2015 IEEE 34th Symposium on Reliable Distributed Systems

PCM: A Parity-check Matrix Based Approach to Improve Decoding Performance of XOR-based Erasure Codes

Yongzhe Zhang*, Chentao Wu*‡, Jie Li*†, Minyi Guo*
*Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
†Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki, Japan
‡Corresponding author: wuct@cs.sjtu.edu.cn

***Abstract*—In large storage systems, erasure codes is a primary technique to provide high reliability with low monetary cost. Among various erasure codes, a major category called XOR-based codes uses purely XOR operations to generate redundant data and offer low computational complexity. These codes are conventionally implemented via matrix based method or several specialized non-matrix based methods. However, these approaches are insufficient on decoding performance, which affects the reliability and availability of storage systems.**

To address the problem, in this paper, we propose a novel Parity-Check Matrix based (PCM) approach, which is a general-purpose method to implement XOR-based codes, and increases the decoding performance by using smaller and sparser matrices. To demonstrate the effectiveness of PCM, we conduct several experiments by using different XOR-based codes. The evaluation results show that, compared to typical matrix based decoding methods, PCM can improve the decoding speed by up to a factor of 1.5× when using EVENODD code (an erasure code for correcting double disk failures), and accelerate the decoding process of STAR code (an erasure code for correcting triple disk failures) by up to a factor of 2.4×.

***Keywords*-Reliability, Erasure Code, Parity-check Matrix, Performance Evaluation**

I. Introduction

With the tremendous requirements on reliability in cloud computing systems, nowadays erasure code is an effective method to provide high reliability for storage systems [17] [36] [24] [34]. In these systems, redundant data are organized by erasure codes, which forms a reliable storage system (e.g., RAID[1]) to tolerate concurrent disk failures. It is an efficient way to provide both high performance and high reliability storage services with low monetary cost.

Among various erasure codes, a major category is based on XOR operations, which is called XOR-based codes. Another category uses addition and multiplication over finite field to generate parities. Right now XOR-based codes show great advantages on encoding/decoding speed [29] [4] [33] [19] [37]. Typically, some XOR-based codes are used for correcting double disk failures in RAID-6 (i.e., EVENODD [1], RDP [6]), and the others can tolerate concurrent disk failures of three or more disks (i.e., STAR [15], HoVer [12]).

Conventionally, there are two approaches to implement XOR-based codes in storage systems. One approach [13] (called "matrix based approach") performs encoding and decoding based on the *generator matrix* [31], which is a general-purpose method to implement XOR-based codes. The other approach (called "non-matrix based approach") uses specialized algorithm for each code (such as EVENODD [1], STAR [15], etc.), which utilizes the properties of the code for encoding and decoding.

However, existing approaches are insufficient to provide high decoding performance[2], which affects the reliability of the storage systems in terms of reconstruction time, and has great impact on data availability as well. On one hand, matrix based method has disadvantages on decoding speed due to the large size of the decoding matrix. On the other hand, non-matrix based methods has many limitations due to the following reasons (detailed illustration will be given in Section II). First, in some scenarios, the decoding speed of several codes are slow [16], such as EVENODD [1] and STAR [15]. Second, the implementation of non-matrix based approach is complex. Third, non-matrix based approach can be used in several specific erasure codes (such as RDP [6], EVENODD [1], STAR [15], etc), but it is not a general-purpose method for most erasure codes.

To address the problem, in this paper, we propose a *Parity-Check Matrix* based (PCM) approach for XOR-based codes, which is a general-purpose approach with high decoding performance. By using parity-check matrix, we discover small and sparse matrices to represent the relationship between lost data and surviving data, thus has much higher decoding efficiency than traditional matrix based approach.

The contribution of our work includes,

1) We propose a novel Parity-Check Matrix based (PCM) approach to implement XOR-based erasure codes based on parity-check matrix, which shows great advantages on decoding performance.
2) We conduct a series of simulations and experiments to compare PCM with traditional approaches on decoding performance. The results show that, PCM achieves high decoding performance by using different erasure codes such as EVENODD and STAR.

The rest of paper is organized as follows. In Section II, we

[1]RAID: Redundant Array of Independent (or Inexpensive) Disks [25].

[2]In this paper, decoding performance refers to the speed of reconstructing lost data under the maximum number of device failures.

1060-9857/15 $31.00 © 2015 IEEE
DOI 10.1109/SRDS.2015.15

introduce related work and our motivation. In Section III, the design and optimization techniques of PCM is illustrated in detail. The evaluation is presented in Section IV. Finally we conclude our work in Section V.

II. Related Work and Motivation

In this section, we introduce the related work and our motivation. To facilitate the discussion, we summarize the symbols used in this paper in Table I.

TABLE I
Symbols Used In This Paper

Symbols	Description
n	the number of disks in a disk array
m	total number of disk failures
k	an integer with $k = n - m$
w	the word size
i	row ID
j	column ID (disk ID)
$C_{i,j}$	an element at the ith row and jth column
D_i	the ith disk
G	a generator matrix
$\hat{G}$	a sub-matrix of G with rows corresponding to the surviving data
H	a parity-check matrix
H_L	a sub-matrix of H with columns corresponding to the lost data
H_S	a sub-matrix of H with columns corresponding to the surviving data
p_i	a parity element defined by the ith row of H

A. Erasure Codes

Erasure codes are widely used in clusters or data centers to prevent data loss with low spatial and monetary cost. Typically, erasure codes can be divided into two categories. One class is based on arithmetic over finite field, and the other is based on XOR operations.

