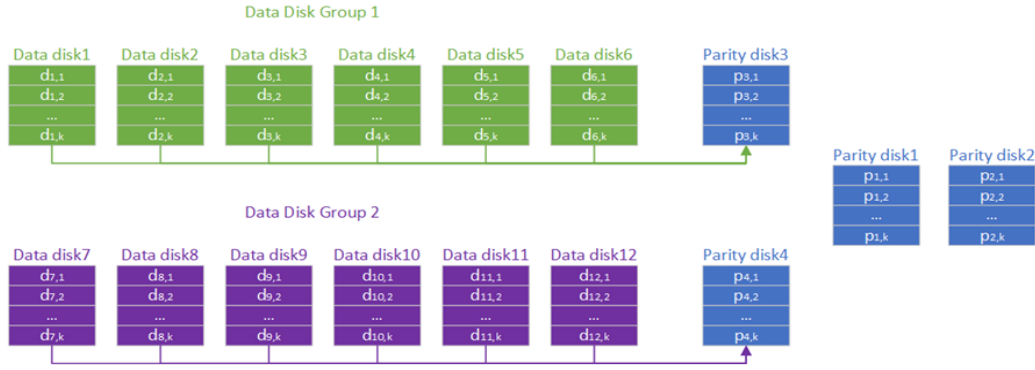


Fig 5. (12, 1) pyramid code.

(16, 12) pyramid code has the same parity 1 disk and parity 2 disk as (15, 12) RS code, as shown in Fig. 4 and Fig. 5. They use the same rule to calculate as formulas (7) and (8) showed. “Parity 1” Means a data block in a parity disk. “data 1” means the block of the data disk corresponding to the row. The factor corresponding to each data block in formula (7) is 1. The factor in formula (8) is “1, 2, 2², 2³”. (16, 12) RS code uses formula (9) to calculate Parity 3. Like formulas (7) and (8), “Parity 3” just uses the corresponding row “data” block to calculate. This is distinct from the EVENODD code which calculates parity across stripes. When an error occurs, Pyramid can focus on the stripe and disk where the error occurred rather than launching all related stripes to fix the error. This is also a necessary condition for Pyramid code to reduce read costs. It also means that all RS code's parity disks contain the information of whole data disks as items. The advantage of this strategy is that the data in any one data disk can exist in 3 different equations. It is determined that when 3 disk failures. However, it also means that no matter how many failures occur, the RS code must activate all disk data. This allows the Pyramid code to improve. (16, 12) Pyramid code has one more parity disk than (15, 12) RS code. The rule of these two parity disks is displayed as formulas (10) and (11).

$$\text{Parity 1} = \text{data 1} + \text{data 2} + \text{data 3} \dots \text{data 12} \quad (7)$$

$$\text{Parity 2} = \text{data 1} + 2 * \text{data 2} + 2^2 * \text{data 3} \dots 2^{11} * \text{data 12} \quad (8)$$

$$\text{RS} - \text{Parity 3} = \text{data 1} + 4 * \text{data 2} + 4^2 * \text{data 3} \dots 4^{11} * \text{data 12} \quad (9)$$

$$\text{Pyramid} - \text{Parity 3} = \text{data 1} + 4 * \text{data 2} + 4^2 * \text{data 3} \dots 4^5 * \text{data 6} \quad (10)$$

$$\text{Pyramid} - \text{Parity 4} = 4^6 * \text{data 7} + 4^7 * \text{data 8} + 4^8 * \text{data 9} \dots 4^{11} * \text{data 12} \quad (11)$$

Formula (10) is the sum of the first six terms in formula (9). Another half of formula (9) is in Parity 4, the additional parity disk of the pyramid code. In other words, the pyramid code splits the data disk corresponding to parity 3 of the RS code into two halves as the last two parities 3 and 4. Here, the pyramid code uses local redundancy instead of the original global redundancy in the RS code [12]. The first 6 data disks are in one group, and the remaining data disks are in another group. If any single disk error occurs, the pyramid code can activate the group of data disks and use the corresponding local redundancy to recover. This allows Pyramid code to perform better when a fewer number of hard drives fail.

