V^2-Code: A New Non-MDS Array Code with Optimal Reconstruction Performance for RAID-6

Ping Xie[1], Jianzhong Huang[1*], Qiang Cao[1], Xiao Qin[2], Changsheng Xie[1]
[1]*School of Computer Science & Technology, Wuhan National Laboratory for OptoElectronics, Huazhong University of Science & Technology, WuHan, Hubei, P. R. China, 430074*
[2]*Department of Computer Science and Software Engineering, Auburn University, AL., USA.*
[*]*Corresponding author: hjzh@hust.edu.cn*

Abstract*—RAID-6 is widely used to tolerate concurrent failures of any two disks in both disk arrays and storage clusters. Numerous erasure codes have been developed to implement RAID-6, of which MDS Codes are popular. Due to the limitation of parity generating schemes used in MDS codes, RAID-6-based storage systems suffer from low reconstruction performance. To address this issue, we propose a new class of XOR-based RAID-6 code (i.e., V^2-Code), which delivers better reconstruction performance than the MDS RAID-6 code at low storage efficiency cost. V^2-Code, a very simple yet flexible Non-MDS vertical code, can be easily implemented in storage systems. V^2-Code's unique features include (1) lowest density, (2) steady length of parity chain, and (3) well balanced computation. We perform theoretical analysis and evaluation of the coding scheme under various configurations. The results show that V^2-Code is a well-established RAID-6 code that outperforms both X-Code and Code-M in terms of reconstruction time. V^2-Code can speed up the reconstruction time of X-Code by a factor of up to 3.31 and 1.79 under single disk failure and double disk failures, respectively.

***Keywords*-RAID-6; Non-MDS Code; Vertical Code; Lowest Density Code; Balanced Computation.**

I. Introduction

RAID techniques [1] are widely used in modern storage clusters to achieve high performance and reliability with acceptable spatial and monetary cost. Evidence shows that the possibility of disk failure occurrence grows with the increasing scale of storage systems [2][3]. RAID-6 coding schemes have increasingly received attention, because such coding schemes can tolerate double concurrent disk failures.

Various erasure coding technologies are applied to implement the RAID-6 layout. According to the structure and distribution of parities, RAID-6 can be mainly categorized into horizontal codes [4][5][6][7][8][9][10] and vertical codes [11][12][13][14]. A typical RAID-6 storage system using horizontal codes is composed of k+2 disks, where the first k disks store original information, and the last two are used as parity disks. Horizontal codes have a common disadvantage - k symbols must be read to recover any one failed symbol. Vertical codes are proposed to disperse parities across all disks instead of dedicated redundant disks (see, for example, X-Code[13], Cyclic code[11], and P-Code[14]).

A key design goals of erasure codes is to promote reconstruction performance [15][16][17]. **M**aximum **D**istance **S**eparable (**MDS**) code is a set of codes that require exact k symbols to reconstruct one failed or erased symbol. Such a limitation induced by the reconstruction chain length (i.e., k) potentially degrades the reconstruction performance. Non-MDS codes (e.g., Weaver codes [18], Hover codes [19], Code-M [20]) are proposed to overcome the reconstruction performance bottleneck. Compared to MDS codes, non-MDS codes use more parity symbols and largely reduce the number of symbols involved in the generation of one parity symbol, thereby increasing reconstruction performance. However, existing non-MDS codes behave poorly in some cases. For example, the Weaver code suffers from low storage efficiency (e.g., $\leq 50\%$); the HoVer code has extremely poor reconstruction performance when a horizontal parity disk fails. In Code-M, the disk array size is restricted by a prime number, which means that it does not support application scenarios of arbitrary number of disks.

The low reconstruction performance hurts data reliability and availability of RAID-based storage systems. On one hand, longer reconstruction time translates to a longer 'window of vulnerability', in which a second disk failure inevitably cause persistent data loss [21]; on the other hand, user requests on the foreground are adversely affected by the online reconstruction. To address these two issues, we propose a new class of XOR-based RAID-6 codes called **V^2-Code**, which is a **V**ertical code and its shape of the parity chain like the letter '**V**' in geometry.

The contributions of this paper is summarized as follows:

- Our V^2-Code, an efficient non-MDS XOR-based RAID-6 code, delivers better reconstruction performance than MDS RAID-6 codes at low storage efficiency cost. The parity chain length of V^2-Code is fixed, i.e., $2m$-1 for a m-row-n-column disk array.
- V^2-Code is a new class of lowest density array codes. The number of parity symbols that are affected by a change of any single information symbol is minimal, which reduces update complexity.
- V^2-Code is a vertical code, in which parity symbols are evenly distributed over all disk drives. The number of operations for computing parity symbol at each column is identical, meaning that computing load is evenly distributed among all disks. Such a load balancing feature naturally overcomes the bottleneck induced by repeated write operations.

- Our V^2-Code has the advantage of high reconstruction performance. The quantitative analysis demonstrates that V^2-Code speeds up the reconstruction time of X-Code by a factor of up to 3.31 and 1.79 under single disk failure and double disk failures, respectively.

The rest of the paper is organized as follows. Section II briefly overviews the related work. The architecture and reconstruction process of V^2-Code are presented in Section III. Section IV describes detailed property analysis and performance evaluation of V^2-Code. Finally, we conclude our work in Section V.