Reed-Solomon (RS) code [35] is a traditional erasure code based on finite field, whose multiplication is expensive [29] [4] [33] [19]. Recently, several erasure codes based on RS codes are proposed to improve the efficiency of storage systems, such as Pyramid Code [14], Local Reconstruction Codes [17] and Locally Repairable Codes [36], SD codes [28] and STAIR codes [20].

XOR-based codes is a major category of erasure codes that uses purely XOR operations in computation, and offers low computational cost on encoding/decoding. Erasure codes for correcting double or triple disk failures are widely used in storage systems. Typically, XOR-based codes can be further divided into two categories, XOR-based codes for correcting double disk failures in RAID-6 (referred to as "RAID-6 codes") and XOR-based codes for tolerating triple disk failures. Typical RAID-6 codes include EVENODD code [1], RDP code [6], Blaum-Roth code [2], X-code [46], Liberation code [27], Liber8tion code [26], Cyclic code [5], B-Code [45], Rotary-code [42], H-code [44], P-code [18], HDP-code [43] and HV-code [37]. Then, STAR code [15], Triple-Star code [41], TP technique [7], HDD1 code [40], RSL-code [8] and RL-code [9], WEAVER codes [11], HoVer codes [12], T-code [39], HDD2 code [40] and Cauchy Reed-Solomon codes [3] are proposed to tolerate three or more disk failures.

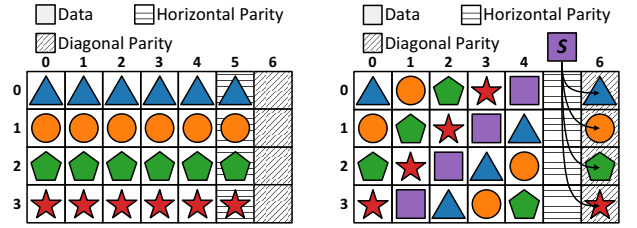
B. Problems in Existing Decoding Approaches

In the last two decades, researchers propose two approaches to implement XOR-based codes in disk arrays, matrix/non-matrix based approaches. Matrix based approach can be applied to various codes, and non-matrix based approach is proposed for specific codes according to the parity layout.

1) Non-matrix based Approach: This approach [46] [1] [6] [15] decodes the erasure codes via specific reconstruction algorithms. We take EVENODD code [1] as an example, which is a typical code for illustrating the decoding process.

EVENODD code [1] is a typical RAID-6 code to tolerate double disk failures, which supports $p + 2$ disks, where p is a prime number. Fig.1 and Fig.2 show the encoding and decoding procedures of EVENODD when $p = 5$, respectively.

In the encoding process, a parity element is calculated via XOR operations among the data elements with the same shape (e.g., $C_{0,5} = C_{0,0} \oplus C_{0,1} \oplus C_{0,2} \oplus C_{0,3} \oplus C_{0,4}$ as shown in Fig.1(a).).



(a) Horizontal parity coding (e.g., $C_{0,5} = C_{0,0} \oplus C_{0,1} \oplus C_{0,2} \oplus C_{0,3} \oplus C_{0,4}$)

(b) Diagonal parity coding (e.g., $S = C_{3,1} \oplus C_{2,2} \oplus C_{1,3} \oplus C_{0,4}$ and $C_{0,6} = S \oplus C_{0,0} \oplus C_{3,2} \oplus C_{2,3} \oplus C_{1,4}$)

Fig. 1. Encoding of EVENODD ($p = 5$). It's represented by a $(p-1) \times (p+2)$ matrix, where the first p and the last two columns delegate the data and the parity columns, respectively. We use different icon shapes to denote different sets of parity chains, which consist of parity elements and their corresponding data elements. S is an internal element calculated by several data elements, and it is finally added to every diagonal parity element.

The decoding process of EVENODD can be divided into many scenarios, regarding to the number of disk failures and the location of failed disks. For decoding a single data/parity element (or a single column), it can be computed via encoding equations (e.g., $C_{0,1} = C_{0,5} \oplus C_{0,0} \oplus C_{0,2} \oplus C_{0,3} \oplus C_{0,4}$ as shown in Fig.1(a).). When double columns fail, first we need to find the recovery chain(s) and decode the starting point(s) of the recovery chain(s), and then decode all lost elements based on the order in the recovery chain(s).

However, non-matrix based approach has the following disadvantages,

- *Low Efficiency*. In some scenarios, the specific decoding algorithm is inefficient, such as EVENODD and STAR [16].
- *High Complexity in Implementation*. When double or triple columns fail, the decoding process is complex by using non-matrix based approach. It is because that many

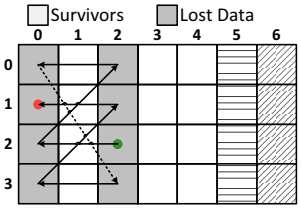


Fig. 2. Decoding of EVENODD ($p = 5$) when columns 0 and 2 fail. First we identify the starting point of recovery chain: $C_{2,2}$. Second we reconstruct data elements according to the corresponding recovery chains until they reach the endpoint ($C_{1,0}$). The order to decode data elements is $C_{2,2}$ $C_{2,0}$ $C_{0,2}$ $C_{0,0}$ $C_{3,2}$ $C_{3,0}$ $C_{1,2}$ $C_{1,0}$.

different scenarios should be considered in the decoding process. For example, when columns 0 and 1 fail, two recovery chains are handled in the decoding process, which is different from the example in Fig.2.

