ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20211213044600
Original scientific paper

A Novel Completely Local Repairable Code Algorithm Based on Erasure Code

Ying FANG, Hai TAN*, Shuaifang WANG*, Xin ZHANG, Gejian LIAO, Jun ZHANG

Abstract: Hadoop Distributed File System (HDFS) is widely used in massive data storage. Because of the disadvantage of the multi-copy strategy, the hardware expansion of HDFS cannot keep up with the continuous volume of big data. Now, the traditional data replication strategy has been gradually replaced by Erasure Code due to its smaller redundancy rate and storage overhead. However, compared with replicas, Erasure Code needs to read a certain amount of data blocks during the process of data recovery, resulting in a large amount of overhead for I/O and network. Based on the Reed-Solomon (RS) algorithm, we propose a novel Completely Local Repairable Code (CLRC) algorithm. By grouping RS coded blocks and generating local check blocks, CLRC algorithm can optimize the locality of the RS algorithm, which can reduce the cost of data recovery. Evaluations show that the CLRC algorithm can reduce the bandwidth and I/O consumption during the process of data recovery when a single block is damaged. What's more, the cost of decoding time is only 59% of the RS algorithm.

Keywords: data recovery; erasure code; HDFS; RS algorithm, storage overhead

1 INTRODUCTION

The most fundamental reason why HDFS was initially widely recognized in the industry is its good storage performance for massive data storage [1, 2]. Not only does it have easy-to-expand storage capacity, it also ensures reliable storage files through a multiple copy strategy. For the file uploaded by clients, HDFS divides it into a plurality of blocks. Each data block generates three replications on different data nodes. When one or more blocks in the file system are damaged, HDFS can still restore data by reading other replications. The advantage of the multi-copy strategy is simple to implement and the extremely fast speed of file recovery, whereas the disadvantages are the cost of a lot of extra storage capacity and the low disk space utilization. To more efficiently increasing data storage pressure, the speed of the underlying hardware expansion cannot keep up with the growth rate of data. Therefore, the defects of the HDFS multi-copy strategy become more and more prominent [3]. In the field of the data reliability research of the distributed file system, two strategies are commonly applied: one is the multi-copy strategy, the other is EC (Erasure Code) method including Array Codes and RS (Reed-Solomon) code most popularly used [4, 5].

EC was originally applied to the traditional communication field as a loss signal fault-tolerant method in the network signal transmission. The basic idea is to segment the transmission signal by a specific algorithm and generate a corresponding check segment for the signal identification in the communication process [6]. For EC, when the part of the signal is lost, the original signal can be recovered by the check segment. Similarly, as a data redundancy technology applied to distributed file systems, EC can be seen as an extension of RAID (Redundant Arrays of Independent Drives) 6. EC divides the raw data into k data blocks, then produces the m-th code blocks by using the encoding matrix operation, so the total number of data blocks that we need to store in the different data nodes, at last, is $n = m + k$. When one or more data blocks of a file are damaged, they can be recovered by decoding from other data blocks. The minimum number of available blocks required in the recovering process is k.

Compared with the replication strategy, the EC method significantly improves disk space utilization and ensures the reliability of the stored data. However, when recovering the damaged data, it is necessary to pull multiple data blocks from different data nodes. This greatly increases the overhead of I/O and the network of the file system [7].

In this paper, after the classic EC algorithm is studied, a new CLRC algorithm is proposed as the HDFS data redundancy strategy, which can improve the performance of the RS code and meet the HDFS storage requirements for improving its space utilization.

The organization of this paper is as follows: The second part is a literature review and the third section describes the proposed CLRC algorithm. The fourth part explains the experimental results and discussion. Finally, the conclusion is presented in the last section.

2 LITERATURE REVIEW

There are two main types of repair, functional repair and accurate repair. There is a big difference between the two kinds of repairs. Functional repair means that when a block is lost, different blocks are generated to replace the lost blocks, and the fault tolerance rate of n-k is maintained as well. The main problem with functional recovery is that when a block of data is lost, a check block is generated to replace it. In this case, although the n-k fault tolerance rate is maintained, when only one block of data needs to be repaired, k blocks need to be retrieved from the system. The network and I/O consumption are k times of the replica strategy. This functional repair is more suitable for systems that are not read frequently, such as archiving systems. Accurate repair means that the recovered block is identical to the lost block. For reading and writing HDFS frequently, using functional repair costs too much. So, we only discuss the accurate repair in this article.