II. Related Work

Thanks to error-correcting capability, erasure coding techniques [22] are widely used in many fields such as telecommunication and data storage. Several well-known erasure coding schemes (e.g., Reed-Solomon codes and parity array codes) are capable of protecting against two or more disk failures in an array of disks. The Vandermonde-based Reed-Solomon code[4] offer high fault tolerance and optimal storage efficiency, but it requires the support of complex Galois field arithmetic (GF(2^w)). Aiming at the computation problem of complex multiply operations in Vandermonde Reed-Solomon codes, Cauchy Reed-Solomon codes[5] adopt the highly-efficient XOR operations. However, both Vandermonde and Cauthy Reed-Solomon codes still suffer from low storage performance [10].

Unlike the Reed-Solomon codes, parity array codes depends solely on XOR operations during encoding and decoding. This simplicity makes parity array coding fit for RAID storage systems. Among the existing parity array codes, RAID-6 is able to tolerate double concurrent node failures. Researchers have proposed many RAID-6 coding schemes, such as EVENODD code[6], RDP code[7], Blaum-Roth code[8], Liberation code[10], Liber8tion code[9], Cyclic code[11], X-Code[13], P-Code[14], and the like. Apart from the above MDS codes, a few non-MDS codes have been proposed to improve reconstruction performance. These non-MDS codes include WEAVER[18], HOVER[19], Pyramid code[23], Flat XOR-Code[24], and Code-M[20]. To make fair comparison, in this paper we focus on the parity array codes while choosing X-Code and Code-M as two baseline coding schemes. Both X-Code and Code-M belong to MDS parity array codes and non-MDS parity array codes, respectively.

A. MDS Parity Array Codes

Due to limited I/O bandwidth and expensive storage space, MDS codes became the primary focus of research for the sake of the best storage efficiency. MDS parity array codes can be divided into the horizontal codes [6][7] and vertical codes [13][14].

EVENODD and RDP are representative horizontal MDS codes for RAID-6 storage systems. Due to exist dedicated parity disks in horizontal codes, EVENODD and RDP suffer from the unbalanced load problem. There are two parity types - row parity and diagonal parity. The row parity of EVENODD is similar to that of RDP; the construction of the diagonal parity in RDP is different from that of EVENODD. From the layout of parities, we easily deduce that EVENODD is not an optimal solution from the perspective of both computational complexity and update complexity. Except for high update complexity, RDP obtains optimal computational complexity.

X-Codes and P-Codes are vertical MDS codes for RAID-6 storage systems, of which parity symbols are dispersed over all disks rather than in dedicated redundant disks. Such a layout achieves optimal computational complexity and update complexity. A recent study [25] shows that X-Codes exhibit balanced computation, and P-Codes are experiencing unbalanced computation.

From the parity layouts of both horizontal and vertical MDS codes, we observe that MDS-based RAID-6 schemes have a common disadvantage – the generation of each parity symbol needs exact k symbols on multiple disk drives; the long parity chain potentially becomes the performance bottleneck of reconstruction processes when any failure occurs.

B. Non-MDS Parity Array Codes

In recent years, I/O bandwidth increases and storage space becomes cheaper; however, the increasing gap between I/O bandwidth and disk capacity makes recovery time of an entire disk increases. Much attention has paid toward how to speed up recovery and improve the reliability of storage systems when disk failure occurs [16][20][17][26]. Non-MDS codes are regarded as favorable coding schemes to improve reconstruction performance of failed storage systems in the research community. Sample non-MDS codes are WEAVER codes[18], Hover codes[19], and Code-M[20].

WEAVER codes focus on tolerating multiple concurrent disk failures. The main drawback of the WEAVER codes is their low storage efficiency ($\leq$50%). HoVer codes belongs to the family of XOR-based lowest density codes. HoVer coding is a hybrid coding scheme that combines horizontal codes and vertical codes, which make it possible to achieve improved reconstruction performance under single and double disk failures. Just as every coin has two sides, HoVer has the following three limitations. First, HoVer's reconstruction cost varies, since it might need to read all information symbols to reconstruct lost symbols when a horizontal parity disk fails or two failed disks are adjoining. Second, HoVer results in unbalanced computing load due to dedicated parity disks. Last, some spare symbols are neither data symbols nor parity symbols, leading to a storage efficiency problem. Code-M is another non-MDS RAID-6 Code. Code-M code is designed to improve reconstruction performance as a lowest density code; however, the size of disk arrays must be restricted by a prime number.

III. The Design of V²-Code

To overcome the deficiency in reconstruction performance of the existing parity array codes, we propose a novel non-MDS code called V^2-Code for large-scale RAID-6 storage systems. Compared to the existing coding schemes, V^2-Code achieves optimal reconstruction performance under RAID-6 storage systems.

Let us denote V^2-Code(m, n) as a specific construction setting of V^2-Code for a m-row-n-column disk array, where $m \geq 2$ and $n \geq (4m\text{-}3)$. Information symbols are placed in an array of size $(m\text{-}1) \times n$, parity symbols are placed in an additional row, so the coded array is of size $m \times n$ (i.e., the first m-1 rows containing information symbols and the last row keeping parity symbols). Like existing array codes [6][13], parity symbols are constructed from information symbols along several *diagonals* of some slopes with the addition operation '+', which denotes the XOR operation here. Notice that each column has m-1 information symbols as well as a parity symbol. That is, information symbols and parity symbols are mixed in each column. If an error or an erasure of a symbol occurs in a column, then this column is considered to be an error or erasure column. This code structure allows surviving symbols to recover any two erased columns.

