Array Erasure Codes with Preset Fault Tolerance Capability

Dan Tang, Ya-Qiang Wang, Hao-Peng Yang
(Corresponding author: Dan Tang)

Software Engineering College & Chengdu University of Information Technology
No.24 Block 1, Xuefu Road, Chengdu 610225, China
(Email: tangdan@foxmail.com)
(Received Jan. 3, 2017; revised and accepted Mar. 11, 2017)

Abstract

The array erasure code, an ideal method for fault tolerance in storage systems, however, is obstructed by its impossibility to set the fault tolerant ability according to dynamic application environment for practical purpose. In view of this, this paper presents a new class of array erasure codes, with the greatest contribution to the array codes which can be obtained according to the preset number of fault tolerance and storage efficiency in dynamic structure, and fault tolerance capability can be presented is not limited in theory. In addition, the new array code has the advantages of simple structure, easy to realize, with no strong constraints to satisfy in structure. Only binary XOR operations are required for coding and decoding for a high operational efficiency, and fixed update penalty and repair cost which will not increase with the expansion of system size or increase of fault tolerance capability.

Keywords: Array Erasure Codes; Preset Fault Tolerance; Strip Size; Weak Constraints

1 Introduction

At present, we have entered into the era of big data, with all industry data explosively growing. More and more data have become an important part of the normal operation of society. In order to deal with the data storage reliability problem caused by rapid growth of data quantity, increasing the stability of a single storage node is certainly a kind of theoretically feasible method. But more effective approach is to use multiple storage nodes to build a storage system, which can make full use of existing equipments, increase storage capacity and improve data access efficiency in parallel, and with a certain fault tolerance strategy, can also effectively enhance the reliability of the whole storage system. In this case, the number of storage nodes is usually used to represent the size of the storage system. At present, large storage systems with more than 100 nodes have become more and more popular, Google and other companies even have some PB level storage systems with more than 3,000 nodes[8] . Compared with the past, although the reliability of a single storage node has been enhanced, in a large storage system with a great amount number of nodes, the probability of failure of multiple nodes in a period is still very high. According to statistics from the Carnegie Mellon University, annual failure and replacement rate of a large storage system with more than 300 nodes is about 5.1% [9]. Of course, this is the probability of failure of nodes in the storage system under normal operation conditions, and if floods, landslides, earthquakes and other natural disasters and administrators misoperation, hacker attacks and other human factors are taken into consideration, fault tolerance and storage system reliability enhancement is more a concern.

Replication technology is the most mature fault tolerance technology in the field of reliability enhancement of the storage system. A fault tolerance scheme depending on replication technology has multiple copies of the same data stored in different nodes of the system. As copies of each node are exactly the same, usually it is not required to strictly distinguish between the parity data (redundant data) and the original data. When a node fails, a copy in any node which is not failed can recover the missing data. The fault tolerance method depending on replication technology is simple and intuitive, easy to realize and dynamically expandable, but the low storage efficiency is a huge defect. Assuming replication technology is used to construct a storage system of t fault tolerance, it needs to copy the original data into a $t+1$ copies to be stored in different nodes, that is, the storage efficiency is only $1/(t+1)$. Especially for large storage systems, this extremely low effective utilization of storage space is very difficult to accept.

Storage system as a whole shall consider not only a certain performance parameter, but the most important performance indicators according to the application environment. Erasure codes can balance all main perfor-

mances of storage system to a certain extent, and it is a kind of fault tolerance method which is more and more important. At present, the RS erasure code is the most widely used in the storage system. The relevant mathematical theory of RS codes is mature, with regular codeword structure and unlimited fault tolerance capability in theory, and the codes have the MDS property, satisfying the theoretically optimal code rate (corresponding to the storage efficiency in a storage system). However, RS codes coding and decoding are performed on a multivariate finite field by complex operations, especially for multiplication and inversion in a finite field. Therefore, the main problem to be solved by fault tolerance method based on RS codes for storage system is not to optimize the coding process, but how to improve the operational efficiency over the finite field. Of course, there are also some scholars who have made a very fruitful work on this issue, increasing operational efficiency over a finite field greatly, and one of the most representative is the reduction computing program for a finite filed[7] and the finite field calculation scheme GF-Complete[6] by Plank et al. In spite of this, the current storage system is still difficult to bear the cost of RS code operations, especially the large storage system.