4.2. Comparison between pyramid code and RS code

Fig. 6 better explains the advantages of the pyramid code compared to the RS code. As shown in case 1, when data disk 2 failure occurs, data recovery specialists only need to activate Data Disk 1~6 and Parity Disk 3 to repair the failure. When they use the RS code, they need to activate all 16 disks. Case 2 shows how Pyramid Code asynchronously repairs errors on data disks. When failure occurs in both Data Disks 6 and 11. They did not belong same group, data recovery specialists can activate Parity Disks 3 and 4 to repair the two occur at the same time. These two failures can use two equations that do not affect each other to repair. Because they belong to different local redundancies.

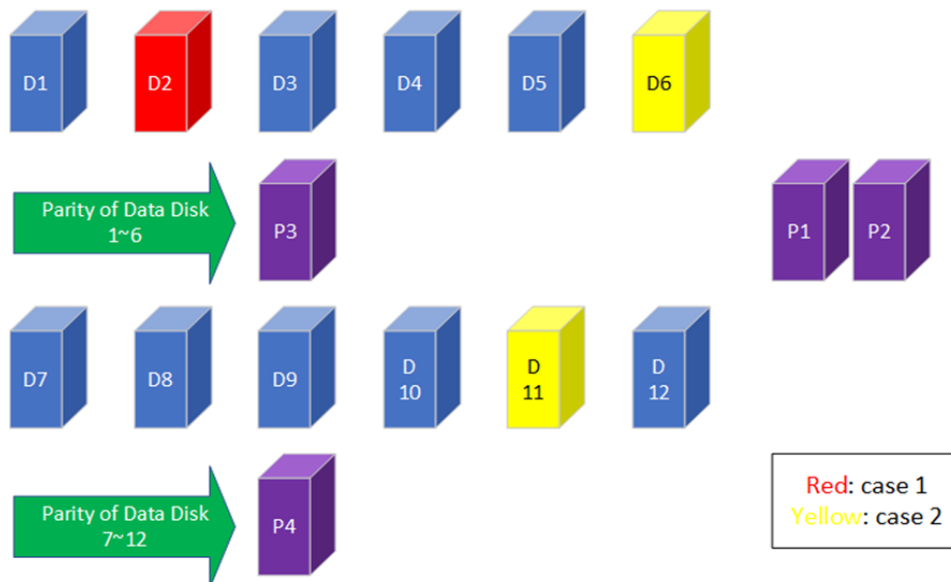


Fig 6. Two cases to describe the advantages of Pyramid Code.

Barbi inputs the incomplete information into three storage model, and check their throughput to compare their repair rate [13]. Vandermonde (14, 10) RS code, Hankel (14, 10) RS code and (15, 10) Pyramid code are three storage models in that experiment [13]. In the experiment, the throughput of the Pyramid code for 1 failure is almost 15 times of the other two RS codes, and when 2 failures occur, the throughput of the Pyramid code can remain more than 5 times of the other two RS codes [13]. However, when failures reach 3, the throughput of the pyramid code is not even as good as Hankel RS code [13]. Therefore, Barbi believes that when failures gradually increase, the efficiency of Pyramid code will slow down [13]. The data shows that when pyramid code fixes the limit of the number of failures that it can fix, it becomes very inefficient. Therefore, Pyramid code is more suitable for scenarios where a small number of errors are fixed.

Compared with RS code, Pyramid code uses an extra Parity disk location in exchange for less read cost [14]. To this end, Huang et al. used (10, 6) pyramid codes to compare (9, 6) Reed-Solomon codes, and the resulting diagram is shown in **Fig. 7**. From this figure, it can be found that the read cost of Pyramid code increases with the increase in errors. When a failure occurs, the read cost of the pyramid code is half that of the Reed-Solomon code. After a failure occurs, the system will repair it immediately instead of waiting for the next failure to occur. Therefore, in most scenarios, Pyramid code will greatly improve repair efficiency and performance. Since cloud storage in reality requires

a lot of disk I/O operations and network transmission, the read cost saved by Pyramid code saves Microsoft Azure Storage millions of dollars [15].

