2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)

Optimizing the Parity-Check Matrix for Efficient Decoding of RS-based Cloud Storage Systems

Junqing Gu[1], Chentao Wu[1]*, Xin Xie[1], Han Qiu[1], Jie Li[1], Minyi Guo[1], Xubin He[2], Yuanyuan Dong[3], and Yafei Zhao[3]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2]Department of Computer and Information Sciences, Temple University, Philadelphia, United States
[3]Alibaba Group, Hangzhou, China
*Corresponding Author: wuct@cs.sjtu.edu.cn

***Abstract*—In large scale distributed systems such as cloud storage systems, erasure coding is a fundamental technique to provide high reliability at low monetary cost. Compared with the traditional disk arrays, cloud storage systems use an erasure coding scheme with both flexible fault tolerance and high scalability. Thus, Reed-Solomon (RS) Codes or RS-based codes are popular choices for cloud storage systems. However, the decoding performance for RS-based codes is not as good as XOR-based codes, which are optimized via investigating the relationships among different parity chains or reducing the computational complexity of matrix multiplications. Therefore, exploring an efficient decoding method is highly desired.**

To address the above problem, in this paper, we propose an Advanced Parity-Check Matrix (APCM) based approach, which is extended from the original Parity-Check Matrix based (PCM) approach. Instead of improving the decoding performance of XOR-based codes in PCM, APCM focuses on optimizing the decoding efficiency for RS-based codes. Furthermore, APCM avoids the matrix inversion computations and reduces the computational complexity of the decoding process. To demonstrate the effectiveness of the APCM, we conduct intensive experiments by using both RS-based and XOR-based codes under cloud storage environment. The results show that, compared to typical decoding methods, APCM improves the decoding speed by up to 32.31% in the Alibaba cloud storage system.

***Index Terms*—Erasure Codes, Parity-Check Matrix, Decoding, Cloud Storage, Performance Evaluation**

I. Introduction

In large scale distributed systems such as cloud storage systems, with high probabilities of disk failures [35] [23], data reliability is a critical issue for tremendous disk devices. Typically, data redundancy methods (i.e., replication and erasure coding) are widely utilized in storage systems to obtain various capabilities of fault tolerance [14] [34] [21] [31]. In cloud storage systems like Hadoop and Microsoft Azure [14], to trade off the performance, reliability and monetary cost, erasure coding schemes are used to store cold data in general, because they can tolerate triple or more disk failures with low storage overhead.

Typically, erasure coding methods in cloud storage systems can be categorized into two types, RS-based codes [32] [14] [38] and XOR-based codes [45] [37] [26] [9] [49]. RS-based codes are variants of Reed-Solomon (RS) codes [32], which are based on the Galois Field (GF) computations. GF is a complex computation operation in mathematical area, so Intel releases its ISA-L library [1] to optimize the instructions' execution of GF calculations, which can improve the encoding/decoding performance of RS-based codes. The flexibilities of RS-based codes are excellent, and most RS-based codes can generate arbitrary number of parity blocks[1] by any number of data blocks (i.e., RS code, Clay Codes). Another type is XOR-based codes, which are generated via XOR operations. An XOR operation is simple and basic calculation for logic circuits. Although XOR-based codes can achieve fast computation on encoding/decoding[2] [24] [5] [16] [36], they are lack of agilities in complex cloud storage environment. For example, EVENODD code [2] is a typical solution for RAID-6 array to tolerate double disk failures, but it cannot tolerate triple disk failures by simply adding a new parity into the layout[3].

To accelerate the decoding process of different erasure codes in cloud storage systems, several methods are proposed in the past two decades. For RS-based codes, speeding up the GF calculations is a general choice [27]. On the other hand, to improve the reconstruction efficiency of XOR-based codes, a few solutions are proposed via investigating the relationships among various parity chains, and a part of them enhance the matrix multiplications as well. Specifically, recovery bandwidth is a significant aspect for cloud storage in data centers [38] [29] [14], and a few literatures focus on this area to minimize the transfer network I/Os [20].

However, compared to XOR-based codes, the decoding process of RS-based codes are more difficult to be improved. There are two main reasons. First, most decoding methods are not suitable for RS-based codes. For example, investigating the relationships among various parity chains is an efficient method to increase the decoding performance, which needs different types of parity chains share a set of blocks. Due to the special layout of RS-based codes, we cannot find any shared blocks in the reconstruction process. Second, existing methods like Intel ISA-L optimize the decoding in the instruction level, which ignores the complex matrix multiplications in the decoding process. It is still a potential direction to decrease the overhead of decoding.

[1]In this paper, a *block* is used to represent *data element* or *chunk* which is the basic access unit in erasure codes [6] [33].

[2]For example, Cauchy-RS codes utilize XOR operations instead of GF to accelerate the encoding/decoding process [4] [28].

[3]As we know, the related solution to tolerate triple disk failures is called "STAR" code [13].

1530-2075/19/$31.00 ©2019 IEEE
DOI 10.1109/IPDPS.2019.00063

To address the above problem, in this paper, we propose an Advanced Parity-Check Matrix based (APCM) approach for improving the decoding performance of RS-based codes. The basic idea of APCM is to minimize the computation cost of matrix multiplications in the decoding process, so it can enhance the efficiency of any matrix-based decoding methods, especially for RS-based codes.

The contribution of our work includes:

1) We propose an Advanced Parity-Check Matrix based (APCM) approach, to provide high decoding performance for RS-based codes.
2) Our intensive study shows that comparing with traditional approaches, APCM achieves better decoding performance under various erasure codes.

The rest of paper is organized as follows. In Section II, we introduce related work and our motivation. In Section III, the design of APCM and corresponding decoding optimization techniques are illustrated in detail. The evaluation is presented in Section IV and the conclusion of our work is in Section V.

II. Related Work And Motivation

Erasure coding is a typical technique to guarantee data reliability with low storage cost, which is widely used in modern cloud storage systems. Erasure codes can be divided into two categories. One class is based on XOR operations, known as XOR-based codes. Typical XOR-based codes include RDP [7] [43], EVENODD [2], STAR code [13], Triple-Star code [39], H-Code [40], HDP Code [41], TIP-Code [48], EH-Code [15], etc. Another type is based on arithmetic calculation over Galois Field (GF). Reed-Solomon (RS) code [32] [25] is a classic GF computation code and can tolerate any numbers of disk failures by setting suitable parameters. Other RS-based codes include Local Reconstruction Codes (LRC) [14], Locally Repairable Code (LRC) [22], Clay codes [38], etc.

In this section, we discuss the related work of the existing decoding schemes and the decoding problem for RS-based codes. To facilitate our discussion, we summarize the symbols used in this paper in Table I.

A. Desired Decoding Methods in Cloud Storage Systems

According to two types of erasure codes, several decoding methods are proposed in the past two decades. The following aspects on decoding methods are desired for cloud storage systems,

- *High Performance*. The decoding method can accelerate the recovery process in terms of low computational cost, small number of reconstruction I/Os, etc.
- *Low Complexity in Implementation*. During the decoding process, a general recovery policy should be provided for various failure scenarios (e.g., single/double/triple disk failures).
- *High Flexibility*. The decoding method can be applied to various erasure codes, especially for all RS-based codes.

Typically, the decoding methods can be divided into two classes, non-matrix based and matrix based approaches, which are illustrated in detail as below.