Array erasure code (usually referred to as array code) is a kind of erasure codes using binary XOR coding and decoding operation, with high operational efficiency. The special two-dimensional coding structure is seemingly complex but can be used completely corresponding to two-dimensional data layout structure normally used in the current storage system of multiple nodes, so it is suitable for use in the storage system. However, in addition to the small size centralized RAID6 system, the EVENODD code[1] is chosen as one of the alternative methods for tolerating 2 failures, few array codes are used in the current commercial storage system. Such a situation can be attributed to the fixed fault tolerance capability of most array codes which are not easy to expand. Such as EVENODD codes[1], X codes[12] are 2 fault tolerance array codes, as long as the constraints of these coding structure are satisfied and in accordance with the coding method, it can make the node fault tolerance of storage system at 2, but only 2. In other words, as long as the use of EVENODD codes or X codes as a fault tolerance method, regardless of the size of a storage system, the maximum fault tolerance is only 2. Similarly, with the use of Star codes[3] and extended X codes[5] and other 3 fault tolerance array codes as a fault tolerance method, the maximum fault tolerance of the storage system is also fixed to 3. Grid codes[4] use two array codes with typical horizontal or vertical data layout as the matched codes, which can be used to obtain high fault tolerance with coding in the horizontal and vertical directions simultaneously. But once two matched codes are determined, the fault tolerance capability of the Grid codes is also determined. Weaver codes[2] can determine the codeword structure according to the requirement of fault tolerance capability, with maximum fault tolerance up to 12, but the fault tolerance capability of the codes is lack of theoretical support, that is, the fault tolerance capability is obtained by computer check. In addition, the storage efficiency of the codes is always lower than 50%, and will quickly decline with the increase of fault tolerance. Reference[10] proposes a new way to reversely determine the structure of coding by fault tolerance, but specific implementation method is not provided.

In view of above problems, this paper presents a new class of array erasure codes, which use the horizontal data layout. All calculations in the new array codes completely are binary field XOR, with high operational efficiency. The repair cost and update penalty are fixed constants, which will not increase with the expansion of storage system size. The codes can be constructed according to the fault tolerance requirements for running storage system, with fault tolerance not restricted in theory, but also the storage efficiency can be set in a particular fault tolerance capability. After a small transformation, the new codes can also be used in areas other than storage[11]. All of these properties make the new array codes already available on the basis of practical application in large storage systems.

2 Array Codes Capable of Presetting Fault Tolerance

Some basic concepts in the field of storage coding, such as data, check, parity, redundancy, element, strip, stripe, horizontal array codes, vertical array codes, coding, decoding, data reconstruction, can refer to literatures[1, 12, 3, 5, 2] for definitions. This paper will continue to use the definitions of these concepts which will not be repeated. The new array code proposed in the paper uses the horizontal data layout structure, so the entire storage array can be divided into data element array and check element array in logic. A column in the storage array corresponds to a storage node, and in the horizontal data structure each column or all shall be data elements, or check elements. A node failure shall mean that all elements the node corresponds to become unknown. For convenience in description, this paper defines the following symbols. In case of no special note afterward, the meaning of these symbols is used here. It is agreed that f is the fault tolerance of the proposed array code to be constructed, n for the number of columns of data element array, r for the number of columns of check element array, m for the number of rows of storage array. Obviously f, m, n, r are positive integers.

2.1 The $m = 2$ Case

For the f fault tolerance array codes, at least f groups is required with n linear independent check equations in each group, that is, the deployment of f group coding chains with different slopes. This paper will use the positive and negative expansion of numbers to determine

the different slopes of deployed coding chains, that is, all slopes of coding chains are taken from the set {1, -1, 2, -2,... }. Actually, the nature of the coding chain is the linear relationship between data elements and check elements. The specific concepts and definitions of the coding chain and its slope shall refer to the literature[10], which will not be repeated hereof. When $f = 1$, it only needs to deploy 1 group of coding chains. In a storage array of $n = 2$, a group of coding chains with slope of 1 is deployed, as shown in Figure 1, in which the XOR sum of elements with the same background color is 0. Of course, the relationship between data elements and check elements can also be expressed by linear equations, and the check relationship among various elements in the storage array in Figure 1 can be expressed by the equation group (1).

$$\begin{cases} d(1,1) + d(2,2) = c(1) \\ d(1,2) + d(2,1) = c(2) \end{cases} \tag{1}$$

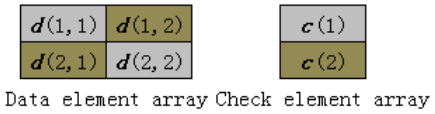

Figure 1: Relationship among elements with $f = 1$

In Figure 1, if any one of columns in the storage array fails, in each equation in the equation group (1) one element will become unknown. Obviously, the unknown element can be obtained by the other two known elements in XOR equation. Therefore, any one invalid column in the array can be effectively recovered.