From the previous paper [8], Dimakis et al. point out that k blocks need to be read and transmited during the repair process of standard RS (k, $n - k$) code, but it is possibly less than k. The author proves the existence of boundaries and proposes the earliest Generating Codes, which can only be repaired functionally. But at present, there is no generating code that can reach the boundary of information theory. Rashmi et al. then point out that there exists an accurate repair that conforms to the boundaries of information theory [9, 10]. Coding design can be divided

into two types according to storage efficiency rate k/n: one is $k/n <= 1/2$; the other is $k/n > 1/2$. For codes with storage efficiency less than or equal to 1/2, a generation code that can be accurately repaired has been found [11].

Cadambe et al. prove that the generated code with accurate repair exists by introducing interference alignment, an asymptotic technology in wireless information theory [12]. But this construction requires exponential field size and can only be applied in an asymptotic mechanism, so it is hard to implement. Although the research of precise generating code is very hot, there is still not much achievement at present.

Another optimization method is to reduce the cost of repair by reducing the I/O and network bandwidth consumption in repair operations (e.g. [13-15]). The measure used in this method is locality, which represents the number of other data blocks needed to reconstruct a lost block. Miyamae et al. propose a novel RS code structure [16]. By dividing the strips into overlapping arrays, each group generates a check block. This kind of RS code constructed can not only improve the efficiency of repairing a single block but also costs less than the standard RS code scheme when multiple blocks are lost at the same time. In recent years, LRC (Locally Repairable Codes) has also been introduced by Window Azure In, but its scheme is stricter in parameter selection, and the time complexity of searching equation coefficients increases exponentially with strip length. Tamo et al. systematically analyzed the correlation between the locality of LRC codes and other parameters and proposed a theoretically optimal construction of LRC codes, which grouped data blocks based on original MDS codes, and then generated additional check blocks with each group of MDS codes [17]. The finite field size of MDS codes used is also different.

3 RESEARCH METHOD

1) ERASURE CODE

Currently, there are Array Codes and RS encoding in the EC algorithms which are widely used in the world [18]. The first one is the earliest EC algorithm, which can be divided into horizontal arrays and vertical arrays depending on their code blocks' storage location. Due to the XOR operation used in the coding process, Array Codes have lower computational complexity and simpler implementation than the RS method. The other one has two coding methods including the Vandermonde coding matrix and Cauchy coding matrix. Particularly, Cauchy has lower computational complexity and higher coding efficiency compared with Vandermonde.

A. Definition and description of nouns

There are a lot of basic concepts in the field of Erasure Code research for the distributed file system. To facilitate the work in the future, a brief description of some terms and their implications are as follows.

Data Stripe

As shown in Fig. 1, the stored data is logically divided into multiple stripes in RAID. Each stripe crosses the entire disk system and contains all the source data with its corresponding check data. Each stripe is composed of multiple strips that are distributed among all the disks. The volume of the stripe is determined by the amount of the original data and the number of disks.

In the distributed file system, the original data is also divided into multiple stripes. However, the distribution for stripes is just determined by the name node rather than crossing all the data nodes. Generally, we assume that the stripe has the same size. We pad the last data strip by zero when the data volume is not enough to fill up.

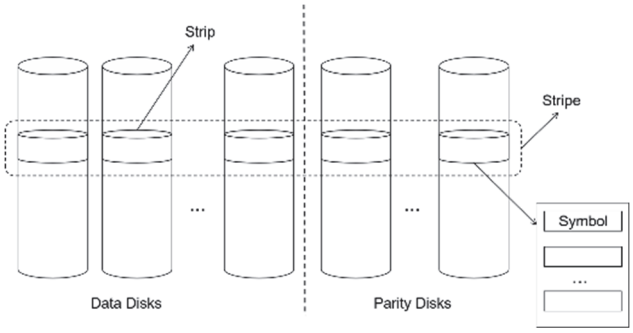

Figure 1 RAID system data stripe

Galois field

The Galois field is a domain that contains only a limited number of elements. It has wide applications in the field of coding. RS coding uses a Galois field $GF(2^W)$. There is a total number of 2^W elements in $GF(2^W)$ with the range:

$$v \in \left[0, 2^W\right) \cap v \in N \tag{1}$$

Each Galois field $GF(2^W)$ can be expressed as a polynomial form and its highest order is $w - 1$:

$$a(x) = \alpha_{W-1}X^{W-1} + \alpha_{W-2}X^{W-2} + ... + \alpha_1 X \tag{2}$$

where the set $\{\alpha_{W-1}, \alpha_{W-2}, ..., \alpha_1\}$ is included in $GF(2^W)$ and the value of each element is 0 or 1.