A. Layout and Encoding of V²-Code

V^2-Code is composed of a m-row-n-column square matrix with a total $m \times n$ symbols. There are two types of symbols in the square matrix: information symbols and parity symbols. $C_{i,j}$ denotes the symbol at the i^{th} row and j^{th} column, with $0 \leq i \leq$(m-1) and $0 \leq j \leq$(n-1), and the parity symbol $C_{m\text{-}1,j}$ at the j^{th} column is constructed using Equation (1).

$$C_{m\text{-}1,j} = \sum_{t=0}^{m\text{-}2} C_{t,<\text{j+m-1-t}>_n} + \sum_{t=0}^{m\text{-}2} C_{t,<\text{j-m+1+t}>_n} \qquad (1)$$

where j=$\{0, 1, \cdots, n\text{-}1\}$, and $<x>_n$=x *mod* n. Geometrically speaking, each parity symbol at the parity row is just the checksums along diagonals of slopes 1 and -1.

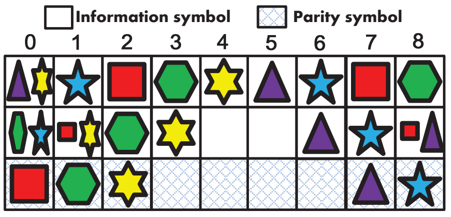

Figure 1. Parity layout of V²-Code(3,9) with a 9-disk array

Fig. 1 depicts an example of V^2-Code(3,9), where a column represents a disk drive and a block denotes a symbol of V^2-Code(3,9). Here, a stripe of V^2-Code(3,9) contains three rows, nine strips, or 27 symbols. The symbols with the same shape and color form a parity chain. For instance, there are four information symbols (e.g.,$C_{0,0}, C_{1,1}, C_{1,3}, C_{0,4}$) and one parity symbol (e.g.,$C_{2,2}$) in a parity chain marked as hexagon (see Fig. 1), parity symbol $C_{2,2}$ can be produced by $C_{0,0}$+$C_{1,1}$+$C_{1,3}$+$C_{0,4}$, and the length of parity chain equals to five.

From the layout described in Fig. 1, we derive the storage efficiency (i.e., the percentage of disk space occupied by non-parity symbols), which is governed by the number m of row rather than the number n of column. Thus, the storage efficiency of V^2-Code-based RAID system equals to $(m\text{-}1)/m$. For example, when m is 4, 75% of the capacity is used for non-parity symbols. Since V^2-Code offers the lowest density, each information symbol participates in two parity chains. Each parity chain contains the same number $2(m\text{-}1)$ of information symbols; in other words, the parity chain length of V^2-Code is constant (i.e., $2(m\text{-}1)$+1). To tolerate double failures, the number of strips in V^2-Code should be greater than or equal to $4(m\text{-}1)$+1.

B. Construction Process

According to the above information/parity layout and the encoding scheme, the construction process of V^2-Code contains the following two steps:

- Label all information symbols;
- Calculate parity symbols according to Equation (1).

C. The Correctness of V²-Code

To prove the correctness of V^2-Code, we consider the one-stripe reconstruction case in this subsection. The reconstruction of multiple stripes is identical to that of one stripe. We have the following lemma and theorem from the perspective of one-stripe construction.

Lemma 1: There exists a sequence of a two-integer tuple $(W_k, W_k^{'})$ that satisfies the following two conditions:

$$W_k = <m\text{-}1+(\frac{k+1}{2}+\frac{1+(-1)^k}{4})(f_2\text{-}f_1)>_m$$

$$W_k^{'} = \frac{1+(-1)^k}{2}f_1 + \frac{1+(-1)^{k+1}}{2}f_2 \;, (k\text{=}0, 1, \cdots, 2m\text{-}1)$$

With expression $0<(f_2\text{-}f_1)<$n, all two-integer tuples $\{(0, f_1), (0, f_2), \cdots, (m\text{-}1, f_1), (m\text{-}1, f_2)\}$ occur only once in the sequence. The proof of this lemma can be found in the literature on RAID-6 codes [6][13][14].

Theorem 1: A m-row-n-column stripe can be reconstructed from any two concurrent column failures.

Proof: Two failed columns are denoted as f_1 and f_2 where $0 \leq f_1 < f_2 < n$. Based on the layout of V^2-Code(m, n), we know that any two symbols in a parity chain do not reside in the same strip. Meanwhile, any information symbol always exists in two parity chains. For any two concurrent failed columns f_1 and f_2, there are two types of missing symbols, namely **independent missing symbols** and **dependent missing symbols**. The first type is a single symbol that fails in a parity chain; the second type occurs when two information symbols miss in a parity chain. In the former case, two independent missing symbols C_{i,f_1} and C_{i,f_2} at the i^{th} row are reconstructed, because the surviving symbols of the corresponding parity chain does not appear in the other failed column.