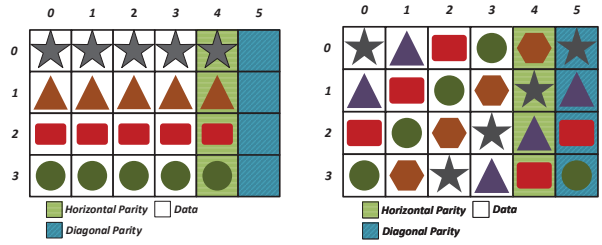
TABLE I
Symbols Used In This Paper

Symbols	Description
n	number of disks in a disk array
r	number of parity disks
k	number of data disks ($k = n - r$)
lr	number of local parity disks ($lr + gr = n - k$) in LRC codes
gr	number of global parity disks ($lr + gr = n - k$) in LRC codes
w	word size
p	a prime number
$B_{i,j}$	an element at the ith row and jth column
D_L	a disk array consists of lost data
D_S	a disk array consists of surviving data
H	a parity-check matrix (matrix size is $r \times n$)
H_L	a sub-matrix of H (columns corresponds to the lost data)
H_S	a sub-matrix of H (columns corresponds to the surviving data)
P_i	a data block which stores the ith parity block
D_i	a data block which stores the ith data block
$\oplus$	an add operation in Galois Field (equal to XOR)
$\otimes$	a multiply operation in Galois Field
H_T	an advanced parity-check matrix after a matrix transformation
H_{TL}	a sub-matrix of H (columns corresponds to the lost data)
H_{TS}	a sub-matrix of H (columns corresponds to the surviving data)
Q	a matrix to convert matrix H to H_T (which is $r \times r$)

B. Non-matrix based Decoding Methods

The key idea of the non-matrix based approach is reducing the I/O cost via investigating the relationships among different parity chains. These methods are used for XOR-based codes.

1) General Approach: Here we take RDP [7] code as an example. As shown in Fig.1, RDP code is a typical RAID-6 code [11] to tolerate concurrent disk failures of any two disks in an array. RDP supports an array consists of $p+1$ disks ($p-1$ data disks and 2 parity disks), where p is a prime number. Fig.1 and Fig.2 show the encoding and decoding processes of RDP code ($p = 5$), respectively.



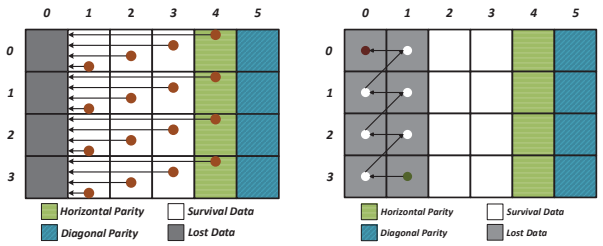
(a) Horizontal parity coding (e.g., $B_{0,4} = B_{0,0} \oplus B_{0,1} \oplus B_{0,2} \oplus B_{0,3}$)

(b) Diagonal parity coding (e.g., $B_{0,5} = B_{0,0} \oplus B_{3,2} \oplus B_{2,3} \oplus B_{1,4}$)

Fig. 1. Encoding of RDP ($p = 5$) can be represented by a $p \times (p+2)$ matrix, where the first p delegate the data columns and the last two columns delegate the parity columns. The pictures use different icon shapes to denote different parity chains.

The encoding process of RDP code is shown in Fig.1. There are two types of parities, which are generated by data blocks via XOR operations. One type of the parity elements is called the row parity[4] (shown in Fig.1(a)). The other is called diagonal parity (shown in Fig.1(b)).

[4]It is also referred to as "horizontal parity".



(a) When a single disk needs to be recovered, the horizontal parity disks are directly used as well as the other surviving data disks to recover the lost disk.

(b) When disks 0 and 1 fail, the recovery process follows the order of a recovery chain, which is started at the starting point $B_{3,1}$ (marked as green) and end at the end point $B_{0,0}$ (marked as red). Then we obtain two recovered disks.

Fig. 2. Decoding process of RDP ($p = 5$), Figure(a) shows the situation when a single disk fails and Figure(b) illustrates the decoding process when double disks fail.

In the decoding process (shown in Fig.2), there are two cases for RDP code. If only single disk fails, we can use either row or diagonal parity to recover the lost data. If double disks fail, both the row and the diagonal parities participate in the reconstruction process (shown in Fig.2).

2) Optimized Approaches: One optimized approach is decreasing the I/O cost [17] [30] [19]. To reduce the I/O cost for reconstruction process, *Xiang et al.* propose a hybrid recovery method called RDOR [44] by exploiting the replicated blocks among various parity chains during single disk reconstruction. Windows Azure Storage (WAS) publishes Local Reconstruction Codes(LRC) [14] [3] [10] to save the I/O overhead, where local parities are added into the original RS codes. During the decoding process, only valid nodes in the corresponding local group are needed. This saves a lot of I/O operations and decreases the bandwidth requirement. Another method is to mix two different erasure codes [42], which guarantees low recovery cost of hot data and small storage overhead of cold data simultaneously.

Several approaches are presented to save the bandwidth in the recovery process. *Li et al.* introduce Repair Pipelining [18], which is based on Partial-Parallel-Repair (PPR) proposed by *Mitra et al* [20]. PPR distributes the transmission overhead to several intermediate nodes, thus reduces the total transmission time. Another method is using a kind of codes named Minimum storage Regeneration Codes (MSR) [8] [46] [12] [38], which reduces network bandwidth, but causes more I/Os and computation cost.

However, although non-matrix based approach can achieve high decoding performance, their flexibilities are restricted, and we need to design specialized reconstruction algorithm for each erasure code as well.

C. Matrix based Decoding Methods

The basic idea of matrix based decoding methods is reconstruct the lost via the generator matrix. They are universal decoding approaches for all erasure codes (including both RS-based and XOR-based codes).

1) General Approach: Here we give the decoding process in Fig.3, the corresponding approach called matrix-based method. The generator matrix G is the key in both encoding and decoding processes, and data is represented as a data vector. The parity blocks are generated by the product multiplication of the data vector and the generator matrix. Fig.3 and Fig.4 show the encoding process of RDP [7] code ($p = 5$) and RS code ($k = 5, r = 2$), respectively.

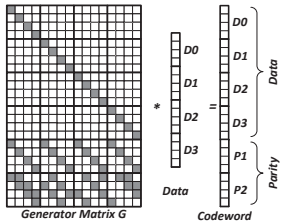


Fig. 3. Matrix-based encoding procedure for RDP code ($p = 5$). The generator matrix G transforms the date vector into a codeword (the encoded data contains original data and parity). The codeword is computed by the dot product of the generator matrix G and the data vector. Since G is a bit matrix, the add operation is equal to XOR and the multiply operation is equal to bitwise AND.

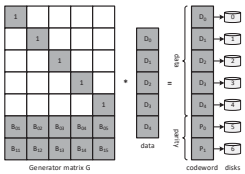


Fig. 4. Matrix-based encoding procedure for RS code with ($k = 5, r = 2$). G is a *coefficient matrix* based on $GF(2^w)$. Each block in the codeword is computed by the dot product of the corresponding row in G and the data vector, where addition and multiplication are both Galois operation.

The decoding process of RS-based codes consists of the following steps. First it constructs a new matrix G' which contains the rows corresponding to the surviving data blocks of the generator matrix, such that the product of G' and data blocks is equal to the vector that consists of the surviving data. Then invert the matrix G' to get the decoding matrix $\hat{G}$. The product of $\hat{G}$ and surviving data blocks is equal to the original data, so we can recover the lost data by matrix multiplications of $\hat{G}$ and the surviving data.

For RS-based codes, the matrix based decoding method is universal for all erasure codes, which can be applied for XOR-based codes as well.The only difference among these codes is the generator matrix G. Obviously, this decoding method has higher flexibility and it's simple to be implemented with low cost. However, the computation cost of this method is usually very high.

2) Optimized Approaches: To decrease the computation cost during reconstruction, Intel develops Intelligent Storage Acceleration Library (ISA-L), which optimizes the instruction sets to accelerate computational speed for erasure codes. Besides ISA-L, Jerasure [24] does the similar function as well.