When fault tolerance is required to reach 2, then 2 sets of coding chains with different slopes shall be deployed. In a storage array with $n = 4$, two sets of coding chains with slopes at 1 and -1, respectively, as shown in Figure 2, are deployed. As to why the n must be equal to or greater than 4, answers will be given below. Figure 2 (a) and (b) show the deployment of 2 coding chains with slopes of 1 and -1 respectively, in which the XOR sum of elements with the same background color is 0. Of course, the check relationship can also be expressed by equation group (2).

$$\begin{cases} d(1,1) + d(2,2) = c(1); d(1,2) + d(2,3) = c(2) \\ d(1,3) + d(2,4) = c(3); d(1,4) + d(2,1) = c(4) \\ d(1,1) + d(2,4) = c(5); d(1,2) + d(2,1) = c(6) \\ d(1,3) + d(2,2) = c(7); d(1,4) + d(2,3) = c(8) \end{cases} \tag{2}$$

After any 2 columns in storage array shown in Figure 2 fail, it can always find at least 2 unknown elements in an equation with only 1 unknown element, and the element values could be known by XOR. After this, if there are other unknown elements, then all exist in the equation of only 1 unknown element, and can be easily recovered.

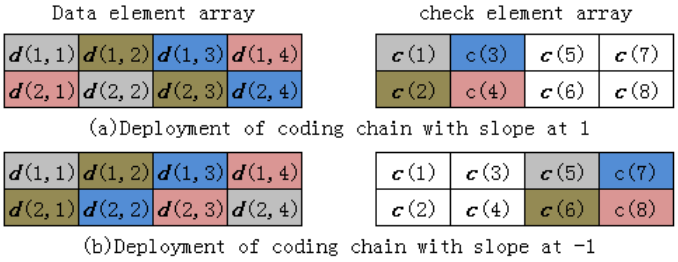

Figure 2: Relationship among elements with $f = 2$

2.2 General Coding and Data Reconstruction Method

In a storage array, $d(i, j)$ is used to represent the element at row i and column j in the data element array, and $c(t)$ as the t check element in the check element array. Among them check elements are arranged at priority, where i, j and t are positive integers, and $1 \le i \le m$, $1 \le j \le n$, $1 \le t \le m \cdot r$. The element identification on the storage array is shown in Figure 3.

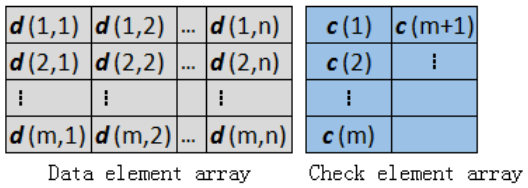

Figure 3: Element ID in storage array

In the identification system of above data elements and redundant elements, each check element may be calculated by expression (3) .

$$c(t) = \sum_{i=1}^{m} d(i, l_c + i \cdot (2 \cdot (l_r \% 2) - 1) \cdot \lceil l_r/2 \rceil - 1) \% n + 1). \tag{3}$$

Where, l_r represents the l_r coding chain used in the present check element, l_c represents that the slope of coding chain which is used for the l_c time, which can be obtained by expression (4), in which, the operator "[]" represents rounding while " %" modulus.

$$\begin{cases} l_r = [(t-1)/n] + 1 \\ l_c = (t-1) \% n + 1 \end{cases} \tag{4}$$

Example 1. *Maximum fault tolerance capability of storage system as 3, constructing the corresponding array code.*

It is assumed that the storage array strip size $t = 3$, and 7 columns of data element array in the storage array. As previously stated, the maximum fault tolerance capability f is 3, the size of data element array is 3×7, the number of check elements is 21, so the number of columns in the check element array is also 7. The maximum fault tolerance capability is 3, so coding chains with slopes of 1, -1 and 2 are deployed. Specific deployment process as