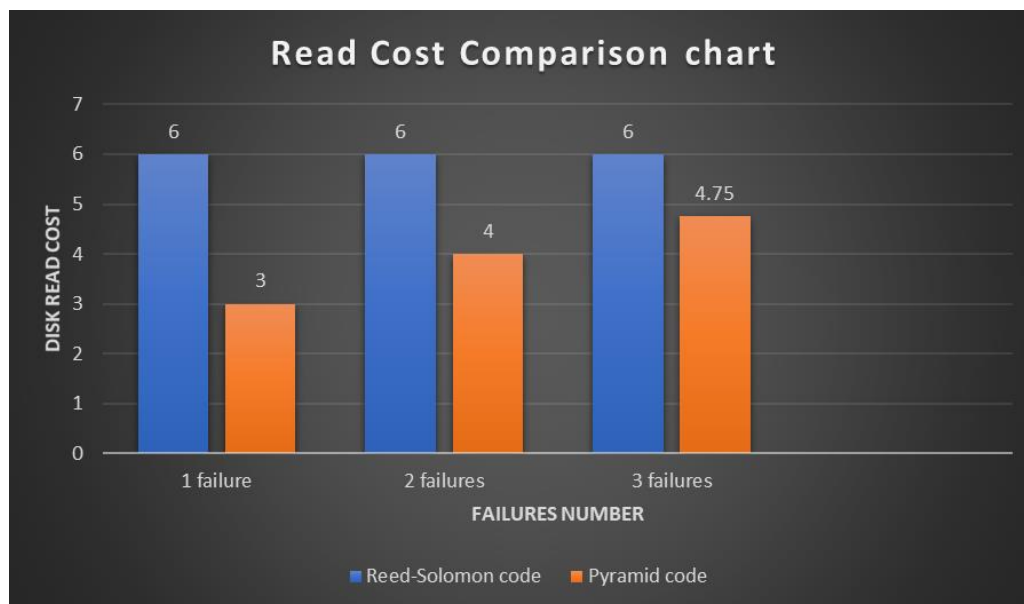


Fig 7. Read cost chart of the RS code and Pyramid code [14].

5. IoT and Storage Requirements

5.1. Internet of Things Technology Overview

The basic idea of Internet of Things (IoT) was first proposed in the 1960s [16]. Today's IoT is continuously integrating basic technologies such as sensors, cloud computing, and actuators [17]. Network technologies such as Bluetooth, ZigBee, and WIFI have also provided technical support to IoT [18]. IoT will develop towards equipment linkage, personalization, and intelligence.

5.2. IoT data processing needs

Big data modal IoT supports modern life in retail, energy, medicine, and so on [19]. In retail, merchants use big data analysis to push products that meet consumer needs [20]. Fast throughput is necessary for rapidly distributing product information and collecting user preferences. From 2021 to 2022, global energy demand has increased by 4.6% [21]. However frequent power outages, load shedding, and environmental leakage have also doubled [22]. For evolution, the storage algorithm needs lower reading costs to recover data due to frequent accidents. IoT in medical systems has the task of preventing accidents, analyzing nutritional status, and supporting postoperative recovery [23]. In the future, it may have the ability to support home rehabilitation operations [24]. The new storage algorithm should not increase redundancy as data increases due to increased daily data from patients. Based on the above discussion, pyramid code is the most suitable storage solution.