TABLE II
SUMMARY ON VARIOUS DECODING APPROACHES FOR ERASURE CODES

Name		RS-based Codes	XOR-based Codes	Performance	Flexibility	Complexity	Level
General non matrix based approach		×	√	low	low	high	program level
Optimized non matrix based approach (i.e., RDOR)		×	√	high	low	high	program level
General matrix based approach		√	√	low	high	low	program level
Optimized matrix based approach	PCM	×	√	medium	high	low	program level
	Jerasure	√	√	medium	high	high	program level
	ISA-L	√	×	high	high	low	instruction level
	APCM	√	√	high	high	low	program level

Another way to achieve fast computation is to optimize the encoding/decoding matrix, which reduces the complexity of its inversion. For example, Parity-Check Matrix (PCM) is proposed by *Zhang et al.* [47] to improve the decoding efficiency of XOR-based codes. It generates a Parity-Check Matrix (PCM) instead of the initial generator matrix, and the encoding and decoding processes by using PCM are presented in Fig.5 and Fig.6.

In Fig.5, the PCM is a $wr \times wn$ matrix, where w is the word length, r is the number of parities, and n is the number of all nodes. Obviously, compared to the original generator matrix G, PCM has a smaller size and simplifies the matrix calculations.

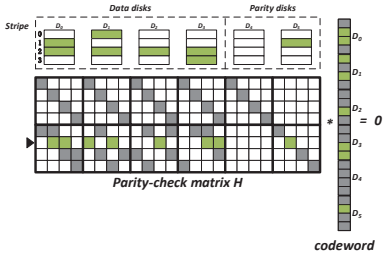


Fig. 5. Construction of a Parity-check Matrix.

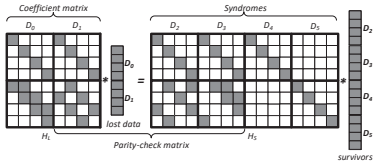


Fig. 6. Decoding Process of a Parity-check Matrix.

D. Our Motivation

We summarize the existing decoding methods in Table II, which shows that existing decoding approaches are not sufficient for RS-based codes. First, most non-matrix based and several matrix based (i.e., PCM) approaches cannot be applied for RS-based codes. Second, although ISA-L library can improve the calculation of the decoding process for RS-based codes, it is an instruction level enhancement and ignores the complex matrix multiplications.

In summary, existing decoding approaches are insufficient for RS-based codes in cloud storage systems, which motivates us to propose a new decoding approach called APCM.

III. ADVANCED PARITY-CHECK MATRIX BASED APPROACH

In this section, we introduce the design of our Advanced Parity-Check Matrix based (APCM) approach, which is a method to improve the decoding process.

A. Advanced Parity-Check Matrix (APCM)

In Section II, we introduce the Parity-Check Matrix (PCM), which is designed for XOR-based codes. In order to apply the PCM method for RS-based codes, we need to transfer the original bit matrix into a new matrix based on arithmetic calculations over Galois Field. Therefore, we propose a new matrix called advanced parity-check matrix (APCM).

Let $n = k + r$ be the total number of blocks which have k data blocks and r parity blocks, the dimension of an advanced parity-check matrix (APCM) for the RS-based code scheme is $r \times n$, which is similar to the structure of the original parity-check matrix. The structure of an APCM is shown in Fig.7. In an advanced parity-check matrix, each row can be assumed as a parity-check equation (see Equation 1) of one parity block, which means the dot product of the row and the corresponding codewords is always 0 (in order to obey the encoding equation). For example, the selected row (green) in Fig.7 is represented by the decoding equations as below, where $\otimes$ and $\oplus$ mean multiply and add operations in Galois Field, respectively.

Since only one parity participates in the calculation of each equation, the last r columns of the matrix compose an identity matrix, just like the example in Fig.7.

$$B_{11} \otimes D_0 \oplus B_{12} \otimes D_1 \oplus B_{13} \otimes D_2 \oplus B_{14} \otimes D_3 \oplus B_{15} \otimes D_4 \oplus 0 \otimes P_0 \oplus 1 \otimes P_1 = 0 \quad (1)$$

$$B_{11} \otimes D_0 \oplus B_{12} \otimes D_1 \oplus B_{13} \otimes D_2 \oplus B_{14} \otimes D_3 \oplus B_{15} \otimes D_4 = P_1 \quad (2)$$

Use row switching or row addition of matrix to change APCM do not influence the parity-check equations, because the right-hand side of the equation is always 0. Therefore,

when we guarantee that the sub-matrixes composed of arbitrarily r columns are full rank, we can convert the matrix and select a proper APCM matrix to simplify the computation. Any erasure coding which has the Maximum Distance Separable (MDS) property can satisfy this condition. And the codes who do not have the property (such as LRC) can simply reduce the size of sub-matrixes until the condition of full rank is guaranteed, the strategy is detailedly introduced in subsection III.E. The design of APCM are discussed in the following subsection.

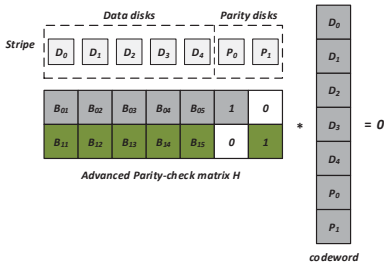
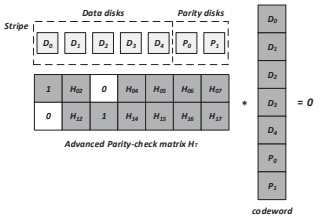
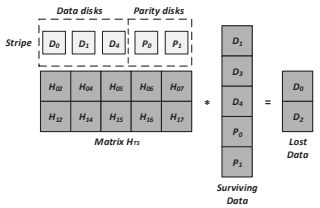


Fig. 7. The Construction of an Advanced Parity-Check Matrix.



(a) After row matrix transformations, we can divide new APCM H_T into two sub-matrixes H_{TL} and H_{TS}.



(b) Since the corresponding columns of lost data constitute an identity matrix H_{TL}, the lost data blocks and corresponding columns can be directly removed from data array D and matrix H_T.

Fig. 8. The Decoding Procedure of Advanced Parity-Check Matrix approach.

B. Encoding with APCM

To encode with advanced parity-check matrix, we simply use the encoding equations (e.g. Equation 2) to generate the parities. Each row of APCM can be converted into an encoding equation of a parity block, therefore we can utilize the matrix to generate the parity data. For example, the selected row (green) in Fig.7 can be convert from Equation 1 to Equation 2, which is because $+$ and $-$ operations are equivalent under the arithmetic over Galois Field and $1 \otimes P_1 = P_1$.

The encoding overhead of APCM is the same as the general approach in terms of both calculation and transmission. The storage overhead of APCM is lower than the general approach, because the size of APCM is $r \times n$, which is smaller than the general approach matrix $n \times k$ (usually k is bigger than r). The encoding procedures of both our APCM and the general approach are based on the predefined encoding equations, so their computational cost is the same as $O(r * k)$.

C. Decoding with APCM

Decoding with parity-check matrix needs to divide APCM into two sub-matrices (H_L and H_S), where H_L must be invertible[5]. Therefore, to apply APCM to RS-based codes, any sub-matrices constructed by r columns in APCM should be invertible.

In APCM approach, if the lost blocks are parity blocks, which means the sub-matrix H_L is an identity matrix, the lost data D_L can be directly calculated by $D_L = H_S \times D_S$. Therefore, if any potential H_L can be transformed into an identity matrix, we can reduce the matrix inversion operations and achieve high decoding performance. Therefore, a matrix transformation is the ***first step*** of decoding with an advanced parity-check matrix (denoted as H). Let the lost blocks $D_L = \{D_1, D_2, ..., D_r\}$, the matrix transformation process is described in Algorithm 1. The main idea of Algorithm 1 is to transform the specific columns, which corresponds to the lost blocks, into an identity matrix. The calculation process of the algorithm is similar to the Gauss-Jordan elimination[6].