As to dependent missing symbols that are not in the same row, if an information symbol C_{i,f_2} at the i^{th} row f_2 column can be recovered, we can reconstruct the dependent missing symbol $C_{<i+f_2\text{-}f_1>_m,f_1}$ on the same parity chain. This reconstruction is possible, because the surviving symbols on the corresponding parity chain exist. Similarly, if information symbol C_{i,f_1} at column f_1 can be recovered, we can reconstruct the dependent missing symbol $C_{<i+f_2\text{-}f_1>_m,f_2}$ on the same parity chain.

Now we consider the case where dependent missing symbols are in the same row. If an information symbol C_{i,f_2} on failed column f_2 can be reconstructed, we can reconstruct dependent missing symbol C_{i,f_1} on the same parity chain. Such a reconstruction is reasonable, because the surviving symbols of the corresponding parity chain exist. Similarly, an information symbol C_{i,f_1} on failed column f_1 can be reconstructed; we can reconstruct the dependent missing symbol C_{i,f_2} on the same parity chain.

We conclude that all symbols can be reconstructed and the reconstruction order is based on the sequence of the two-integer tuple in Lemma 1. In summary, V^2-Code can tolerate any two concurrent column failures.

D. Reconstruction Process

In this subsection, the reconstruction algorithms for V^2-Code are presented under both single-disk-failure and double-disk-failure scenarios.

The decoding rules of all lost symbols are as follows:

$$C_{i,j}=\sum_{t=1}^{i} C_{i\text{-}t,<j\text{-}t>_n}+\sum_{\text{t=1}}^{\text{m-1-i}} C_{i+t,<j+t>_n}+\sum_{\text{t=1}}^{\text{m-1}} C_{m\text{-}1\text{-}t,<j+(m\text{-}1\text{-}i)+t>_n} \tag{2}$$

$$C_{i,j}=\sum_{t=1}^{i} C_{i\text{-}t,<j+t>_n}+\sum_{t=1}^{m\text{-}1\text{-}i} C_{i+t,<j\text{-}t>_n}+\sum_{t=1}^{m\text{-}1} C_{m\text{-}1\text{-}t,<j\text{-}(m\text{-}1\text{-}i)\text{-}t>_n} \tag{3}$$

1) Reconstruction algorithm for single-disk failures: In the case of single-disk failures, all failed symbols are independent missing symbols in Theorem 1. Therefore, it is straightforward to retrieve any information symbol $C_{i,j}$ in the j^{th} column according to Equation (2) or (3), with $0 \leq i \leq (m\text{-}2)$ and $0 \leq j \leq (n\text{-}1)$. Similarly, we can recover the lost parity symbol $C_{m\text{-}1,j}$ using Equation (1).

2) Reconstruction algorithm for double-disk failures: A strip distance represents an interval between one strip and another strip. There exist two strip-distance metrics — **S**trip **MI**nimum **D**istance(**SMID**) and **S**trip **MA**ximum **D**istance(**SMAD**). Assume there are n strips and the sequence numbers of two failed strips are f_1 and f_2, then both SMID and SMAD are min($<f_1\text{-}f_2>_n, <f_2\text{-}f_1>_n$) and max($<f_1\text{-}f_2>_n, <f_2\text{-}f_1>_n$), respectively.

Compared to the single-disk reconstruction, the double-disk reconstruction is much more complicated. Three cases of double-disk reconstruction are 1) two failed strips f_1 and f_2 are adjoining and SMID is one, 2) two failed strips f_1 and f_2 are not adjoining where SMID is $2(m$-1)+1, and 3) two failed strips f_1 and f_2 are in two different strips where SMID is greater than and equal to $4(m$-1)+1.

Compared to Case 1 that has only one recovery algorithm, both Cases 2 and 3 have multiple recovery algorithms to recover failed blocks. Although the recovery algorithms in the latter two cases exhibit the same computational complexity, they have different I/O complexity. We demonstrate each reconstruction process using a specific recovery algorithm.

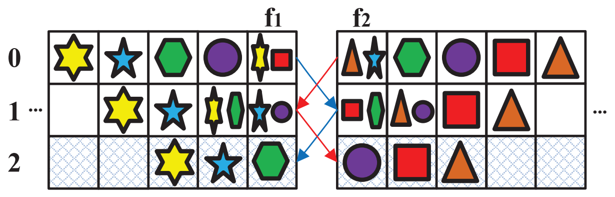

Figure 2. Reconstruction with two failed strips f_1 and f_2 in V^2-Code(m, n), where SMID is one and m equals to 3

Case 1: Two failed strips f_1 and f_2 are adjoining and SMID is one. There is only one way to retrieve the two strips. Fig. 2 shows a reconstruction layout in the V^2-Code(m, n) stripe where m=3. Intersections exists between parity chains of the f_1 strip and parity chains of the f_2 strip; On the other hand, different parity chains in a strip also intersect. Therefore, it is possible to reduce I/O complexity and achieve high reconstruction performance when the failures occur.

In this case, SMAD of the two adjoining failed strips is greater than or equal to $4(m$-1)+1. We observe from Fig. 2 that there are two recovery chains: (1) $C_{0,f_1} \rightarrow C_{1,f_2} \rightarrow C_{2,f_1}$ and, (2) $C_{0,f_2} \rightarrow C_{1,f_1} \rightarrow C_{2,f_2}$. All the symbols in a recovery chain depend on each other during the reconstruction process. The reconstruction process is described as follows. First, we identify two starting points of the recovery chain (i.e., information symbols C_{0,f_1} and C_{0,f_2}). Second, we determine two endpoints of the recovery chain (i.e., parity symbols C_{2,f_1} and C_{2,f_2}). Third, we reconstruct failed symbols according to the corresponding recovery chains. Algorithm 1 outlines the double-disk reconstruction process for strips f_1 and f_2 in a V^2-Code(m, n)-based stripe.