6. Application of Storage Technology in IoT

6.1. Data collection and storage

As shown in Fig. 8, the storage algorithm is not only part of the IoT collection and storage of data. The "IoT Data Collection" at the bottom of the figure is usually completed by various sensors. For example, CMOS temperature sensor, LMX20 integrated illumination sensor, and humidity sensor based on QCM (quartz crystal microbalance) [26-28]. These are excellent IoT data collection equipment that has emerged with the development of physics, chemistry, and other fields. The data collected by these devices will be transmitted to the middle-level database along the network. WSN

(Wireless Sensor Network) as a solution has become a core component of IoT [25]. When complex and diverse data enters the database, strengthening data relationships, and organizing and classifying the data is the work that the database needs to complete [29]. When IoT's "Cloud Analytics" requests data from the database, user privacy data will inevitably be touched. Supported by the concept of "never trust always verify", Zero Trust Internet of Things (ZIoT) is becoming a secure data storage option [30]. The final analysis results will be fed back to the underlying equipment to improve user experience, equipment working conditions, etc.

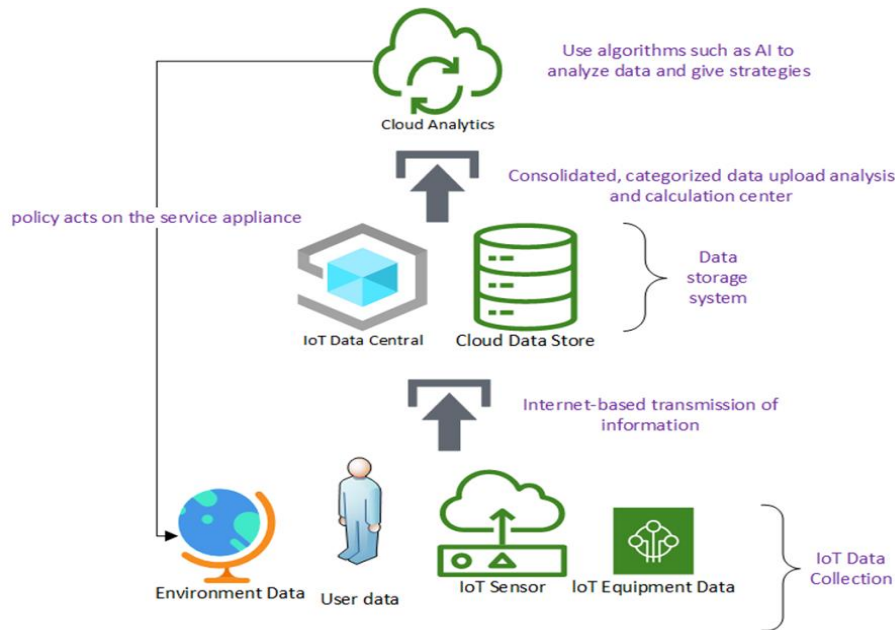


Fig 8. Collecting and storing data in IoT Code.

6.2. Storage and edge computing

With the popularity of the IoT model, more terminals have led to the gradual replacement of central computing by edge computing. Edge computing means computing not only at the central location but also at locations close to the data source or edge devices [31]. Edge computing will reduce the computing center burden and provide better privacy protection for sensitive data. However, there is not much storage space left for edge computing close to the data source. This requires the storage algorithm to have strong scalability and the ability to quickly update data. The pyramid code meets this requirement. Controllable redundancy and adjustable local redundancy allow Pyramid code to flexibly process data in a small space. This is something that traditional RS code cannot do.

7. Conclusion

Pyramid code is the most promising solution to become the backbone storage algorithm of IoT. Due to the huge redundant space and slow recovery speed, full back up is not suitable for IoT solutions in big data mode. Erasure coding requires space several times larger than the data itself to store the "Generator matrix" and "Parity Check Matrix". Erasure coding is more suitable for communication verification than data storage. Traditional RS code also has the same problems as erasure coding. The EVENODD code under the MDS standard already has the prototype of the hard disk storage mode. For large-scale data, EVENODD code can also be processed. However, there is room for improvement in data recovery efficiency. Cross-row verification in the EVENODD code structure cannot reduce read cost. Pyramid code is a storage algorithm with strong scalability, high data recovery efficiency, and low read cost that meets the needs of IoT. However, pyramid code cannot handle bursts of errors well. There is still a long way to go before pyramid codes are applied to IoT.