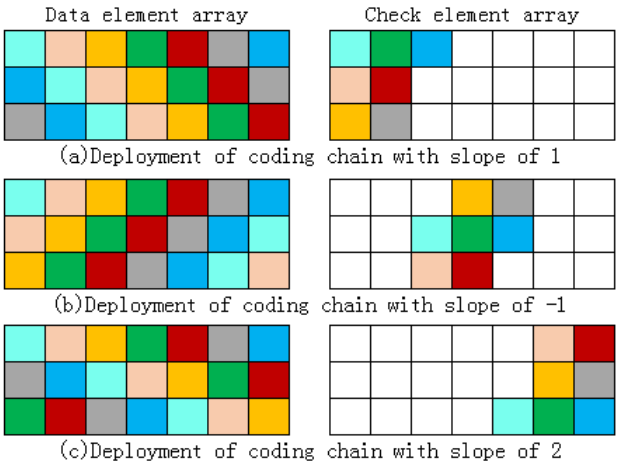

Figure 4: Relationship among elements with $f = 3$

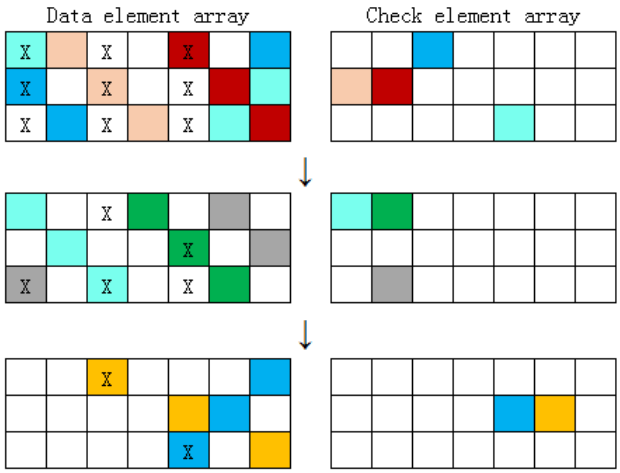

Figure 5: Successful recovery of failure node data

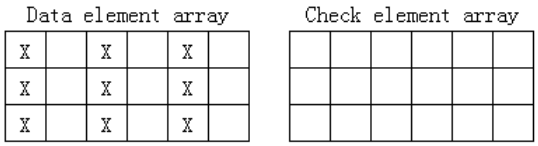

Figure 6: Failed failure node data recovery

shown in Figure 4, XOR sum of elements with the same background color is 0.

We can use the method of this paper to construct a specific fault tolerance array code according to the requirement of application environment. And the next will be a brief introduction of how to restore and reconstruct after the failure of nodes. Again, this paper only considers the case of node error, that is, the entire storage node failure causes the loss of all data elements on the node, which also corresponds to the column failure in the storage array. The basic idea of data recovery on the failure node can be summed up as, that is, to find the coding chains with only one failure element, obviously the failure element on the coding chain can be calculated by other valid elements. Continue to repeat this process until all the invalid elements are fully recovered.

Example 2. *It is assumed that the nodes 1, 3 and 5 in the storage system fail, with all failure nodes replaced and renewed, the elements at 1, 3 and 5 in the data element array in storage array become unknown. The whole data recovery process is shown in Figure 5. Elements marked with "X" represent invalid element with unknown value, and elements with the same background color shall be a coding chain with only 1 failure element, which could be recovered by XOR of other elements on the coding chain.*

Continue to extend the example. It is assumed that the size of data element array is 3×6, and elements on 1, 3, and 5 columns fail. However, as shown in Figure 6, it is impossible to find any coding chain that contains only a failure element. In this case, all data in the storage system are lost.

2.3 Constraint Conditions and Fault Tolerance Capability Guarantee

From the example 2 in the previous section it can be concluded that, if you need to grant the storage array of strip size 3 with fault tolerance of 3, in addition to the deployment of 3 coding chains of different slopes, the number of columns of data element array shall be at least not less than 7, otherwise fault tolerance is not guaranteed. By further expansion of this conclusion, the use of f coding chains with different slopes to construct array code cannot guarantee the fault tolerance f, the number of fault tolerance and array size also need to meet a certain conditions. According to the data reconstruction method of the previous section, to ensure that at least 1 failure element in one or more coding chains with only 1 failure element, then the fault tolerance f, strip size m and the number of columns of data element array n shall meet the conditions:$n \geq m \cdot f - f + 1$. The fault tolerance capability can be guaranteed in the case of satisfying the constraints as proved below.