Case 2: Two failed strips f_1 and f_2 are not adjoining where the SMID is $2(m$-1)+1. There are several approaches to retrieving the two strips. Fig. 3 shows a layout for reconstruction in a V^2-Code(m, n) stripe where m=3. All lost symbols are independent of each other during the course of reconstruction. Similar to the above case (Case 1), intersections exist in both parity chains of intra-strip and parity chains of inter-strip, which makes it possible

Algorithm 1: Reconstruction algorithm of double-disk failures of f_1 and f_2 where the SMID is one in a V^2-Code(m,n)-based stripe

```
Step 1: Identify the double failed columns: $f_1$ and $f_2$( $f_1 < f_2$)
Step 2: Start reconstructing the lost symbols of strips $f_1$ and $f_2$.
switch $0 \leq f_1 < f_2 \leq n-1$ do
    Step 2-A: Reconstruct two starting points $C_{0,f_1}$ and $C_{0,f_2}$ of
    the recovery chains based on Equations (3) and (2), respectively.
    Step 2-B: Recover the remaining lost symbols in the two
    recovery chains.
    Two cases start synchronously:
    case starting point is $C_{0,f_1}$
        repeat
            if i=m-1 then
                Reconstruct the lost parity symbol with
                Equation (1).
            else
                (1) Reconstruct the next lost information
                symbol(in column $f_1$) with Equation (3);
                (2) Then reconstruct the next lost information
                symbol(in column $f_2$) with Equation (2).
            end
        until at the endpoint of the recovery chain.;
    case starting point is $C_{0,f_2}$
        repeat
            if i=m-1 then
                Reconstruct the lost parity symbol with
                Equation (1).
            else
                (1) Reconstruct the next lost information
                symbol(in column $f_1$) with Equation (3);
                (2) Then reconstruct the next lost information
                symbol(in column $f_2$) with Equation (2).
            end
        until at the endpoint of the recovery chain;
endsw
```

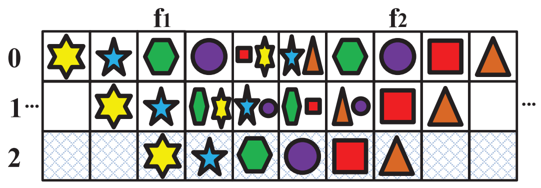

Figure 3. Reconstruction with two failed strips f_1 and f_2 in V^2-Code(m,n), where the SMID is 2(m-1)+1 and m=3

to reduce I/O complexity and achieve good reconstruction performance.

In the failure scenario, SMAD of the two failed strips f_1 and f_2 is no less than 2(m-1)+1. Fig. 3 demonstrates the reconstruction under double disk failures with two different strips where SMID is 2(m-1)+1 and m=3. Algorithm 2 shows the reconstruction process of concurrent failures of f_1 and f_2 with SMID=2(m-1)+1 and SMID=4(m-1)+1 in the V^2-Code(m,n) stripe.

Case 3: Two failed strips f_1 and f_2 are in two different strips where SMID is no less than 4(m-1)+1, and SMAD of the two failed strips f_1 and f_2 is no less than 4(m-1)+1. We also have several choices to reconstruct the two failed strips. Fig. 4 shows a specific layout for reconstruction in the V^2-Code$(3,n)$ stripe. All the lost symbols are independent of one another during the reconstruction. There are no intersection between parity chains of the f_1 strip and parity chains of the f_2 strip, but parity chains in a strip have intersections. The corresponding reconstruction process is listed in Algorithm 2.

Algorithm 2: Reconstruction algorithm of double-disk failures in the V^2-Code(m,n) with SMID=2(m-1)+1 and SMID=4(m-1)+1

```
Step 1: Identify the two failed column: $f_1$ and $f_2$( $f_1 < f_2$)
Step 2: Reconstruct lost symbols of the $f_1$ column.
foreach lost symbol $C_{i,j}$ in the failure column $f_1$ do
    if i=m-1 then
        Reconstruct the lost parity symbol with Equation (1).
    else
        Reconstruct the lost information symbol with Equation (2);
    end
end
Step 3: Reconstruct lost symbols of the $f_2$ column.
foreach lost symbol $C_{i,j}$ in the failure column $f_2$ do
    if i=m-1 then
        Reconstruct the lost parity symbol with Equation (1).
    else
        Reconstruct the lost information symbol with Equation (3);
    end
end
```

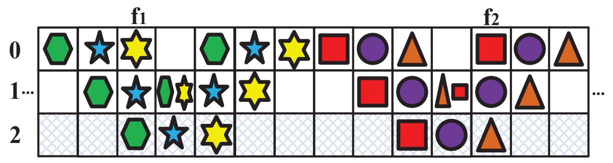

Figure 4. Reconstruction with two failed strips f_1 and f_2 in V^2-Code(m,n), where SMID is no less than 4(m-1)+1 and m=3

IV. Performance and Property Analysis

In this section, we first summarize the properties of V^2-Code, including lowest density code, steady parity chain length, balanced computation, and flexibility. Then, we comprehensively evaluate the performance of V^2-Code in the normal mode and degraded mode. Finally, we fully compare V^2-Code with X-Code and Code-M under the same settings (e.g., both the number of disk and the amount A of user information in a disk array is identical).