- *Low Flexibility*. Non-matrix based approach can be used in several specific erasure codes (such as STAR [15] and EVENODD [1]), but it is not a general-purpose method for all erasure codes.

2) Matrix based Approach: Matrix based approach [13] is a popular choice to implement various erasure codes. Given a generator matrix of an erasure code, it can provide standardized encoding and decoding algorithms.

By using matrix based approach, the encoding of XOR-based erasure codes is represented by a matrix-vector product. The data is a vector of k words. Each word contains w elements, and each element represents a data block in disk array. A generator matrix G transforms the data into a *codeword*, which contains the original data and m additional parity words. Fig.3 shows the encoding of EVENODD code using generator matrix when $p = 5$, which generates two redundant parity words by using p data words.

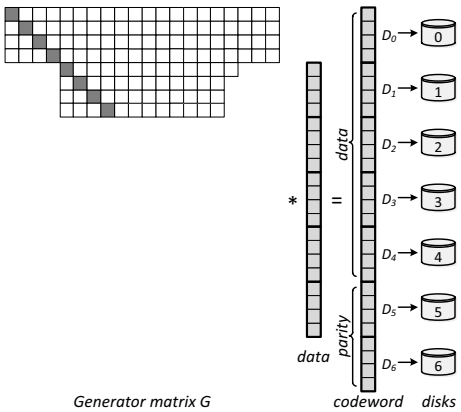


Fig. 3. Matrix based encoding procedure for EVENODD code ($p = 5$) by using generator matrix, which transforms the data into a codeword (the encoded data). G is a *bit matrix*, and the element in this matrix is either 0 or 1. Each element in the codeword is computed by the dot product of the corresponding row in G and the data, where an addition is an XOR operation and a multiplication is a bitwise AND.

When some data words or parity words are lost, the decoding process of matrix based approaches abides by the following steps. First it constructs a new matrix $\hat{G}$ from the rows in the generator matrix corresponding to the surviving data, such that the product of $\hat{G}$ and the original data equals to the surviving data. Second, the original data is recovered by multiplying the generalized inverse of $\hat{G}$ with the surviving data. Finally the lost data is recovered by multiplying the generator matrix G with the original data.

In previous literatures, several optimization techniques are proposed to increase the decoding performance for matrix based approach. *Bit matrix scheduling* [27] is introduced to accelerate the calculation of matrix-vector product by reusing the recovered elements. And then, Huang *et al.* [16] give two greedy approaches to optimize the decoding of Cauchy Reed-Solomon (CRS) codes [3]. Later, Luo *et al.* [23] study the cache effect and presents *DWG XOR-Scheduling Algorithm* to reduce cache miss via scheduling XOR operations, which improves the decoding performance in implementation. Recently, Li *et al.* [21] propose a scheduling algorithm to reduce data transmission for degraded reads in distributed systems.

However, the decoding efficiency of matrix based approach is restricted as well. A critical problem is that, it cannot decode some erasure codes (such as STAR [15] and Triple-Star [41]) efficiently, which is mainly caused by the large size of the decoding matrix. Here we take STAR and Triple-Star code as examples. Fig.4 shows the *decoding complexity*[3] [38] of these two codes by using matrix based approach and non-matrix based approach. It is clear that matrix based approach has significant drawback on decoding performance.

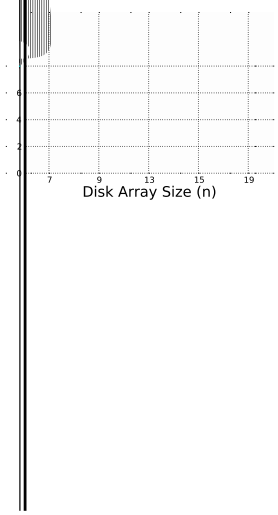


TABLE II
SUMMARY ON EXISTING DECODING APPROACHES FOR XOR-BASED CODES

Decoding Approaches	Decoding Efficiency	Complexity in Implementation	Flexibility
Matrix based	low	low	high
Non-matrix based	medium	high	low
PCM	high	low	high

III. PARITY-CHECK MATRIX BASED APPROACH

In this section, we introduce our Parity-Check Matrix based (PCM) approach, which is an efficient decoding method for the implementation of XOR-based erasure codes.

A. Parity-check Matrix for Erasure Codes

A parity-check matrix is a bit matrix which describes the linear relationships among data elements in a codeword. For an XOR-based MDS code that has k data words and m parity words with a word size w, the dimension of its parity-check matrix is $mw \times nw$ where $n = k+w$. In a parity-check matrix, each column represents an element in a codeword, and each row is a parity-check equation, which specifies a subset of elements in the codeword with an XOR sum of 0. A property of parity-check matrix is that, the product of a parity-check matrix and the corresponding codeword is always 0.

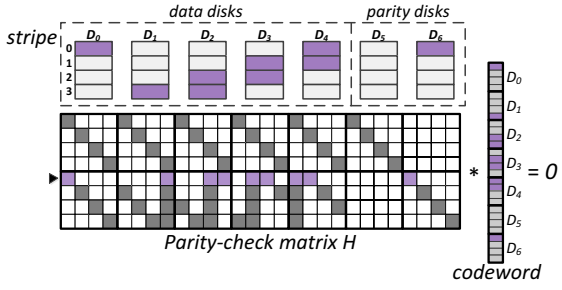


Fig. 5. Parity-check matrix of EVENODD code ($p = 5$). Each column of parity-check matrix corresponds to an element in the codeword (or a data block in a stripe), and each row represents a parity-check equation, which specifies a subset of elements with an XOR sum of 0. In this example, the selected row represents a parity-check equation with 9 elements, and the corresponding data blocks in the stripe have an XOR sum of 0.