The ***second step*** is to divide new APCM (denoted as H_T) into two sub-matrices H_{TS} and H_{TL}. H_{TS} contains the columns corresponding to the surviving blocks and H_{TL} contains the columns corresponding to the lost blocks. In the Galois Field, the subtraction operation is same as addition operation and equal to XOR operation. Therefore, the product of H_{TL} and lost data D_L is equal to the product of H_{TS} and surviving data D_S, which is denoted by $H_{TL} \times D_L = H_{TS} \times D_S$. Since H_{TL} is an identity matrix, D_L can be directly calculated by the equation. Suppose the lost data is $\{D_0, D_2\}$, an example decoding process is illustrated in Fig.8.

Algorithm 1 Matrix Transformation Process

```
k = 0;
for each i ∈ D_L do
  for each element j in row H[k] do
    H[k][j] = H[k][j]/H[k][i];
  end for
  for each row in matrix H do
    subtract the row H[r] with H[r][i] * H[k];
  end for
  k++;
end for
```

[5]in other words, the matrix should be full rank.

[6]Gauss-Jordan and APCM share the same mathematic principle, but APCM aims to transform specific columns to the identity matrix, which is different from Gauss-Jordan.

Because the right-hand side of the equation is 0, we can prove that the row matrix transformation cannot change the matrix's rank or the establishment of the equation.

***Theorem** 3.1:* Row Matrix transformation cannot change the establishment of the equation $H \times D = 0$.

Proof: The row matrix transformation for a $r \times k$ matrix H is mathematically equal to left multiply a $r \times r$ matrix P, since the left side of the equation is 0, we can obtain the following equation.

$$H \times D = 0 \tag{3}$$

$$Q \times H \times D = Q * 0 \tag{4}$$

$$Q \times H \times D = 0 \tag{5}$$

$$H_T \times D = 0 \tag{6}$$

Thus, we prove that the matrix transformation does not affect the establishment of the equation. ■

In summary, the decoding procedure of any erasure code abides by the following steps,

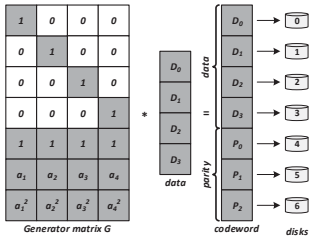
1) Transform H into a new APCM matrix H_T, in order to change the corresponding columns H_L into an identity matrix.
2) Divide H_T into two parts H_{TS} and H_{TL}. H_{TL} is constructed by the columns corresponding to the lost data, and H_{TS} is constructed by the columns corresponding to the surviving data.
3) Since H_{TL} is an identity matrix, we calculate the lost data by equation $D_L = H_{TS} \times D_S$.

D. Computational Complexity Analysis

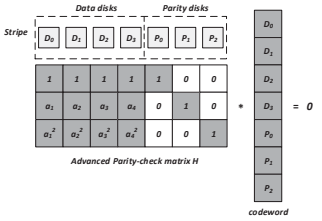
In this section, we compare the computational complexity of APCM decoding approach with other decoding approaches, including the general matrix based and PCM [47] approaches.

1) In the general matrix based approach, the decoding matrix is calculated from the encoding matrix. k rows in the encoding matrix correspond to k surviving blocks, which are chosen to construct a $k \times k$ matrix H. Then we calculate the inverse matrix H^{-1} to get the final decoding matrix. In summary, to generate the decoding matrix, the computational complexity is $O(k^3)$.
2) In the PCM decoding approach, matrix H is split into two sub-matrices, an $r \times k$ matrix H_S and an $r \times r$ matrix H_L. The decoding matrix is generated by function $H_L^{-1} \times H_S$, thus the computational complexity is $O(r^3 + r^2 \times k)$.
3) In the APCM decoding approach, since the matrix transformation operation can be regarded as the matrix multiplications operation, the calculation complexity is up to $O(r^2 \times k)$.

In conclusion, APCM decoding approach can reduce computational complexity during the reconstructions process. An increasing of data blocks (the parameter k) leads to a better performance while an increase of parity blocks (the parameter r) leads to a worse performance. Typically, r is much smaller than k, thus the computation cost of APCM is much lower than general matrix based approach.



(a) The traditional encoding matrix for the RS code.



(b) An APCM can be directly generated from the encoding matrix of RS code.

Fig. 9. The Construction of an RS Advanced Parity-Check Matrix.

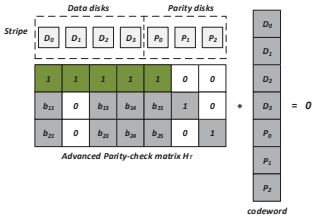
E. Case Studies

In this section, we use Reed-Solomon (RS) code [32] and Local Reconstruction Codes (LRC) [14] as examples to show the working process of APCM decoding approach. In our evaluation (Section IV), we also evaluate the effectiveness of APCM on two Minimum Storage Regenerating(MSR) codes, including Clay Codes [38] and Optimal-access MDS code [46]. The structure and calculation processes of the Advanced Parity-Check Matrix for MSR code is similar to RS code.

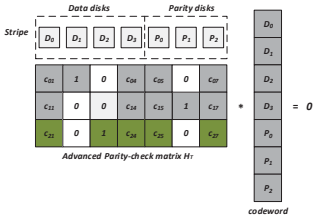
1) An example of RS code with APCM: Assume the number of parity disks is three $(r = 3)$, the total number of data disks is four $(k = 4)$, and the number of disks is $n = k + r = 7$. Fig.9(a) shows the matrix-based encoding procedure of a RS code. In order to generate an APCM matrix, the last r rows of the encoding matrix are utilized as a part of APCM (denoted by H_r). Since each row of H_r is a generation function of a parity block, an $r \times r$ identity matrix can be attached to H_r to construct the APCM (denoted as H), which is shown in Fig.9(b).

Suppose the lost blocks are $\{D_1, D_2, P_1\}$, then a matrix transformation is applied to H, which changes the corresponding three columns into an identity matrix. The transformation procedure follows Algorithm 1, and it is illustrated in Fig.10. The transformed matrix (denoted as H_T) can be divided into two sub-matrices, H_{TS} and H_{TL}. H_{TS} is constructed by columns $\{1, 4, 5, 7\}$ and H_{TL} consists of columns $\{2, 3, 6\}$. As mentioned before, H_{TL} is an identity matrix. Finally, the lost data can be recovered by the equation $H_{TS} \times D_S$.

2) An example of LRC codes with APCM: The LRC (Local Reconstruction Codes) is a kind of non-MDS code, which is based on original RS Code. A typical LRC codes can be represented by LRC (k, lr, gr), where k, lr, gr represent the total number of data blocks, the number of local parity

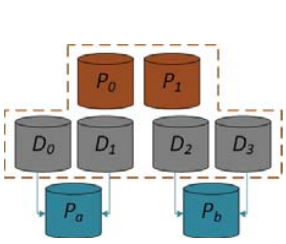


(a) First step of Matrix Transformation is to transform the second column into (1, 0, 0). Element b_{ij} in the matrix above is generated by XOR the second and third rows with the first row. For instance, $b_{11} = a_1 \oplus a_2$ and $b_{21} = a_1^2 \oplus a_2^2$.

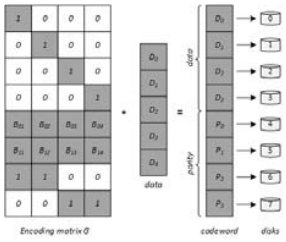


(b) Since the 6th column has already had only one 1 in it, we can directly use the column. Therefore, the third column is transformed into (0, 0, 1) to construct an identity matrix. Element c_{ij} is generated by XOR the first and second rows with the third row. For instance, $c_{11} = b_{11} \oplus (b_{21} * b_{13}/b_{23})$.