For array codes of horizontal layout, the occurrence of node level failure can be divided into the following three types: 1. All failure nodes in the check element array; 2. Failure nodes in check element array and data element array; 3. All failure nodes in the data element array. When all of the failure nodes are in the check element array, it just needs to re-do a coding operation after the replacement of the failure nodes. This is the simplest case of data reconstruction. When the data element array and the check element array have failure nodes, it is relatively easy to deal with. In the array code with f fault tolerance, each data element is passed by f coding chains, so a data element is related to f check elements. According to the constraint conditions it can be known that, so two check elements associated with the same data element must not appear in the same column. As a result, all of the failure elements in data element array can be recovered completely, and then all the failure elements in the

check element array can be recovered by coding. And for all of the f failure nodes in the data element array, the following mathematical induction is used to prove.

First of all, we mark the f failure columns in data element array as $E_1, E_2 \cdots, E_f$, where d_i shall be the distance between the failure column E_i and the failure column on its right. It is assumed that $\max(d_1, d_2 \cdots, d_f) = d_f$, where $i = 1, 2 \cdots, f$. Therefore, by the principle of the pigeon cage, it is known that $d_f \geq m$ is established. At that time, when there was only one failure column, the column will be recorded as E_1. By the coding chain deployment method it can be known, each data element is passed by one coding chain, apparently all the failure elements on the column can be successfully restored. Assuming that all the failure elements can be recovered when $f = k$, where k is a positive integer. In the following, a discussion on $f = k + 1$ is conducted. Be known by the precondition, $\max(d_1, d_2 \cdots, d_k) = d_k \geq m$. Therefore, if the inequality $d_1 \geq \lceil m/2 \rceil$ is established, it is clear that all elements on the first failure column E_1 can be recovered by the coding chains with slopes of 1 or -1. In the same way, if $d_k \geq \lceil m/2 \rceil$ is established, all the elements on the final failure column E_{k+1} can be recovered by coding chains with slopes of 1 or -1. At this point, the above two situations can be changed into $f = k$, and according to the aforementioned assumptions, all the failure data can be recovered. When $d_1 < \lceil m/2 \rceil$ and $d_k < \lceil m/2 \rceil$, the first failure element in all of the failure columns can be recovered by coding chains with slopes of $\pm 1, \pm 2, \cdots, \pm k, k+1$. And when all the failure elements in the first row on all failure columns are recovered successfully, only to repeat the same steps, the remaining failure elements can be effectively restored. Quod erat demonstrandum.

The inequality $n \geq m \cdot f - f + 1$ is the constraint condition that is needed to satisfy when we are constructing the specific fault tolerance capability of array codes in this paper.

2.4 Satisfaction of Expected Storage Efficiency

Storage efficiency is an important performance index to measure the effective utilization of storage space. As mentioned earlier, in the storage system reliability enhancement field, replication technology is the method of fault tolerance without any operations, and the principle is simple and easy to implement. It is usually not recommended the use of replication technology in large storage systems in consideration of storage efficiency. Compared with that, the fault tolerance system based on erasure codes can greatly improve the storage efficiency under the condition of the same fault tolerance capability. Of course, in this process, the calculations are required. For array codes, as mentioned before, with the storage efficiency as the standard, it can be divided into MDS codes and non MDS codes. MDS codes have a theoretical optimal value in storage efficiency, and its typical representation includes EVENODD codes[1], X codes [12], Star codes[3] and extended X codes[5] and so on. But the fault tolerance capability of array codes with MDS property is only 2 or 3, which obviously cannot meet the demand of modern large storage system reliability enhancement. In order to improve the fault tolerance of array codes, researchers have designed some array codes without the MDS property, with fault tolerance capability greatly improved compared with array code with MDS property, but mostly with great sacrifice in storage efficiency, such as the Weaver codes with fault tolerance at 10, the storage efficiency is less than 20%.

From the last section, we can know that the total number of data elements in the array erasure codes generated by this method is $m \cdot n$, and check elements $f \cdot n$. Obviously, the storage efficiency of array erasure codes constructed by the new method is $m \cdot n/(m \cdot n + f \cdot n) = m/(m + f)$. Therefore, the factor affecting the storage efficiency in specific fault tolerance is the strip size m. Figure 7 shows the storage efficiency varies with the size of strip size with the fault tolerance at 20, 30, 50 and 100. Obviously, with fault tolerance capability unchanged, increasing strip size can effectively improve the storage efficiency. Thus, under a certain fault tolerance, of course, it can also set the storage efficiency.