Any XOR-based erasure codes can be defined with a parity-check matrix. Fig.5 shows the parity-check matrix of EVENODD code when $p = 5$. In this example, the construction of parity-check matrix is based on the encoding of EVENODD (as shown in Fig.1). Each encoding equation corresponds to a row in the parity-check matrix. In this row, several columns are set to 1 and the others are set to 0. The columns with 1's corresponds the elements appeared in the encoding equation. For example, the selected row in Fig.5 represents the encoding equation $C_{0,6} = C_{3,1} \oplus C_{2,2} \oplus C_{1,3} \oplus C_{0,4} \oplus C_{0,0} \oplus C_{3,2} \oplus C_{2,3} \oplus C_{1,4}$.

The parity-check matrix is not unique for erasure codes. For example, we can use row switching (swap two rows) or row addition (replace a row by the sum of that row and another row) to generate a new parity-check matrix, which is valid for erasure codes as well. Therefore, in some scenarios, we need to select a proper parity-check matrix to achieve high efficiency. This problem is discussed in Section III-C.

B. Encoding with Parity-check Matrix

To encode with parity-check matrix, we simply use the encoding equations. The elements in each encoding equation are represented by the 1's in each row of the parity-check matrix.

For some erasure codes, a parity element participates in the generation of some other parity elements, thus the order of calculating parity elements should be arranged properly. Due to this reason, the encoding is performed as below,

(1) Explore the dependencies of all parity elements via a graph model. For two parity elements A and B, if the generation of A depends on B, then we connect B to A with a directed edge.
(2) Sort all parity elements in topological order. A parity element should be placed after all other parity elements it depends on.
(3) Calculate all parity elements in the order of step 2 according to the encoding equations.

There is no significant difference on encoding between PCM and existing methods (matrix/non-matrix based approaches). It is because that the encoding procedure of these approaches are all based on the encoding equations defined by the coding scheme.

C. Decoding with Parity-check Matrix

To decode with parity-check matrix (denoted as H), we first divide H into two sub-matrices H_S and H_L. H_S contains the columns corresponding to the surviving disks, and H_L contains the columns corresponding to the lost disks. According to the property of H (the dot product of each row and the codeword is always zero), the product of H_S and the survivors is equal to the product of H_L and the lost data. We give an example to illustrate the relation of equality in Fig.6, which is using EVENODD to decode the two lost disks D_0 and D_2.

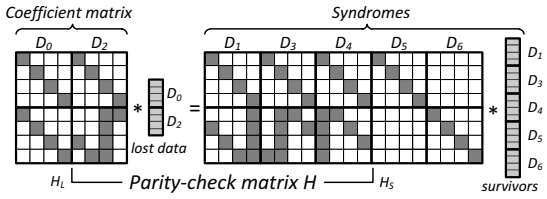


Fig. 6. Example of decoding 2 lost disks on D_0 and D_2 for EVENODD code ($p = 5$). The parity-check matrix H is divided into H_L and H_S, where H_L consists of the columns of H corresponding to the lost data, and H_S contains the columns corresponding to the survivors. The relation of equality in this figure is ensured by the property of parity-check matrix. Using this equation, the lost data can be calculated by $H_L^{-1} \cdot H_S \cdot survivors$.

In Fig.6, the right side is the product of H_S and the survivors, which is denoted as *syndrome*. The left side is the product of H_L and the lost data. Obviously the lost data can be calculated by multiplying the (generalized) inverse of H_L

(denoted as H_L^{-1}) with the syndromes. A typical algorithm to calculate generalized inverse of a bit matrix is presented in [12].

Given the parity-check matrix of an erasure code (denoted as H) and the list of failed disks, the decoding procedure abides the following steps,

(1) Divide H into two parts H_S and H_L. The columns of H_L correspond to the lost data, and the columns of H_S correspond to the surviving data.
(2) Calculate the syndromes by multiplying H_S with the survivors.
(3) Calculate the lost data by multiplying H_L^{-1} with the syndromes.

From the encoding/decoding algorithms of PCM, the efficiency is highly related to the number of 1's in the parity-check matrix, because it is proportional to the number of XORs in both encoding equations and the calculation of syndromes. For some erasure codes, the parity-check matrix constructed by their original encoding equations is not sparse, so we propose an optimization technique in Section III-D1 to address this problem.

D. Optimization Techniques

In this subsection, we introduce three optimization techniques to improve the decoding efficiency of our PCM approach. The first technique generates a sparse parity-check matrix, the second technique extends parity-check matrix to improve the efficiency for EVENODD and STAR codes, and the last technique improves the decoding efficiency for erasure codes tolerating three or more disk failures.

1) Constructing Sparse Parity-check Matrix: As illustrated in Section III-A, the parity-check matrix is not unique. Among various choices of parity-check matrices, a sparse one can improve the encoding and decoding efficiency. Our first optimization is to construct a sparse parity-check matrix.

A class of valid constructions of parity-check matrix can be obtained by row addition, which preserves the property that the dot product of each row and the codeword is zero. Here we present an algorithm which iteratively performs row additions to achieve this goal. Assume p_i is the parity element calculated by the encoding equation in the ith row, we generate the sparse matrix by Algorithm 1,

Particularly, this optimization method is useful for CRS code [3], which can reduce both encoding and decoding complexity compared to the original implementation of CRS code.