Fig. 10. Matrix Transformation Procedure of RS code.



(a) The structure of LRC codes ($k = 4, lr = 2, gr = 2$).



(b) The encoding matrix of LRC codes ($k = 4, lr = 2, gr = 2$).

Fig. 11. An example of LRC codes with ($k = 4, lr = 2, gr = 2$).

blocks and the number of global parity blocks, respectively. In the construction of LRC, k data blocks are divided into lr groups, and each group contains one local parity block, which is calculated by XOR operation. Fig.11 shows the layout and the encoding of an LRC codes.

As shown in Fig.11, there is a sample LRC codes with $lr = 2$, $gr = 2$, and $k = 4$. According to the decoding process of LRC, we have two groups consisting of one local parity and two data disks in each group. The situation of local reconstruction only use $\oplus$ which is XOR operation and cannot be applied with APCM method. The situation of global reconstruction is entirely the same as the reconstruction of RS code. Thus, we suppose that three data disks fail and a joint reconstruction which use both local and global parity disks to recover the lost data.

The APCM for LRC codes can be constructed by using the

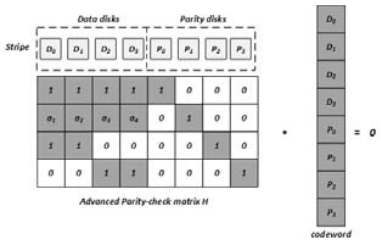
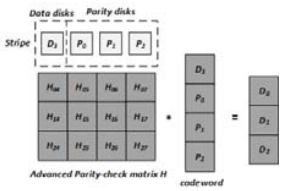


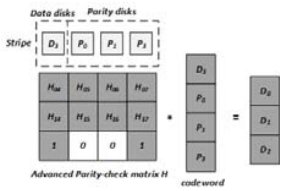
Fig. 12. The Construction of an Advanced Parity-Check Matrix for LRC when ($k = 4, lr = 2, gr = 2$).

$lr+gr$ encoding equations, which is shown in Fig.12. The last $gr + lr$ columns of the APCM also from an identity matrix. According to the construction of LRC [14], it does not have the Maximum Distance Separablecan (MDS) property and can tolerate arbitrary $gr + 1$ disk failures and up to $gr + lr$ disk failures. Therefore, we can guarantee that arbitrary $gr + 1$ columns in the matrix can compose a full rank sub-square-matrix, with arbitrary $gr + 1$ rows.

The process to generate H_T and H_{TS} is similar to the process of RS code, except that we can choose arbitrary $gr+1$ rows from $gr + gl$ rows. Assume that $D_L = \{D_1, D_2, D_3\}$, in the example of Fig.12, two H_T can be generated from the original APCM, which is illustrated in Fig.13.



(a) The decoding matrix H_{TS} for LRC codes when we choose $D_S = \{D_3, P_0, P_1, P_2\}$ to decode.



(b) The decoding matrix H_{TS} for LRC codes when we choose $D_S = \{D_3, P_0, P_1, P_3\}$ to decode.

Fig. 13. Two different H_{TS} when we choose different blocks to reconstruct the lost data.

IV. EVALUATION

In this section, we conduct a series of experiments by using both XOR-based and RS-based codes, to demonstrate the effectiveness of APCM on decoding performance. Specifically, we implement APCM into Alibaba cloud storage system to get the experimental results. A full system evaluation is also implemented in a four-node Hadoop cluster.

A. Evaluation Methodology

1) Erasure codes Selection: We select the following erasure codes in our evaluation, including four RS-based codes and two XOR-based codes.

- RS-based Codes:
  - *Reed-Solomon Code* [32]: A typical erasure code that can tolerate arbitrary number of disk failures.

- *Local Reconstruction Codes* [14]: An erasure code scheme which use more storage space to obtain better decoding performance.
- *Clay Codes* [38]: A type of Minimum Storage Regenerating (MSR) codes, which is a vector code and utilize less recovering data when reconstruct a lost disk.
- *Optimal-access MDS code* [46]: Another type of Minimum Storage Regenerating (MSR) codes, which has a similar structure and property to Clay Codes.

- XOR-based Codes:
  - *RDP Code* [7]: A typical RAID-6 code for $p+1$ disks to tolerate double disk failures.
  - *TIP-Code* [48]: A typical erasure code for $p+2$ disks to tolerate triple disk failures.

2) Evaluation Environment: Our experiments are conducted on two different platforms. One is conducted on a DELL R730 cluster with four servers, and the configuration of each server is described in Table III. We use it to explore the performance of APCM approach under various of erasure coding settings and different block sizes for typical RS-based codes (RS code, LRC codes, Clay codes, etc.). The other platform is based on the Alibaba cloud storage system which is described in Table III, for exploring the performance of APCM approach under various of data block sizes. Currently, Alibaba Cloud uses RS code as their prime erasure coding scheme, therefore we have only evaluated the performance of RS code in this platform.

TABLE III
DETAILS OF THE EVALUATION PLATFORM

Description	Alibaba Cloud V41 Server	DELL R730 Server
CPU	2 × Intel Xeon 2.5GHz	Intel Xeon 3.0GHz
NIC	Mellanox 25Gbps	1Gbps
Memory	128GB	32GB
Disk	24TB SSD	8TB HDD
OS	Linux 3.10	Linux 3.19

3) Metrics and Methods for Experiments: To evaluate the decoding performance in implementation, we implemented our work in Intel ISA-L [1] (Intelligent Storage Acceleration Library) and compared it with several approached.

1) RS-based Codes:
   - APCM approach implemented in Intel ISA-L.
   - General matrix-based decoding approach implemented in Jerasure.
   - General matrix-based decoding approach implemented in Intel ISA-L.
2) XOR-based Codes:
   - APCM approach implemented by C.
   - PCM approach [47] implemented by C.
   - General matrix-based XOR decoding approach implemented by C.

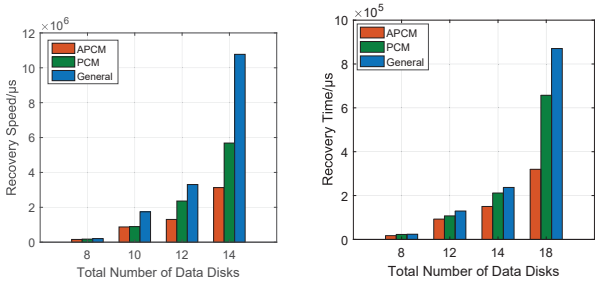
The metrics of these evaluations is ***recovery time***, which is measured on the time used to recover all failed blocks. To evaluate the decoding performance, we randomly generate k data blocks, then encode them into n blocks (which means $r = n - k$ parity blocks are generated). For RS code and LRC codes, we randomly drop r blocks as the decoding scenarios. For XOR-based codes, we randomly drop r blocks as the disk failure scenario, which is 2 for RDP code and 3 for TIP code. For Clay codes, we randomly drop 1 block as the disk failure scenario. Finally, we recover the failed blocks from the surviving blocks and check the correctness. We also vary the block size from 1KB to 32KB with fixed erasure code setting to evaluate the decoding performance.

B. Experimental Results of XOR-based Codes

We choose two XOR-based codes, RDP code and TIP-Code, to compare our methods with PCM approach (since it can be used in XOR-based codes) and general matrix-based method. These two codes can respectively tolerate double and triple disk failures.

In the experiments for RDP code, we vary the number of data blocks (parameter k from 8 to 14) while holding the number of parity blocks to 2. The results are shown in Fig.14(a). In this diagram, APCM can improve the decoding performance by up to 70.94% comparing with the general matrix-based method. Compared with the PCM approach, APCM can achieve up to 44.93% increment. With the increasing of k, APCM can achieve a better performance.