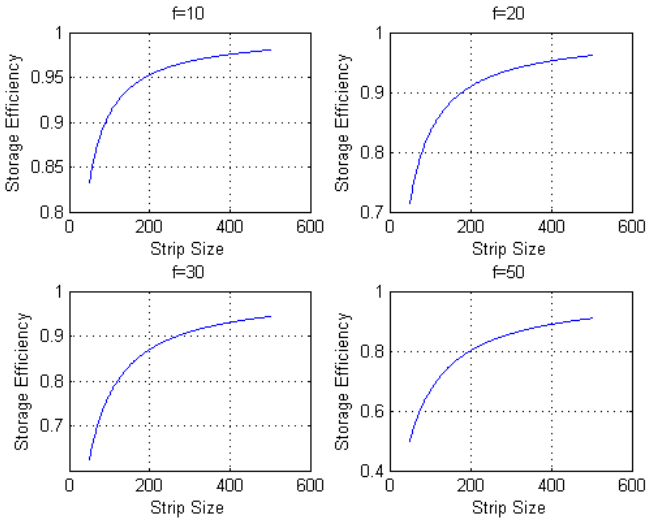

Figure 7: Influence of strip size on storage efficiency

Example 3. *Fault tolerance capability is 4, and storage efficiency is not less than 80%, constructing the corresponding array code.*

According to the previous description of array codes construction method, the storage efficiency can be expressed as $m \cdot n/(m \cdot n + f \cdot n) = m/(m + f)$, then the inequality $m/(m + f) \geq 0.8$ needs to be established. Then the inequality $m \geq 4 \cdot f$ holds, and from the preconditions it is known that when $f = 4$, $m \geq 16$. In this example $m = 16$, then it is inferred that $n \geq 61$ when $n \geq m \cdot f - f + 1$. In this example, $n = 61$, we can know the total number of check elements $f \times n = 4 \times 61 = 244$, and it is inferred that the check node number is $\lceil 244/16 \rceil = 16$.

And then, the basic information is known throughout the storage array. The array has 16 rows, namely the strip size is 16. The data element array has 61 columns, check element array has 16 columns, a total of 976 data elements, and 244 check elements, with storage efficiency of exactly 80%. According to the proof of the previous section, respectively with the 4 coding chains with slope of $\pm 1, \pm 2$ deployed, the fault tolerance capacity of the array code is up to 4.

3 Experiments and Analysis

3.1 Operational Burden

Because most of the code word structures of array codes are irregular and it is difficult to use a precise formula to express the overall workload of coding or data reconstruction. Usually the times of XOR calculations of generating a check bit is used to measure the complexity of coding, and the times of XOR calculations to recover a lost data element for the complexity of data recovery. It can be known from the preceding text that the length of each coding chain is $m+1$, so that the operational effort required to generate a single check element or to recover a single failure element is $m-1$. However, stripe size grows with the enlargement of strip size m or fault tolerance f, which means that the number of check elements will also increase, so does the overall operational burden. Therefore, changes in the strip size and fault tolerance would affect coding and data recovery calculations, as one of the factors we need to consider. Assuming that the strip size is 200, fault tolerance 50, with the storage of 1G data as an example, XOR calculations needed to generate all check elements or recover failure data on 50 nodes is about 4.2×10^9, and it takes a general personal computer with dominant frequency of 3.2G 16s to complete. Figure 8 shows the influence of strip size increasing on array code coding and data recovery operation and Figure 9 the influence of fault tolerance on coding and data recovery operational burden.