2) Extension of Parity-check Matrix: Some codes such as EVENODD and STAR have *redundancy problem* in their parity-check matrix, due to the utilization of internal values in their encoding procedure. We take EVENODD with $p = 5$ as an example, whose encoding process is shown in Fig.1. In specialized non-matrix based approach for EVENODD, a special element S (shown in Fig.1(b)) is calculated once in encoding, and added to all diagonal parity elements. However, when constructing parity-check matrix using the algorithm in Section III-A, S is actually calculated multiple times in the encoding algorithm of PCM, thus decreases the efficiency. From the original parity-check matrix for EVENODD (shown in Fig.7(a)), we can see that the matrix is not sparse.

Algorithm 1: Generating sparse parity-check matrix

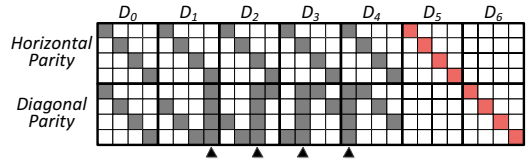
```
xors[i]: the number of XORs to calculate p_i.
from[i]: the other row that is added to the ith row.
visit[i] : whether the ith row is calculated.
mw: the number of rows in a parity-check matrix.

update = 1;
while update = 1 do
    update = 0;
    for i = 0 to mw do
        xors[i] = number of 1's in the ith row minus 2;
        from[i] = -1;
        visit[i] = 0;
    end
    while there exists some visit[i] = 0 do
        x = row ID which satisfies visit[x] = 0 and has
        minimum ones[x];
        for i = 0 to mw do
            if i ≠ x and visit[i] = 0 then
                dis = hamming distance of row i and x;
                if dis − 2 ≤ xors[i] then
                    xors[i] = dis − 2;
                    from[i] = x;
                end
            end
        end
        if from[x] ≠ −1 then
            add row from[x] to row x;
            update = 1;
        end
        visit[x] = 1;
    end
end
```

In our approach, we extend the parity-check matrix with additional rows/columns to support the usage of internal elements, which can improve the sparsity of the parity-check matrix. To achieve this goal, first we add a column in the parity-check matrix to represent the internal elements S. Then, all the diagonal parity elements can be calculated from S directly (set the corresponding column to 1), which reduces the number of 1's in each row. Finally, we add an additional parity-check equation to the matrix to compute S from the data elements. The extended parity-check matrix is shown in Fig.7(b). Using this optimization, we reduce seven 1's in the matrix.

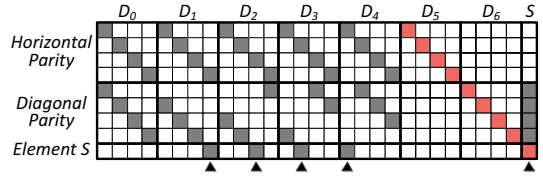
In addition, two scenarios should be considered in the decoding process. First, since S is not actually stored on the disks, we need to allocate an additional buffer to store S during encoding and decoding. Second, in decoding procedure, S is regarded as a lost element, thus is computed from the syndromes by the decoding matrix H_L^{-1}.

3) Iterative Reconstruction: The iterative reconstruction is derived from the observation that, when decoding three or more lost disks, existing optimization may not take the most advantage of recovered elements.

To improve the efficiency of decoding, we do not recover



(a) Original parity-check matrix with redundancy problem caused by repeatedly calculating special element S in each diagonal parity equation. The selected columns corresponds to the elements involved in the calculation of S. Due to the encoding procedure of EVENODD, these columns are set to 1 in for diagonal parity elements, which makes this matrix dense. When using this matrix in encoding, the efficiency is lower than the specialized encoding algorithm of EVENODD.



(b) Extended parity-check matrix with an additional row to calculate S, and an additional column with element S for each diagonal parity element. When using this parity-check matrix for encoding, S is only calculated once by the last row of this matrix, instead of being calculated multiple times using the original parity-check matrix as shown in Fig.7(a). Obviously, this matrix is sparser than the original one.

Fig. 7. Parity-check matrix for EVENODD ($p = 5$).

all the failed disks at one time. Instead, we recover them one by one, thus several iterations should be performed. Once the data on some disk is recovered, they are regarded as survivors, then we redo the decoding procedure to handle the rest of the failed disks. According to the decoding procedure in III-C, the decoding matrix H_L^{-1}

and the decoding of Jerasure is implemented by matrix based approach.

TABLE III
DETAILS OF THE TEST PLATFORM

Name	Machine 1	Machine 2
CPU	Intel Core E5-2620	Intel Core i5-4430
Memory	12GB	32GB
L1/L2/L3 Cache	32KB/256KB/15MB	32KB/256KB/6MB
OS	Ubuntu 14.04 64-bit	Ubuntu 14.04 64-bit

The performance evaluation is conducted on two different machines as described in Table III. No other irrelevant applications are running during the experiments. We use the configuration of $n = 16$ in our evaluation.

To evaluate the decoding performance, we first create a 256M memory region with randomly generated content, then encode it into 16 pieces. To simulate disk failures, we randomly drop m of them. Finally we decode the original information using the remaining pieces. We vary the packet size[4] from 1KB to 64KB to find the peak speed of the two implementations. Our results are averaged over 1000 repeated experiments.