For TIP-Code, we vary the number of data blocks (parameter k from 8 to 18) and holding the number of parity blocks to 3. The results are presented in Fig.14(b). It shows that, compared with the general matrix-based approach and PCM methods, the maximum improvements are 63.26% and 51.35%, respectively.



(a) Recovery time comparison among APCM, PCM and the general matrix-based decoding approach for RDP code when the code configuration is set to $k = (8, 10, 12, 14)$ and $r = 2$.

(b) Recovery time comparison among APCM, PCM and the general matrix-based decoding approach for TIP code when the code configuration is set to $k = (8, 12, 14, 18)$ and $r = 3$.

Fig. 14. Comparison of XOR-based code between APCM, PCM and the general matrix-based method.

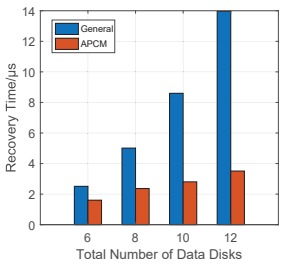
C. Experimental Results of RS Code

RS code is a typical erasure code that can tolerate arbitrary number of disk failures. To evaluate the performance of APCM, we implement the APCM both on an ordinary DELL R730 server and Pangu file system, which is storage system of Alibaba Cloud. On both platforms, r blocks are randomly

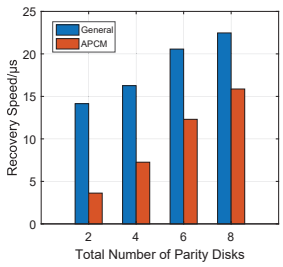
chosen as the lost data and the recovery time is the average time of 1000 repeated experiments.

1) Experimental Results of RS code in Alibaba Cloud: Pangu file system supports erasure coding in Alibaba Cloud. The system uses the third-party libraries, such as Jerasure and Intel ISA-L, to implement the erasure code with matrix-based method. We select a set of data block sizes (2KB to 32KB) in our evaluation, and implement APCM in the Pangu file system combined with Intel ISA-L library. The original Intel ISA-L and Jerasure are included in our comparison. Since the erasure code scheme in Pangu is RS code with $(k = 4, r = 3)$ and $(k = 8, r = 3)$ configurations, we evaluate the performance of APCM approach under these two scenarios.

Fig.16 shows the comparison between APCM and Intel ISA-L library in terms of recovery time with the varying of data block size. In Fig.16(a), we can find out that under the RS code configuration $RS(k = 8, r = 3)$, recovery time is optimized by 84.89% compared to the Jerasure and 32.31% compared to the ISA-L, respectively. Fig.16(b) shows that under the RS code configuration $RS(k = 4, r = 3)$, recovery time increment is up to 30.40% compared to the Jerasure, and up to 6.10% compared to the ISA-L.



(a) Recovery time comparison between APCM and the general matrix-based approach implemented in ISA-L when the code configuration is set to $k = (6, 8, 10, 12)$ and $r = 2$.

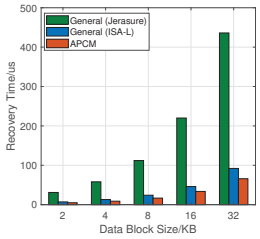


(b) Recovery time comparison between APCM and the general matrix-based approach implemented in ISA-L when the code configuration is set to $k = 12$ and $r = (2, 4, 6, 8)$.

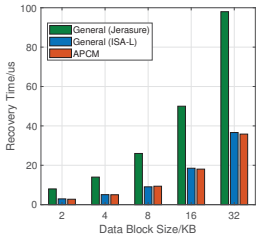
Fig. 15. Comparison of RS code between Advanced Parity-Check Matrix approach and the general matrix-based approach implemented in ISA-L.

2) Experimental Results of RS code on DELL Server: In this experiment on one hand, we vary the number of data blocks (parameter k from 6 to 12) while holding the number of parity blocks to 2, to find the impact of parameter k on APCM. On the other hand, to explore the impact of parameter r, we vary the parity blocks (parameter r from 2 to 8) staying the same value of the number of data blocks $(k = 12)$.

Fig.15 shows the comparison between APCM and Intel ISA-L library in terms of recovery time with different configurations of erasure codes. As illustrated in Fig.15(a), the APCM approach can improve the recovery time in all cases when we set r to constant 2 and vary k. The best optimization ratio is 74.86% when k is 12, and when k is 6, the minimum optimization ratio is 36.12%. Fig.15(b) shows that APCM approach can also improve the recovery time when k is set to a constant 12. The best optimization ratio can achieve to 74.40% when r is 2, and when r is 8, the minimum optimization ratio is 29.36%.



(a) Recovery time comparison among APCM, the general matrix-based approach implemented in ISA-L and the general matrix-based approach implemented in Jerasure when the RS code configuration is set to $k = 11$ and $r = 3$ and block data size is from 2KB to 32KB.



(b) Recovery time comparison among APCM, the general matrix-based approach implemented in ISA-L and the general matrix-based approach implemented in Jerasure when the RS code configuration is set to $k = 7$ and $r = 3$ and block data size is 2KB to 32KB.

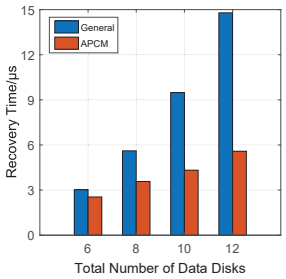
Fig. 16. Comparison of RS code between APCM, general matrix-based approach implemented in ISA-L and the general matrix-based approach implemented in Jerasure in the cloud storage system of Alibaba Cloud.

D. Experimental Results of LRC Codes

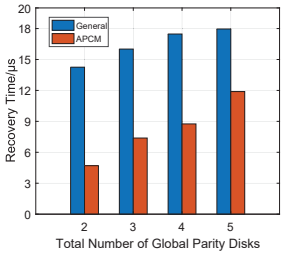
Local Reconstruction Codes or Locally Repairable Codes (LRC) is a category of RS-based code, which utilize additional storage space to obtain good decoding performance. Microsoft and Facebook have already designed and implemented two different LRC codes in their cloud storage systems. We implement the Microsoft LRC codes into Intel ISA-L library and apply APCM approach on it.

In our experiment, we set the number of local parity lr=2 and gr is a constant as well with $(gr = 2)$. We vary the number of data blocks (parameter k from 6 to 12) to find the impact of parameter k on the APCM method. In order to find the impact of parameter gr on APCM method, we vary the number of global parity blocks (parameter gr from 2 to 8) and set the number of data blocks to the constant 12. We also vary the data block size from 1KB to 32KB, to evaluate the decode performance of APCM approach applied on LRC codes. The experiment environment is described in Table III.

Fig.17 shows the comparison between APCM and Intel ISA-L library in terms of recovery time with the varying of code configuration and Fig.18(a) illustrates the impact of data block size on the decoding performance. As illustrated in Fig.17(a), the APCM approach reduce the recovery time in all cases when we set gr to constant 2 and vary k. The best optimization ratio is 62.25% when k is 12, and when k is 6, the minimum optimization ratio is 15.99%. Fig.17(b) shows that APCM approach saves the recovery time when we set k to a constant 12 via changing the number of global parity blocks gr. The best optimization ratio is 67.01% when gr is 2, and when gr is 8, the minimum optimization ratio is 33.79%. Fig.18(a) shows that the recovery time are decreased from 15.39% to 64.88% when the data block size is varying from 1KB to 32KB.

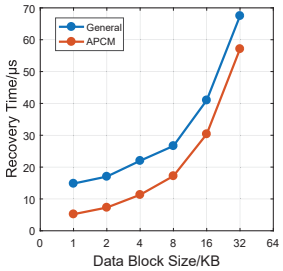


(a) Recovery time comparison between APCM and the general matrix-based approach implemented in ISA-L when the RS code configuration is set to $k = (6, 8, 10, 12)$ and $gr = 2$.