A. Property Analysis

Our V^2-Code is a Non-MDS vertical coding scheme, which takes full advantage of the layout of both diagonal parities and anti-diagonal parities. From the geometrical structure of V^2-Code, we highlight four features as follows.

1) Lowest density code: According to the construction process of V^2-Code(m,n), each parity symbol at the m^{th} row (i.e., parity row) is independently calculated from information symbols. In particular, each information symbol affects two parity symbols in the parity row. All parity symbols only depend on information symbols; the parity symbols are independent of one another. Since updating one information symbol only needs to update two parity symbols, V^2-Code has the optimal update complexity.

2) Steady parity chain length: In V^2-Code, each parity chain contains the same number $2m$-2 of information symbols. Thus, the length of parity chain is constant, i.e., $2m$-1 for a given row number m. The parity chain length has nothing to do with the column number n of a disk array. Hence, V^2-Code achieves better reconstruction performance than other MDS RAID-6 codes at marginal storage efficiency cost.

3) Balanced computation: In V^2-Code, each column has one parity symbol, which is the checksum of $2m$-2 information symbols; consequently, the number of operations for computing a parity symbol at each column is $2m$-3. This balanced computation property makes V^2-Code applicable to applications that require evenly distributed computations.

4) High Flexibility: Parameter m in V^2-Code(m, n) represents an arbitrary row number where m is no less than 2; n represents an arbitrary number of column where n is greater than and equal to $4m$-3. The arbitrary numbers of rows and columns make our approach applicable to large-scale RAID systems. In contrast, many other codes (e.g., EVENODD, RDP, P-Code, Code-M) are limited to RAID systems using a prime number as the disk array size. For example, Code-M(S, C) is defined by a C-row-S*C-column matrix, where C+1 must be a prime number; the number of strip-sets in Code-M can not be less than 3; and the column number in Code-M must grow at a step of C strips. Therefore, V^2-Code is very flexible in terms of RAID size.

B. Performance Analysis

In this subsection, we comprehensively discuss V^2-Code and quantitatively evaluate its performance in the normal mode and the degraded mode (e.g., single-disk failure as well as double-disk failures).

1) Performance in the normal mode: Basic operations are read and write/update in a RAID system in the normal mode. Read operations include small reads (reading one symbol at a time) and strip reads (reading one strip at a time). Similarly, write operations also include small writes and strip writes. In the normal mode, the I/O overheads of V^2-Code(m, n) are listed as follows. (1) A small read does not cause any additional overhead; I/O time of small reads in V^2-Code is the same as that in the existing RAID codes. (2) A strip read does not utilize all the disk bandwidth, since the parity symbol in that strip does not need to be accessed. (3) A strip write should update $2m$-2 disks. (4) A small write/update operation on an information symbol causes two additional write/update operations, which are optimal for any code tolerating double disk failures.

2) Performance in the degraded mode: In the degraded mode, decoding complexity is a very important performance metric for RAID systems, because it greatly affects both user response time and reconstruction time, and the reliability of RAID systems has an inverse relationship with the reconstruction time.

We first consider the case of single disk failure in the degraded mode. According to Equations (1), (2), and (3), $2m$-3 XOR operations are required to recover any one failed symbol, $2m$*$(m$-1$)$-1 symbols should be read and m symbols should be written to recover one failed strip. Accordingly, to recover m*$A/((m$-1$)$*$n)$ symbols in one failed disk, the decoding computational complexity is m*$(2m$-3$)$*$A/((m$-1$)$*$n)$ XOR operations, and the decoding I/O complexity is to access $(m$*$(2m$-1$)$-1$)$*$A/((m$-1$)$*$n)$ symbols.

Now we discuss the double-disk-failure case in the degraded mode. In this case, $2m$*$(2m$-3$)$ XOR operations are needed to recover two strips. That is, the decoding computational complexity is $2m$*$(2m$-3$)$ XORs. In what follows, let us discuss the I/O complexity of V^2-Code(m, n) codes for each case described in Section III.

Case 1 - the two failed strips are adjoining and SMID is one: According to Fig. 2 and Algorithm 1, $2m$*$(2m$-3$)$ symbols should be read and $2m$ symbols should be written to reconstruct two strips. Therefore, the decoding I/O complexity lies in accessing $4m$*A/n symbols when rebuilding m*$A/((m$-1$)$*$n)$ lost symbols in each of the two failed disks.

Case 2 - the two failed strips are in different strips and SMID is $2(m$-1$)$+1*:* To recover two strips (see Fig. 3 and Algorithm 2), a reconstruction mechanism must read $2m$*$(2m$-3$)$ symbols and write $2m$ symbols. Therefore, to recover m*$A/((m$-1$)$*$n)$ lost symbols in each of the two failed disks, the decoding I/O complexity becomes accessing $4m$*A/n symbols.