B. Numerical Results on Decoding Performance

1) Comparison with Matrix based Decoding Approach: First, we compare PCM with matrix based approach on decoding complexity, which is shown in Fig.8. We summarize the improvements over matrix based method in Table IV.

For erasure codes tolerating triple disk failures including STAR, Triple-Star and TIP codes, PCM reduces more than 78.2% decoding cost, and the improvement can be even higher when n increases. We also observe that, by using matrix based approach, the decoding complexity is nearly proportional to the array size. In comparison, it is less than 4 in all scenarios when using PCM approach.

Then, for RAID-6 codes for correcting double disk failures, PCM decreases the decoding complexity of EVENODD and Liberation by up to 45.0% and 12.0%, respectively. For CRS codes, PCM saves the decoding complexity by up to 18.27%. In a few cases, PCM is slightly worse than the matrix based approach. It is reasonable because our method calculates syndromes, which may bring additional overhead when array size is small.

2) Comparison with Non-matrix based Decoding Approach: EVENODD, STAR and Triple-Star codes can be implemented via non-matrix based approach. Fig.9 shows the decoding complexity for these codes. It is clear that PCM provides better decoding performance for all codes. In this comparison, PCM achieves better effects when the array size (n) is small.

C. Experimental Results on Decoding Performance

Fig.10 shows the comparison between PCM and Jerasure in terms of decoding speed by using various erasure codes. We choose the peak speed among all packet sizes, and the improvements of PCM over Jerasure are summarized in Table V. Clearly, PCM is much more efficient than Jerasure. For STAR, Triple-Star and TIP codes, PCM accelerates the decoding process by up to a factor of $2.4\times$, $2.7\times$ and $2.6\times$, respectively. For EVENODD, the speedup of PCM over Jerasure is approximate $1.5\times$.

[4]Packet size: the size of an element (or a chunk).

TABLE V
COMPARISON BETWEEN PCM AND JERASURE IN TERMS OF DECODING SPEED IN A 16-DISK ARRAY

Coding Method	Machine 1			Machine 2		
	Speed (GB/s)		Speedup	Speed (GB/s)		Speedup
	PCM	Jerasure		PCM	Jerasure	
EVENODD	4.70	3.11	1.51×	7.06	4.74	1.49×
Liberation	4.77	4.47	1.07×	7.20	6.62	1.09×
STAR	2.94	1.23	2.40×	4.44	2.00	2.22×
Triple-Star	3.07	1.14	2.70×	4.68	1.84	2.54×
TIP-code	2.94	1.14	2.59×	4.38	1.65	2.66×
CRS (m=2)	4.09	3.34	1.22×	6.06	5.25	1.16×
CRS (m=3)	2.75	2.33	1.18×	3.98	3.66	1.09×
CRS (m=4)	2.13	1.80	1.18×	3.08	2.83	1.09×

D. Further Discussion

1) Space Overhead: PCM approach can be implemented with low space overhead. The space overhead includes memory size to store the parity-check matrix and related buffers for optimizing decoding efficiency.

Fig.12 shows the space overhead of PCM we measured in terms of memory consumption (in MB). From this figure, we can see that PCM consumes up to 1MB memory, which is negligible ($10\text{MB}/12\text{GB} = 0.08\%$).

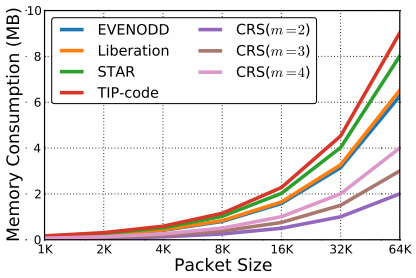


Fig. 12. Space overhead of PCM approach in terms of memory consumption for various XOR-based codes (Total memory size is 12GB).

2) Encoding Complexity: Except for the improvements on decoding performance, PCM can enhance the encoding efficiency for several erasure codes as well. In this part, we present the numerical results to demonstrate the efficiency of PCM on encoding performance.

We use the ***encoding complexity*** [38] as the metric in our mathematical analysis, which is defined as the average number of XOR operations to encode a data element. Because non-matrix based approach cannot be applied with various erasure codes, it is excluded in our evaluation. Here we only present the results for PCM and matrix based approaches by using some typical erasure codes, such as EVENODD, STAR and CRS codes.