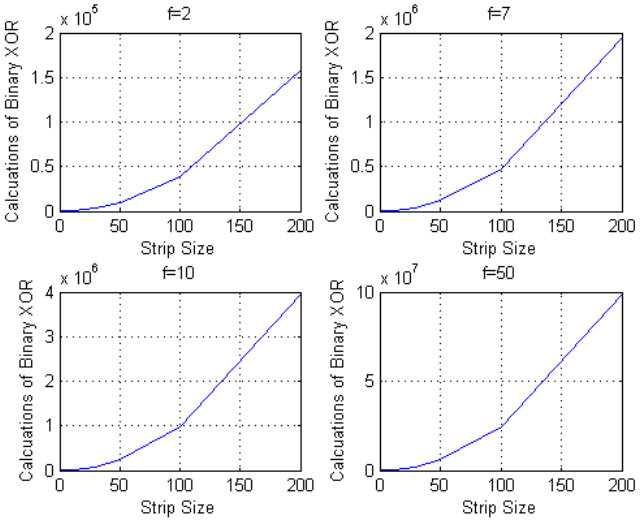

Figure 8: Influence of strip size on operational burden

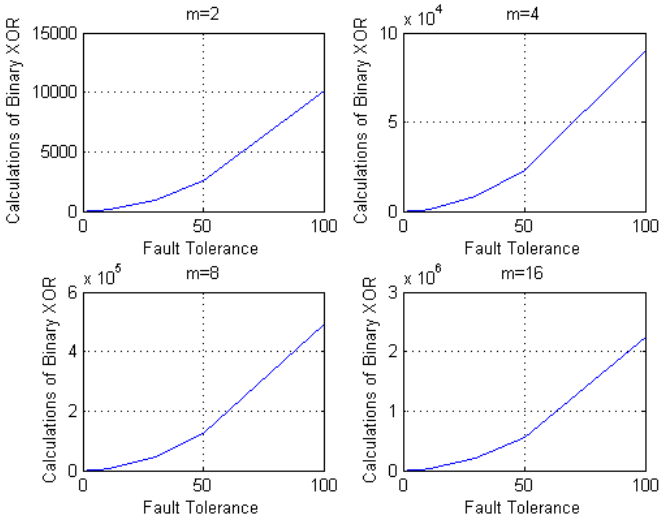

Figure 9: Influence of fault tolerance on operational burden

3.2 Constraint Conditions of Array Code Construction

Most of the array erasure codes in the construction process will restrict the storage array size namely stripe size or strip size. Satisfying the constraints is the basic condition of coding to reach a certain fault tolerance capability. Table 1 shows the comparison of several kinds of array codes during construction with constraints on stripe size and strip size, where p represents a prime number, s_r stripe size, s_c strip size, obviouslys_r, s_c are positive integers greater than 1, which will no longer be listed in the table. From this table it is not difficult to find that most array erasure codes have very strict constraints on stripe size or strip size, and usually requires stripe size as prime or satisfies a linear relationship with a prime. Constraints on the stripe or strip size will greatly limit the application and extension of array codes. In the practical process of array codes, in order to deal with the situation mentioned above, researchers have proposed a number of compromise solutions, the most typical approach is when the number of nodes in storage array is not a prime, add 1 or more virtual nodes with storage data values as 0 to complement the prime. In the process of coding or data recovery these virtual nodes participate in operations. However, with the expansion of storage scale, increase of storage node data, the interval between adjacent primes is expanding, virtual node number will increase dramatically to complement the prime, along with the invalid calculations of encoding and data reconstruction increase. As a result, such schemes are greatly restricted in large storage. Weaver codes have a high fault tolerance, without any special requirements on stripe size and strip size, but the code is not based on any systematical coding method and lack of theoretical support. The fault tolerance ca-

pability of Weaver codes depends on computer engineering test, so it is difficult to apply in large storage system. Grid codes have better fault tolerance capability, to achieve simple storage with efficiency up to 80%, but the code in the construction process need to find two other array codes (must be the typical horizontal code or vertical code) to form matched codes. Fault tolerance capability of Grid codes depend on the fault tolerance capability of the matched codes selected. In addition, the Grid code is not a typical horizontal or vertical array code, and its storage layout combines the characteristics of horizontal and vertical arrays. Such a pattern brings in high fault tolerance and at the same time also makes its expansion subject to certain constraints in practical situation.

Table 1: Constraints on construction of array erasure codes

Array Codes	Stripe Size	Strip Size
EVENODD Codes	$s_r = p$	$s_c = s_r - 1$
X Codes	$s_r = p$	$s_c = s_r$
Liberation Codes	N/A	$s_c = p, p \geq s_r$
B Codes	TBD by the mathematical problem solving results	$s_c = 2 \cdot s_r \| s_c = 2 \cdot s_r + 1$
P Codes	$s_r = p \| s_r = p - 1$	$s_c = s_r/2$
RDP Codes	$s_r = p - 1$	$s_c = s_r$
Star Codes	$s_r = p$	$s_c = s_r - 1$
Weaver Codes	N/A	N/A
Grid Codes	TBD by matched codes	TBD by matched codes
Array codes presented in this paper	$s_r - \lceil f \cdot n/m \rceil \geq f \cdot (s_c - 1) + 1$	N/A

Unlike construction of most other array codes, the new method is based on the preset fault tolerance to construct the corresponding array codes, so there is no special constraint similar to prime for the array size. But as mentioned earlier, fault tolerance capability f, strip size m and column number n in data element array shall satisfy a linear constraint $n \geq m \cdot f - f + 1$. The number of fault tolerance f and block size m are usually determined by application environments, and when the two parameters are determined, n is determined. And under specific environment, when it is needed to limit the n, by the fault tolerant quantity and above inequality the value of m is calculated. Obviously, the constraint condition is easily satisfied, that is, the strength of the constraint condition is very weak.