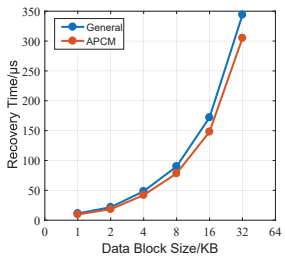


(b) Recovery time comparison between APCM and the general matrix-based approach implemented in ISA-L when the RS code configuration is set to $k = 12$ and $gr = (2, 3, 4, 5)$.

Fig. 17. Comparison of LRC codes between APCM and the general matrix-based approach implemented in ISA-L.



(a) Recovery time comparison of LRC codes between APCM and the general matrix-based approach implemented in ISA-L. The code configuration is $k = 12$ and $gr = 2$.



(b) Recovery time comparison of Clay codes between APCM and the general matrix-based approach implemented in ISA-L. The code configuration is $k = 2$ and $r = 2$.

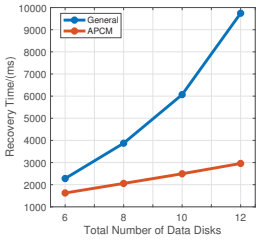
Fig. 18. Recovery time comparison of LRC codes and Clay codes between APCM and the general matrix-based approach implemented in ISA-L, with a changing of block size from 1KB to 32KB.

E. Experimental Results of Clay Codes

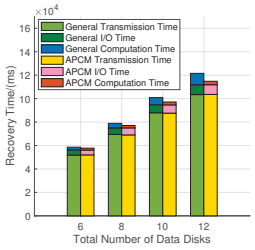
Clay Codes is a type of Minimum Storage Regenerating (MSR) codes, which is a kind of vector code and utilize less recovering data when reconstruct a lost disk.In our experiment, we directly set the code configuration of Clay Codes to $(k = 2, r = 2)$, and vary the block data size from 1KB to 32KB, to evaluate the decoding performance of APCM approach applied on Clay Codes. The experiment environment is described in Table III.Fig.18(b) shows that the recovery time can be improved from 11.34% to 15.50% when the block data size changes.

F. Experimental Results of Full-System Evaluation

In order to demonstrate the effectiveness of the APCM decoding method on full system, we set up a Hadoop cluster consisting of one name node and three data nodes (four nodes in total). Four DELL R730 servers are deployed as these nodes and connected with a gigabit Ethernet. Two RS-based codes, including the RS code and MSR code [46], are selected to compare the effectiveness of APCM method and general matrix-based decoding approach. The configuration of the DELL R730 server is described in Table III.



(a) Decoding time comparison between APCM and the general matrix-based approach.



(b) Full recovery time comparison between APCM and the general matrix-based approach.

Fig. 19. Comparison between APCM and the general matrix-based approach in terms of decoding time and full recovery time (when the RS code configuration is set to $k = (6, 8, 10, 12)$ and $r = 3$). In the figures, light green and yellow bars represent the transmission time of APCM and general matrix-based approach while dark green and pink bars represent the I/O time, blue and red bars represent the decoding time.

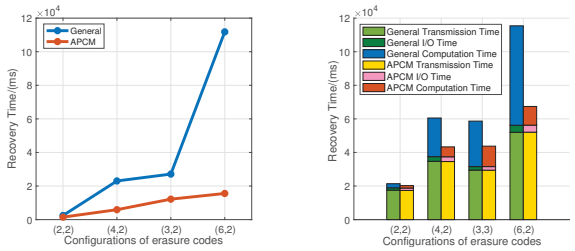
1) Full-System Experimental Results of RS code: In this experiment, we vary the number of data nodes (parameter k from 6 to 12) while holding the number of parity nodes (parameter $r = 3$), to find out the impact of parameter k on the APCM method. We set the block size to 1KB and storage capacity of each node is 1GB. In our experiment, we assume that r nodes are corrupted, and a total capacity of rGB data should be recovered.

Fig.19 shows the comparison between APCM and general matrix-based approach in terms of decoding time and full recovery time. In Fig.19(a), we can find out that under RS($k = 12, r = 3$), decoding time is optimized by 69.63% compared to the general matrix-based approach. Fig.19(b) illustrates that the network transmission time is about $10.6\times$ of the computation time and $12.3\times$ of the disk I/O time. APCM method can optimize up to 5.55% compared to the general matrix-based approach in the entire recovery process. We find out that the transmission time is almost the same in both methods. The detailed evaluation results are shown in Table VII.

2) Full-System Experimental Results of MSR code: In this experiment, we vary the number of data nodes (parameter k from 2 to 6) while setting the number of parity to 2 when $k = \{2, 4, 6\}$ and to 3 when $k = 3$ because of the special constraint of MSR-code. We set the block size to 1KB and the size of a node to 1GB.

Fig.20 shows the comparison between APCM and general matrix-based approach in terms of decoding time and full recovery time. Fig.20(a) illustrates under MSR($k = 6, r = 2$), the decoding time is optimized by 81.19% compared to the general matrix-based approach. In Fig.20(b), we can find out that the decoding time grows fast when the matrix size increases and the ratio of transmission time to decoding time decrease with the size grows, especially in MSR $(k = 6, r = 2)$ which makes the decoding time larger than the transmission time. The ratio decrease from $12.0\times$ to $4.7\times$ for APCM

and from $7.1\times$ to $0.9\times$ for general method. This is because the encoding matrix of MSR is $(l \times n) \times (l \times k)$, in which $l = r^{\frac{n}{r}}$. The large matrix of MSR results in a much higher computational overhead comparing with RS or LRC codes. APCM method can optimize up to 41.61% compared to the general matrix-based approach in the entire recovery process. The detailed evaluation results are shown in Table.VII.



(a) Decoding time comparison between APCM and the general matrix-based approach.

(b) Full recovery time comparison between APCM and the general matrix-based approach.

Fig. 20. Comparison between APCM and the general matrix-based approach in terms of decoding time and full recovery time (when the Optimal-access MDS code configuration (k, r) is set to $\{(2,2),(4,2),(3,3),(6,2)\}$).

G. Analysis

The detailed improvements of APCM approach comparing with the general matrix based decoding method are listed in Tables V and VI. From these tables, we can find APCM gets high decoding performance, and there are several reasons to achieve these gains. First, Parity-Check Matrix is an efficient method to reduce the decoding overhead. Second, APCM reduces the matrix inversion cost, which simplifies the matrix computation further. Third, APCM is a flexible approach, which can be applied to all erasure codes and combined with different libraries such as Jerasure and ISA-L. The full time evaluation shows that APCM method does not have side-effect on network transmission, since the transmission time is almost the same in both full system experiments. It also illustrates that APCM performs much better when the size of encoding matrix is large.

TABLE IV
IMPROVEMENT OF APCM IN TERMS OF DECODING COMPLEXITY XOR-BASED CODES

Coding	k				
	8	10	12	14	18
RDP Code	23.35%	49.83%	60.53%	70.94%	/
TIP Code	27.04%	/	28.42%	36.60%	63.26%

V. CONCLUSION

In this paper, we propose a novel Advance Parity-Check Matrix based (APCM) approach for RS-based erasure codes in cloud storage environment. Based on APCM, we design the decoding algorithms for high decoding efficiency. To demonstrate the effectiveness of APCM, we conduct several experiments by using both RS-based and XOR-based codes.