Case 3 - the two failed strips are in different strips and SMID is no less than $4(m$-1$)$+1*:* According to Fig. 4 and Algorithm 2, to recover two strips, $4m$*$(m$-1$)$-2 symbols should be read and $2m$ symbols should be written. So the decoding I/O complexity is to access $(2m$*$(2m$-1$)$-2$)$*$A/((m$-1$)$*$n)$ symbols, when recovering m*$A/((m$-1$)$*$n)$ lost symbols in each of the two failed disk.

C. Performance Comparisons

In this subsection, we choose X-Code and Code-M as two baseline coding schemes, because they share some common characteristics with V^2-Code. For example, the parity symbol construction of our V^2-Code is similar to that of X-Code, and Code-M is also a non-MDS array code for RAID-6.

We compare V^2-Code with X-Code and Code-M under the same number of total disks and the same I/O bandwidth for reconstruction. We recover valid user information symbols in a failed disk; a similar approach can be found in [27][26]. The results of the two comparison scenarios are given in terms of reconstruction performance ratio.

1) Comparisons between V^2-Code and X-Code: We evaluate the recovery performance of V^2-Code(m, n) with X-Code(p, p) under the same disk number, i.e., n=p. To recover

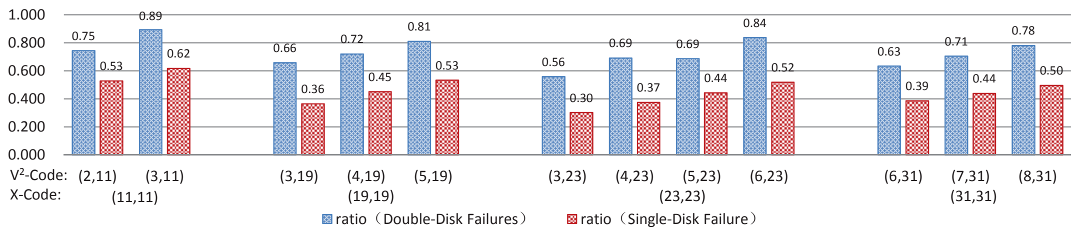

Figure 5. Comparisons of the reconstruction times between V^2-Code and X-Code with same number of disks.

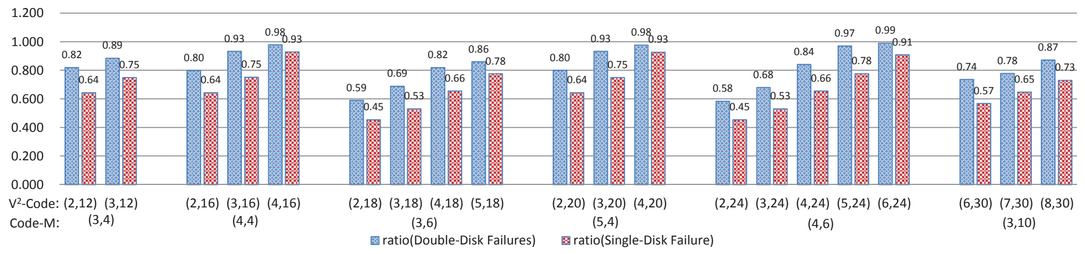

Figure 6. Comparisons of the reconstruction times between V^2-Code and Code-M with same number of disks.

one failed disk, the computational complexity of X-Code is $(p\text{-}3)*A/(p\text{-}2)$ and the I/O complexity of X-Code is $(p^2\text{-}2p\text{+}3)*A/(p*(p\text{-}2))$. To recover double failed disks, the computational complexity of X-Code is $2(p\text{-}3)*A/(p\text{-}2)$ and the I/O complexity of X-Code is $(p*A)/(p\text{-}2)$.

Fig. 5 shows the reconstruction time ratios between V^2-Code(m,n) and X-Code when the number of disks is set to 11, 19, 23 and 31, respectively. We observe that under a single disk failure, V^2-Code consistently outperforms X-Code; V^2-Code speeds up the reconstruction time of X-Code by a factor of up to 3.31. Under double disk failures, V^2-Code still outperforms X-Code in all the tested cases, and the reconstruction speedup is up to a factor of 1.79.

2) Comparisons between V^2-Code and Code-M: We assess the recovery performance of both V^2-Code(m,n) and Code-M(S,C) in the case of same disk number, i.e., n=$S*C$. When recovering one faulty disk, the computational complexity and I/O complexity of Code-M(S,C) are $(2C\text{-}3)*A/(S*(C\text{-}1))$ and $(2C\text{-}1)*A/(S*(C\text{-}1))$, respectively. When it comes to rebuilding double failed disks that are in the same or adjoining strip-set, the computational and I/O complexities of Code-M(S,C) are $(4C\text{-}6)*A/(S*(C\text{-}1))$ and $(3C\text{-}1)*A/(S*(C\text{-}1))$, respectively.

Fig. 6 shows the ratios of reconstruction performance between V^2-Code and Code-M under various configurations using 12, 16, 18, 20, 24 and 30 disks, respectively. It is clear that under all the configurations, V^2-Code consistently exhibits higher recovery speed than that of Code-M in both the single-disk-failure case and the double-disk-failure case. In addition, we observe that V^2-Code improves the recovery performance of Code-M under the same disk number, storage efficiency, and RAID capacity.