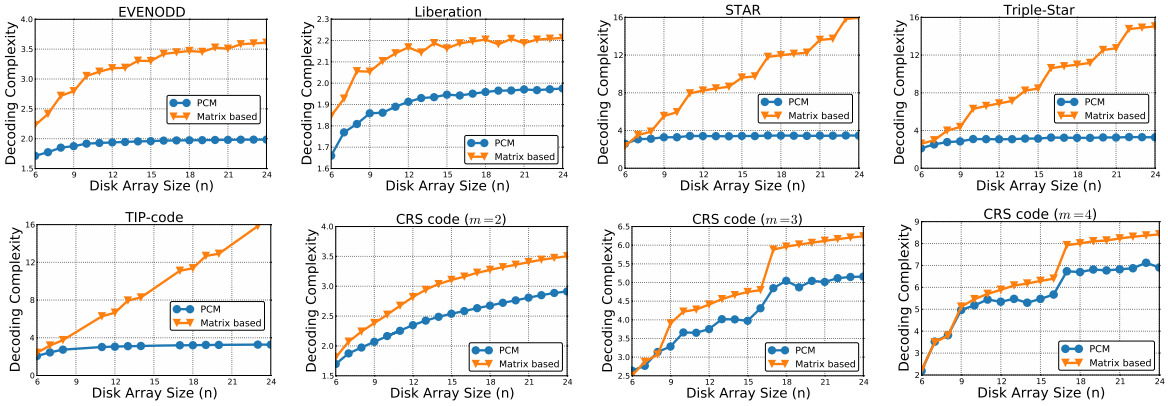
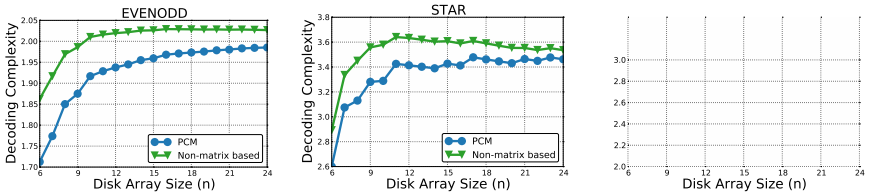
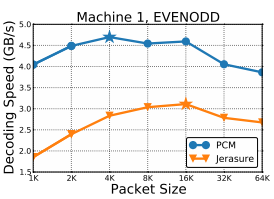


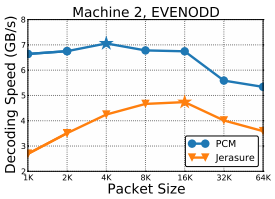
Fig. 8. Comparison between PCM and matrix based approach in terms of decoding complexity.

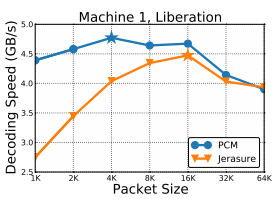
TABLE IV
IMPROVEMENT OF PCM OVER MATRIX BASED METHOD IN TERMS OF DECODING COMPLEXITY

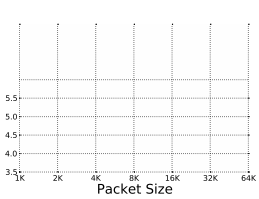
Coding Method	n (the number of disks in a disk array)									
	6	8	10	12	14	16	18	20	22	24
Typical RAID-6 Erasure Codes										
EVENODD	23.03%	31.93%	37.16%	39.07%	40.86%	42.44%	43.11%	43.87%	44.63%	44.97%
Liberation	10.11%	12.03%	11.49%	11.77%	11.58%	11.17%	11.13%	10.95%	10.74%	10.70%
Array Codes Tolerating Triple Disk Failures										
STAR	0.00%	19.29%	44.64%	58.52%	60.76%	64.96%	71.12%	72.00%	74.88%	78.31%
Triple-Star	18.59%	30.38%	50.80%	55.26%	61.78%	69.32%	70.65%	73.94%	77.60%	78.20%
TIP-code	13.89%	26.89%	/	54.12%	62.17%	/	71.77%	74.89%	/	79.61%
CRS Codes under Various Configurations										
CRS ($m=2$)	6.15%	11.91%	14.15%	16.68%	18.05%	18.23%	18.27%	17.75%	17.23%	16.84%
CRS ($m=3$)	−5.32%	−2.17%	13.22%	14.91%	13.92%	10.13%	15.35%	16.81%	16.97%	17.29%
CRS ($m=4$)	1.99%	−0.60%	5.15%	9.26%	14.15%	11.31%	16.34%	16.76%	17.28%	17.94%




Machine 1, EVENODD
Decoding Speed (GB/s)
Packet Size
PCM
Jerasure


Machine 2, EVENODD
Decoding Speed (GB/s)
Packet Size
PCM
Jerasure


Machine 1, Liberation
Decoding Speed (GB/s)
Packet Size
PCM
Jerasure


Packet Size

REFERENCES

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

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

[3] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," International Computer Science Institute, Tech. Rep. TR-95-048, August 1995.

[4] V. Bohossian, C. Fan *et al.*, "Computing in the RAID: A reliable array of independent nodes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 2, pp. 99–114, 2001.

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

[6] 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 USENIX FAST'04*, San Francisco, CA, March 2004.

[7] P. Corbett and A. Goel, "Triple parity technique for enabling efficient recovery from triple failures in a storage array," September 2011, US Patent 8,015,472.

[8] G. L. Feng, R. Deng, F. Bao, and J. C. Shen, "New efficient MDS array codes for RAID part I: Reed-Solomon-like codes for tolerating three disk failures," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1071–1080, 2005.

[9] ——, "New efficient MDS array codes for RAID part II: Rabin-like codes for tolerating multiple (≥ 4) disk failures," *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 1473–1483, 2005.

[10] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[11] J. Hafner, "WEAVER codes: Highly fault tolerant erasure codes for storage systems." in *Proceedings of the USENIX FAST'05*, San Francisco, CA, December 2005.

[12] ——, "HoVer erasure codes for disk arrays," in *Proceedings of the IEEE/IFIP DSN'06*, Philadelphia, PA, June 2006.

[13] J. Hafner *et al.*, "Matrix methods for lost data reconstruction in erasure codes," in *Proceedings of the USENIX FAST'05*, San Francisco, CA, December 2005.

[14] C. Huang, M. Chen, and J. Li, "Pyramid Codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proceedings of the IEEE NCA'07*, Cambridge, MA, July 2007.

[15] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Transactions on Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[16] C. Huang, J. Li, and M. Chen, "On optimizing XOR-based codes for fault-tolerant storage applications," in *Proceedings of the ITW'07*, Tahoe City, CA, September 2007.