3.3 Repair Cost and Update Penalty

Repair cost usually refers to the total number of storage nodes that are required to reconstruct a failure data element. The repair cost is an important performance index in the storage system, which is closely related to data reconstruction, data update and degraded reading and writing. By the coding chains deployment method, each chain has $m+1$ elements, including m data elements and 1 check element. Therefore, the number of nodes for reconstruction of a failure data element is m, which will not increase with the increase of storage system scale and fault tolerance capability.

Update penalty is an unique index in the horizontal layout array code, which refers to the number of check nodes that need to change 1 minimum data bit. When the data update is more frequent, the update penalty is too high, which will lead to the check node's access overheat and reduce the overall I/O performance of the storage system. In this paper, we can know that each data element has f different code chains, that is, each data element is related with f different check elements. In other words, when any one data element is changed, it always involves the f check nodes, that is, the update penalty is f, and it has reached the theoretical optimal value of the f fault tolerance array code.

4 Conclusion

Same with most current array codes, the proposed array codes still use only binary XOR in coding, with high operational efficiency. Differing from most previous array code construction methods, the method proposed in this paper is a kind of array code construction method based on specific fault tolerance, which can construct any array code of any fault tolerance capability according to environmental requirements. And satisfying the specific fault tolerance, by changing the strip size it adjusts the storage efficiency. In addition, the proposed array codes are subject to weak constraints when constructing, which are easy to meet. Finally, the array codes constructed by the proposed method also have the theoretical optimal update penalty and the repair cost which does not change with the storage size and fault tolerance capability. It is hoped that the method proposed in this paper can be used for the practical application of array codes in the field of storage, and it plays a positive role in other fields.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (Grant No. 61501064) and Sichuan Provincial Science and Technology Project (Grant No. 2017JQ0057, 2016GZ0122). And I want to take my deepest gratitude to Professor WANG Xiao-Jing for his guidance.

References

[1] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An optimal scheme for tolerating double disk failures in raid architectures," *ACM Sigarch Com-*

puter Architecture News, vol. 22, no. 2, pp. 245–254, 1995.

[2] J. L. Hafner, "Weaver codes: Highly fault tolerant erasure codes for storage systems," in *Conference on File and Storage Technologies*, pp. 16–16, 2005.

[3] C. Huang and L. Xu, "Star:an efficient coding scheme for correcting triple storage node failures," in *Conference on Usenix Conference on File and Storage Technologies*, pp. 15–15, 2005.

[4] M. Li, J. Shu, and W. Zheng, "Grid codes: strip-based erasure codes with high fault tolerance for storage systems," *ACM Transactions on Storage*, vol. 4, no. 4, pp. 1–22, 2009.

[5] Q. C. Meng, *Research of Erasure Codes Applied in Distributed Storage System*, PhD thesis, Beijing: Graduate University of the Chinese Academy of Sciences, 2007.

[6] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast galois field arithmetic using intel simd instructions," in *Fast: Usenix Conference on File and Storage Technologies*, 2013.

[7] J. S. Plank and L. Xu, "Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications," in *IEEE International Symposium on Network Computing and Applications*, pp. 173–180, 2006.

[8] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *Proceedings of the Vldb Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[9] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you?," in *Usenix Conference on File and Storage Technologies*, p. 1, 2007.

[10] D. Tang and H. P. Shu, "A class of array erasure codes with high fault-tolerance," *Scientia Sinica Informationis*, vol. 46, no. 4, pp. 523–538, 2016.

[11] J. Wang, X. Yu, and M. Zhao, "Fault-tolerant verifiable keyword symmetric searchable encryption in hybrid cloud," *International Journal of Network Security*, vol. 17, no. 4, pp. 471–483, 2015.

[12] L. Xu and J. Bruck, "X-code: Mds array codes with optimal encoding," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 272–276, 1999.

Biography

Dan Tang received PhD degree from Graduate University of Chinese Academy of Sciences, associate professor. His research interests include coding theory, secret sharing.

Ya-Qiang Wang received PhD degree from the College of Computer Science, Sichuan University. He is a lectur at Chengdu University of Information Technology. His research interests include big data analysis, fault tolerance.

Hao-peng Yang undergraduate student of Chengdu University of Information Technology. His research interests include big data analysis, fault tolerance.