V. Conclusion

In this paper, we proposed a novel coding scheme called V^2-Code to tolerate up to two concurrent disk failures for large-scale RAID-6 storage systems. Our V^2-Code is a non-MDS RAID-6 code, which achieves better reconstruction performance than that of MDS RAID-6 codes at small storage efficiency cost. V^2-Code is a lowest density code, with its parity chain length fixed at $2(m\text{-}1)\text{+}1$ for a given number m of rows in a disk array. V^2-Code's storage efficiency is very high (i.e., $(m\text{-}1)/m$). We conducted extensive theoretical analysis for V^2-Code under various configurations. The results show that V^2-Code outperforms both X-Code and Code-M in terms of reconstruction time. For example, V^2-Code speeds up the reconstruction time of X-Code by a factor of up to 3.31 and 1.79 in the single-disk-failure case and the double-disk-failure case, respectively.

V^2-Code exhibits an array of benefits such as balanced computation, flexible RAID size, and fixed parity chain length. These advantages allow our V^2-Code to be applied to a wide range of in-production applications running in distributed storage environments. As a future direction, we plan to extend V^2-Code to a distributed storage systems comprised of heterogeneous data nodes.

Acknowledgment

This work is sponsored in part by the National Basic Research Program of China (973 Program) under Grant No.2011CB302303, the National Natural Science Foundation of China under Grant No.60933002, and National High Technology Research and Development Program of China (863 Program) under Grant No.2013AA013203. Xiao Qin's work was supported by the U.S. National Science Foundation under Grants CCF-0845257(CAREER), CNS-0917137(CSR), and CCF-0742187(CPA).

References

[1] D. A. Patterson, G. Gibson, and R. H. Katz, *A case for redundant arrays of inexpensive disks (RAID)*. ACM, 1988, vol. 17, no. 3.

[2] 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 *Proceedings of the 5th USENIX conference on File and Storage Technologies*, vol. 6, 2007, p. 2.

[3] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of the 5th USENIX conference on File and Storage Technologies*, 2007, pp. 2–2.

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[5] M. Luby and D. Zuckermank, "An xor-based erasure-resilient coding scheme," Tech Report, Tech. Rep., 1995.

[6] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," *Computers, IEEE Transactions on*, vol. 44, no. 2, pp. 192–202, 1995.

[7] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 1–14.

[8] M. Blaum and R. M. Roth, "On lowest density mds codes," *Information Theory, IEEE Transactions on*, vol. 45, no. 1, pp. 46–59, 1999.

[9] J. S. Plank, "A new minimum density raid-6 code with a word size of eight," in *Network Computing and Applications, 2008. NCA'08. Seventh IEEE International Symposium on*. IEEE, 2008, pp. 85–92.

[10] J. S. plank, "A new mds erasure code for raid-6," in *Technical Report CS-07-602, University of Tennessee*, 2007.

[11] Y. Cassuto and J. Bruck, "Cyclic lowest density mds array codes," *Information Theory, IEEE Transactions on*, vol. 55, no. 4, pp. 1721–1729, 2009.

[12] L. Xu, V. Bohossian, J. Bruck, and D. G. Wagner, "Low-density mds codes and factors of complete graphs," *Information Theory, IEEE Transactions on*, vol. 45, no. 6, pp. 1817–1826, 1999.

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

[14] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-code: A new raid-6 code with optimal properties," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 360–369.

[15] J. S. Plank, A. L. Buchsbaum, R. L. Collins, and M. G. Thomason, "Small parity-check erasure codes-exploration and observations," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005, pp. 326–335.

[16] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang, and Z. Song, "Pro: a popularity-based multi-threaded reconstruction optimization for raid-structured storage systems," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, 2007, pp. 277–290.

[17] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "Workout: I/o workload outsourcing for boosting raid reconstruction performance," in *Proceedings of the Seventh USENIX Conference on File and Storage Technologies (FAST09)*, 2009.

[18] J. L. Hafner, "Weaver codes: Highly fault tolerant erasure codes for storage systems," in *Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, vol. 4, 2005, pp. 16–16.

[19] J. L. hafner, "Hover erasure codes for disk arrays," in *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. IEEE, 2006, pp. 217–226.

[20] S. Wan, Q. Cao, C. Xie, B. Eckart, and X. He, "Code-m: A non-mds erasure code scheme to support fast recovery from up to two-disk failures in storage systems," in *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*. IEEE, 2010, pp. 51–60.

[21] Q. Xin, E. L. Miller, T. Schwarz, D. D. Long, S. A. Brandt, and W. Litwin, "Reliability mechanisms for very large storage systems," in *Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*. IEEE, 2003, pp. 146–156.

[22] F. MacWilliams and N. Sloane, "The theory of error-correcting codes," 2006.

[23] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*. IEEE, 2007, pp. 79–86.

[24] K. M. Greenan, X. Li, and J. J. Wylie, "Flat xor-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–14.

[25] C. Wu, X. He, G. Wu, S. Wan, X. Liu, Q. Cao, and C. Xie, "Hdp code: A horizontal-diagonal parity code to optimize i/o load balancing in raid-6," in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 209–220.

[26] S. Wu, D. Feng, H. Jiang, B. Mao, L. Zeng, and J. Chen, "Jor: A journal-guided reconstruction optimization for raid-structured storage systems," in *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*. IEEE, 2009, pp. 609–616.

[27] M. Sivathanu, V. Prabhakaran, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Improving storage system availability with d-graid," *ACM Transactions on Storage (TOS)*, vol. 1, no. 2, pp. 133–170, 2005.