The development of IoT is inseparable from the need for efficient storage technology. The evolving IoT will prioritize scalability, real-time processing, enhanced security at the edge, and

seamless integration of cloud storage. Addressing data privacy concerns, promoting data interoperability, and promoting energy-efficient storage solutions are critical to shaping the future of storage technology in these dynamic areas. As the information age develops, these innovations will continue to play a vital role in supporting data-driven aspects.

References

- [1] Peng, Ying, Yan Zhu, Jinling Luo. A network storage framework based on SAN. 2012 7th International Conference on Computer Science & Education (ICCSE), 2012, 7th: 818-821. 2012.
- [2] Toka, Laszlo, Matteo Dell'Amico, Pietro Michiardi. Data transfer scheduling for p2p storage. 2011 IEEE International Conference on Peer-to-Peer Computing, 2011: 132-141. 2011.
- [3] Wu, Jiyi, Lingdi Ping, Xiaoping Ge, Ya Wang, Jianqing Fu. Cloud storage as the infrastructure of cloud computing. 2010 International conference on intelligent computing and cognitive informatics, 2010: 380-381. 2010.
- [4] Chervenak, Ann, Vivekenand Vellanki, Zachary Kurmas. Protecting file systems: A survey of backup techniques. Joint NASA and IEEE Mass Storage Conference, Vol. 99. 1998.
- [5] Rizzo, Luigi. Effective erasure codes for reliable computer communication protocols. ACM SIGCOMM computer communication review, 1997, 27(2): 24-36.
- [6] Reed, Irving S., Gustave Solomon, polynomial codes over certain finite fields. Journal of the society for industrial and applied Mathematics, 1960,8(2): 300-304
- [7] Lone, Faisal Rasheed, Arjun Puri, Sudesh Kumar. Performance comparison of Reed Solomon code and BCH code over Rayleigh fading channel. arXiv preprint arXiv:1307.6930, 2013.
- [8] Sklar, Bernard. "Reed-solomon codes." Downloaded from URL <http://www.informit.com/content/images/art.sub.--sklar7.sub.--reed-solomon/elementLinks/art.sub.--sklar7.sub.--reed-solomon.pdf> (2001): 1-33.
- [9] Bianchi, Valentina, Marco Bassoli, Ilaria De Munari. Comparison of FPGA and microcontroller implementations of an innovative method for error magnitude evaluation in reed-solomon codes. Electronics, 2020, 9(1): 89.
- [10] Sun, Zhonghua, Cunsheng Ding. The extended codes of a family of reversible MDS cyclic codes. arXiv preprint arXiv:2307.04076, 2023.
- [11] Wang, Yihan. EVENODD Code Implementation and Generalizations. 2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE), 2022: 64-68. 2022.
- [12] Wang, Zihao, Zheng Xie, Dan Tang. An Erasure Code with Low Recovery-overhead Based on a Particular Three-hierarchical Redundancy Structure. International Journal of Network Security, 2022, 24(5):965-974.
- [13] Barbi, Roberta. Pyramid Codes to the rescue: A preliminary evaluation for cloud storage systems. Proceedings of the Doctoral Symposium of the 17th International Middleware Conference, 2016: 1-2.
- [14] Huang, Cheng, Minghua Chen, Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. ACM Transactions on Storage (TOS), 2013, 9(1): 1-28.
- [15] Huang, Cheng, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, Sergey Yekhanin. Erasure Coding in Windows Azure Storage, 2012 USENIX Annual Technical Conference (USENIX ATC 12), 2012: 15-26.
- [16] Tawalbeh, Lo'ai, Fadi Muheidat, Mais Tawalbeh, Muhannad Quwaider. IoT Privacy and security: Challenges and solutions. Applied Sciences, 2020,10(12): 4102.
- [17] Grammatikis, Panagiotis I. Radoglou, Panagiotis G. Sarigiannidis, Ioannis D. Moscholios. Securing the Internet of Things: challenges, threats and solutions, Internet of Things, 2018, 5: 41-70.
- [18] Lee, Changmin, Luca Zappaterra, Kwanghee Choi, Hyeong-Ah Choi. Securing smart home: Technologies, security challenges, and security requirements, 2014 IEEE Conference on Communications and Network Security, 2014, 67-72.