[17] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin *et al.*, "Erasure coding in windows azure storage," in *Proceedings of the USENIX ATC'12*, Boston, MA, June 2012.

[18] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-Code: A new RAID-6 code with optimal properties," in *Proceedings of the ICS'09*, Yorktown Heights, NY, June 2009.

[19] N. Joukov, A. M. Krishnakumar, C. Patti, A. Rai, S. Satnur, A. Traeger, and E. Zadok, "RAIF: Redundant array of independent filesystems," in *Proceedings of the MSST'07*, San Diego, CA, September 2007.

[20] M. Li and P. Lee, "STAIR codes: a general family of erasure codes for tolerating device and sector failures in practical storage systems." in *Proceedings of the USENIX FAST'14*, Santa Clara, CA, February 2014.

[21] S. Li *et al.*, "Exploiting decoding computational locality to improve the I/O performance of an XOR-coded storage cluster under concurrent failures," in *Proceedings of the SRDS'14*, Nara, Japan, October 2014.

[22] M. Luby, M. Mitzenmacher *et al.*, "Practical loss-resilient codes," in *Proceedings of the STOC'97*, Paso, TX, May 1997.

[23] J. Luo, L. Xu, and J. Plank, "An efficient XOR-scheduling algorithm for erasure codes encoding," in *Proceedings of the IEEE/IFIP DSN'09*, Estoril, Lisbon, June 2009.

[24] S. Muralidhar, W. Lloyd *et al.*, "F4: Facebooks warm BLOB storage system," in *Proceedings of the USENIX OSDI'14*, Broomfield, CO, October 2014.

[25] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for Redundant Arrays of Inexpensive Disks (RAID)," in *Proceedings of the ACM SIGMOD'88*, Chicago, IL, June 1988.

[26] J. Plank, "A new minimum density RAID-6 code with a word size of eight," in *Proceedings of the IEEE NCA'08*, Cambridge, MA, July 2008.

[27] ——, "The RAID-6 liberation codes," in *Proceedings of the USENIX FAST'08*, San Jose, CA, February 2008.

[28] J. Plank, M. Blaum, and J. Hafner, "SD codes: Erasure codes designed for how storage systems really fail," in *Proceedings of the USENIX FAST'13*, San Jose, CA, February 2013.

[29] J. Plank *et al.*, "A performance evaluation and examination of open-source erasure coding libraries for storage." in *Proceedings of the USENIX FAST'09*, San Francisco, CA, February 2009.

[30] J. Plank and K. Greenan, "Jerasure: A library in c facilitating erasure coding for storage applications–version 2.0," Technical Report UT-EECS-14-721, University of Tennessee, Tech. Rep., 2014.

[31] J. Plank and C. Huang, "Tutorial: Erasure coding for storage applications," in *Slides presented at USENIX FAST'13*, San Jose, CA, February 2013.

[32] J. Plank, M. Thomason *et al.*, "A practical analysis of low-density parity-check erasure codes for wide-area storage applications," in *Proceedings of the DSN'04*, Florence, Italy, June 2004.

[33] J. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for Fault-Tolerant network storage applications," in *Proceedings of the IEEE NCA'06*, Cambridge, MA, July 2006.

[34] K. Rashmi, N. Shah *et al.*, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proceedings of the SIGCOMM'14*, Snowbird, UT, June 2014.

[35] I. 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.

[36] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proceedings of the VLDB'13*, Riva del Garda, Italy, August 2013.

[37] Z. Shen and J. Shu, "HV code: An all-around mds code to improve efficiency and reliability of RAID-6 systems," in *Proceedings of the IEEE/IFIP DSN'14*, Atlanta, GA, June 2014.

[38] P. Subedi and X. He, "A comprehensive analysis of XOR-Based erasure codes tolerating 3 or more concurrent failures," in *Proceedings of the IPDPSW'13*, Cambridge, MA, May 2013.

[39] D. Tang, X. Wang, S. Cao, and Z. Chen, "A new class of highly fault tolerant erasure code for the disk array," in *Proceedings of the PEITS'08*, Guang Zhou, China, August 2008.

[40] C. Tau and T. Wang, "Efficient parity placement schemes for tolerating triple disk failures in RAID architectures," in *Proceedings of the AINA'03*, Xi'an, China, March 2003.

[41] Y. Wang, G. Li, and X. Zhong, "Triple-Star: A coding scheme with optimal encoding complexity for tolerating triple disk failures in RAID," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 3, pp. 1731–1472, 2012.

[42] Y. Wang and G. Li, "Rotary-code: Efficient mds array codes for raid-6 disk arrays," *IEEE Transactions on Computers*, vol. 8, no. 12, pp. 1917–1926, 2009.

[43] C. Wu *et al.*, "HDP code: A Horizontal-Diagonal parity code to optimize I/O load balancing in RAID-6," in *Proceedings of the DSN'11*, 2011.

[44] C. Wu, S. Wan, X. He, Q. Cao, and C. Xie, "H-Code: A hybrid MDS array code to optimize partial stripe writes in RAID-6," in *Proceedings of the IPDPS'11*, Anchorage, Alaska, May 2011.

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

[46] 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.

[47] Y. Zhang, C. Wu *et al.*, "TIP-code: A three independent parity code to tolerate triple disk failures with optimal update complexity," in *Proceedings of the DSN'15*, Rio de Janeiro, Brazil, June 2015.