TABLE V
IMPROVEMENT OF APCM IN TERMS OF DECODING COMPLEXITY RS-BASED CODES

Coding	k			
	6	8	10	12
RS Code	36.12%	52.84%	67.39%	74.86%
LRC Codes	15.99%	36.40%	54.40%	62.25%
Coding	r			
	2	4	6	8
RS Code	74.40%	55.37%	40.18%	29.37%
LRC Codes	67.02%	53.89%	49.90%	33.79%

TABLE VI
IMPROVEMENT OF APCM IN TERMS OF DECODING COMPLEXITY WITH VARIOUS DATA BLOCK SIZE

Coding	Data Block Size					
	1KB	2KB	4KB	8KB	16KB	32KB
Clay Codes	11.3%	14.5%	15.5%	13.3%	12.9%	13.9%
LRC Codes	64.9%	57.1%	35.5%	35.5%	25.9%	15.4%
RS Code (k=8,r=3)	/	30.4%	32.3%	30.6%	26.9%	28.4%
RS Code (k=4,r=3)	/	6.1%	3.4%	−3.3%	2.4%	2.2%

TABLE VII
IMPROVEMENT OF APCM ON FULL SYSTEM EVALUATION

RS	k			
	6	8	10	12
Decoding Time	28.65%	46.84%	58.9%	69.63%
Full Recovery Time	1.38%	2.32%	3.68%	5.55%
MSR	(k, r)			
	(2,2)	(4,2)	(3,3)	(6,2)
Decodint Time	40.13%	74.37%	55.06%	81.19%
Full Recovery Time	1.81%	28.43%	25.44%	41.61%

Compared with existing matrix based implementation methods (including Intel ISA-L and Jerasure libraries), APCM shows the following advantages: 1) speeds up the decoding process by up to 32.31% in the Alibaba cloud storage system; 2) is effective for both XOR-based and RS-based codes.

VI. ACKNOWLEDGEMENT

This work is partially sponsored by the National Key R&D Program of China (No.2018YFB0105203), the National 973 Program of China (No.2015CB352403), the National Natural Science Foundation of China (NSFC) (No.61628208), the Natural Science Foundation of Shanghai (No.18ZR1418500), the Alibaba Group through Alibaba Innovative Research (AIR) program, and the U.S. National Science Foundation grants CCF-1717660, CCF-1813081 and CNS-1702474.

REFERENCES

[1] "Intel storage acceleration library(open source version)," https://goo.gl/zkVl4N.
[2] 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.
[3] M. Blaum and S. R. Hetzler, "Integrated interleaved codes as locally recoverable codes: properties and performance," *International Journal of Information and Coding Theory*, vol. 3, no. 4, pp. 324–344, 2016.
[4] 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.

[5] V. Bohossian, C. C. Fan, P. S. LeMahieu, M. D. Riedel, L. Xu, and J. Bruck, "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.

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

[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 *Proc. of the USENIX FAST'04*, San Francisco, CA, March 2004.

[8] A. G. Dimakis and P. B. e. a. Godfrey, "Network coding for distributed storage systems," in *IEEE INFOCOM 2007*, 2007, pp. 2000–2008.

[9] Y. Fu and J. Shu, "D-code: An efficient raid-6 code to optimize i/o loads and read performance," in *Parallel and Distributed Processing Symposium*, 2015, pp. 603–612.

[10] P. Gopalan, C. Huang, B. Jenkins, and S. Yekhanin, "Explicit maximally recoverable codes with locality." *Electronic Colloquium on Computational Complexity*, vol. 20, p. 104, 2013.

[11] H. Anvin, "The mathematics of RAID-6," http://kernel.org/pub/linux/kernel/people/hpa/raid6.pdf, May 2009.

[12] Y. Hu, X. Zhang, P. P. Lee, and P. Zhou, "Generalized optimal storage scaling via network coding," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 956–960.

[13] 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.

[14] 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 *Usenix annual technical conference*. Boston, MA, 2012, pp. 15–26.

[15] Y. Jiang, C. Wu, J. Li, and M. Guo, "Eh-code: An extended mds code to improve single write performance of disk arrays for correcting triple disk failures," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2015, pp. 34–49.

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

[17] H. Li, Y. Zhang, Z. Zhang, S. Liu, D. Li, X. Liu, and Y. Peng, "Parix: Speculative partial writes in erasure-coded systems." pp. 581–587, 2017.

[18] R. Li, X. Li, P. P. Lee, and Q. Huang, "Repair pipelining for erasure-coded storage," in *Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC17)*, 2017, pp. 567–579.

[19] S. Li, S. Wan, D. Chen, Q. Cao, C. Xie, X. He, Y. Guo, and P. Huang, "Exploiting decoding computational locality to improve the I/O performance of an XOR-coded storage cluster under concurrent failures," in *Proc. of the SRDS'14*, Nara, Japan, October 2014.

[20] S. Mitra, R. Panta, M.-R. Ra, and S. Bagchi, "Partial-parallel-repair (ppr): a distributed technique for repairing erasure coded storage," in *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 2016, p. 30.

[21] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang *et al.*, "f4: Facebooks warm blob storage system," in *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*. USENIX Association, 2014, pp. 383–398.

[22] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *IEEE International Symposium on Information Theory Proceedings*, 2015, pp. 2771–2775.

[23] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population." in *Proc. of the USENIX FAST'07*, San Jose, CA, February 2007.

[24] 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.

[25] J. S. Plank, "A tutorial on reed–solomon coding for fault-tolerance in raid-like systems," *Software: Practice and Experience*, vol. 27, no. 9, pp. 995–1012, 1997.

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

[27] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast galois field arithmetic using intel simd instructions." in *FAST*, 2013, pp. 299–306.

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

[29] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.

[30] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for i/o, storage, and network-bandwidth." in *FAST*, 2015, pp. 81–94.

[31] K. Rashmi and N. B. e. a. Shah, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. of the SIGCOMM'14*, Snowbird, UT, June 2014.

[32] 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.

[33] Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence, "Fab: Building distributed enterprise disk arrays from commodity components," in *ACM SIGARCH Computer Architecture News*, vol. 32, no. 5. ACM, 2004, pp. 48–58.

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

[35] 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 *Proc. of the USENIX FAST'07*, San Jose, CA, February 2007.

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

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

[38] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy *et al.*, "Clay codes: Moulding mds codes to yield an msr code," in *Proc. 16th USENIX Conference on File and Storage Technologies, Oakland, CA, USA*, vol. 2018, 2018, pp. 139–154.

[39] Y. Wang, G. Li, and X. Zhong, "Triple-Star: A coding scheme with optimal encoding complexity for tolerating triple disk failures in RAID."

[40] 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 *Proc. of the IPDPS'11*, Anchorage, Alaska, May 2011.

[41] 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.

[42] M. Xia, M. Saxena, M. Blaum, and D. Pease, "A tale of two erasure codes in hdfs," pp. 213–226, 2015.

[43] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in rdp code storage systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 119–130, 2010.

[44] L. Xiang, Y. Xu, J. C. S. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using raid-6 codes:algorithms and performance evaluation," *Acm Transactions on Storage*, vol. 7, no. 3, pp. 1–34, 2011.

[45] 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.

[46] M. Ye and A. Barg, "Explicit constructions of optimal-access mds codes with nearly optimal sub-packetization," *IEEE Transactions on Information Theory*, vol. PP, no. 99, pp. 1–1, 2017.

[47] Y. Zhang, C. Wu, J. Li, and M. Guo, "Pcm: A parity-check matrix based approach to improve decoding performance of xor-based erasure codes," in *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015, pp. 182–191.

[48] ——, "Tip-code: A three independent parity code to tolerate triple disk failures with optimal update complextiy," in *Ieee/ifip International Conference on Dependable Systems and Networks*, 2015, pp. 136–147.

[49] Y. Zhu, P. P. Lee, Y. Hu, L. Xiang, and Y. Xu, "On the speedup of single-disk failure recovery in xor-coded storage systems: Theory and practice," in *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*. IEEE, 2012, pp. 1–12.