- [19] Mahbub, Mobasshir, M. Mofazzal Hossain, Md Shamrat Apu Gazi. IoT-Cognizant cloud-assisted energy efficient embedded system for indoor intelligent lighting, air quality monitoring, and ventilation. *Internet of things*, 2020, 11: 100266.
- [20] Ma, Benedict Jun, Yi Zhang, Shuai Liu, Yishuo Jiang, Yifan He, Ke Yan. Operational strategies for IoT-enabled Brick-and-Mortar retailers in a competitive market, *Computers & Industrial Engineering*, 2022, 173: 108665.
- [21] U.S. Energy Information Administration. (2021). Annual Energy Outlook 2021-With Porjections to 2050. Accessed: Jun. 30, 2021. [Online]. Available: <https://www.eia.gov/outlooks/aeo/>
- [22] Saleem, M. Usman, M. Rehan Usman, M. Arslan Usman, Christos Politis. Design, deployment and performance evaluation of an IoT based smart energy management system for demand side management in smart grid. *IEEE Access* 10, 2022: 15261-15278.
- [23] Nižetić, Sandro, Petar Šolić, Diego Lopez-de-Ipiña Gonzalez-De, Luigi Patrono. Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of cleaner production*, 2020, 274: 122877.
- [24] Bisio, Igor, Chiara Garibotto, Mehrnaz Hamedani, Fabio Lavagetto, Valeria Prada, Andrea Sciarrone. Towards IoT-based eHealth services: A smart prototype system for home rehabilitation. 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019.
- [25] Gulati, Kamal, Raja Sarath Kumar Boddu, Dhiraj Kapila, Sunil L. Bangare, Neeraj Chandnani, G. Saravanan. A review paper on wireless sensor network techniques in Internet of Things (IoT). *Materials Today: Proceedings*, 2022, 51: 161-165.
- [26] Zambrano, Benjamin, Esteban Garzón, Sebastiano Strangio, Felice Crupi, Marco Lanuzza. A 0.05 mm², 350 mV, 14 nW Fully-Integrated Temperature Sensor in 180-nm CMOS. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021, 69(3): 749-753.
- [27] Ahmad, M. D., SZ Mohammad Noor, NF Abdul Rahman, Faranadia Abdul Haris. Lux meter integrated with internet of things (IoT) and data storage (LMX20). 2021 IEEE International Conference in Power Engineering Application (ICPEA). 2021.
- [28] Chang, Qijie, Daofu Wu, Yanyi Huang, Chengyao Liang, Libo Liu, Huanbin Liu, Yan Liu, Jing Qiu, Xiaosheng Tang, Genquan Han. Ultrahigh Response Humidity Sensor Based on Lead-Free Cs₂SnCl₆ Perovskite Films. *IEEE Electron Device Letters*, 2022, 43(5): 805-808.
- [29] Nasar, Mohammad, and Mohammad Abu Kausar. Suitability of influxdb database for iot applications. *International Journal of Innovative Technology and Exploring Engineering*, 2019, 8(10): 1850-1857.
- [30] Wang, Jin, Jiahao Chen, Neal Xiong, Osama Alfarraj, Amr Tolba, Yongjun Ren. S-BDS: An effective blockchain-based data storage scheme in zero-trust IoT. *ACM Transactions on Internet Technology*, 2023, 23(3): 1-23.
- [31] Li, Junxia, Jinjin Cai, Fazlullah Khan, Ateeq Ur Rehman, Venki Balasubramaniam, Jiangfeng Sun, P. Venu. A secured framework for sdn-based edge computing in IOT-enabled healthcare system. *IEEE Access*, 2020, 8: 135479-135490.