The addition operation can be regarded as bitwise XOR in the bit operation in Galois field $GF(2^W)$ and $GF(2^W)$. However, in the Galois field $GF(2^W)$, the multiplication operation can be regarded as bitwise addition in the bit operation. However, the multiplication and division operations are much more complicated in Galois field $GF(2^W)$. For example, in $\left\{a, b \in GF\left(2^W\right)\right\}$, a and b are two elements in the Galois field $GF(2^W)$, then the process of calculation is:

(1) According to Eq. (2), the elements a and b can be expressed as polynomial form $a(x)$ and $b(x)$,

(2) Remainder for the original polynomial $a(x) \cdot b(x)$ as $c(x)$,

(3) According to Eq. (2), converting the value of the $c(x)$ into the number.

The process of a division operation is similar to the multiplication operation in $GF(2^W)$. To reduce the computational complexity, the results of all the elements in the Galois field can be stored in the data table. The result can be directly obtained by searching the data table when needed, but this method is only applied to the low-order Galois field, and the memory consumption rises sharply

with the increase of w. By the way, the complexity of the operation can also be reduced by converting the multiplication and division operations into logarithmic addition and subtraction.

MDS code (Maximum Distance Separable Codes)

The performance of the EC algorithm can be judged by minimum column spacing $d_{\min}$ and locality r. The definition of $d_{\min}$ is: when $d \geq d_{\min}$ blocks are damaged, and the file cannot be repaired. Obviously, for the $n = m + k$ blocks generated by the EC algorithm, the range of values of the $d_{\min}$ is [1, m + 1]. When the $d_{\min}$ is the maximum value, the EC is called an MDS code. The locality r is the minimum number of blocks required for repairing the data when one single block is corrupted. In addition, the smaller the locality r, the lower the cost of the EC algorithm and the smaller the amount of computation for data recovery.

B. Array Codes

The typical Array Codes have EVEN-ODD code and X code [19-20] and their encoding methods are similar. However, the location of their code block is different. The first one is a horizontal array, and the other one is a vertical array. EVEN-ODD can be represented as an array with the size of $\{(k-1) * (k+2) \mid k$ is a prime number greater than the number 2$\}$. The data block is stored in the (1 ~ k) column, and the code block is stored in two separate parity disks.

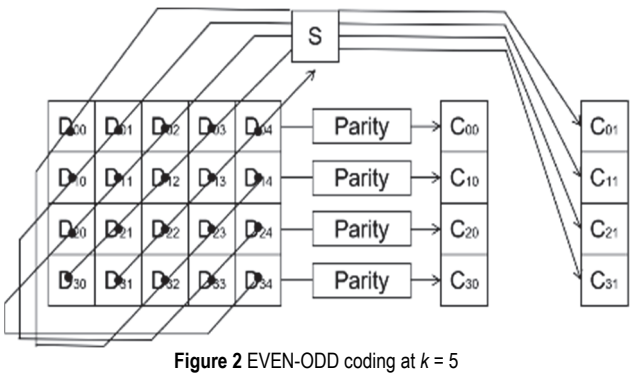

Figure 2 EVEN-ODD coding at k = 5

Fig. 2 shows the EVEN-ODD code with k = 5. The EVEN-ODD code has two columns of code blocks whose encoding can be calculated by the following equations.

$$C_{i,0} = \sum_{t=0}^{k-1} D_{i,t} \left(i = 0,1,...,k-2\right) \tag{3}$$

$$C_{i,1} = S + \sum_{t=0}^{k-2} D_{t,(i-t)\bmod k} \left(i = 0,1,...,k-2\right) \tag{4}$$

The S can be calculated by the following equation.

$$S = \sum_{t=0}^{k-1} D_{t,k-1-t} \tag{5}$$

For example, in Fig. 2, the {C00, C10, C20, C30} can be calculated by the Eq. (3), the {C01, C11, C21, C31} can be calculated by the Eq. (4) and Eq. (5). When $d_{\min}$ = 3, EVEN-ODD code can recover the data only when the loss of the columns is less than or equal to 2.

The X code does not have a separate check disk that is different from the EVEN-ODD code. The X code represents the original data as an array of $\{(k-2) * k \mid k$ is a prime number greater than 2$\}$ with the code block placed at the last two lines of the array. In the X code, the data block can be expressed as D_{ij}, $\{(i,j) \mid i \in [0,k), j \in [0,k)\}$. We use $C_{k-2,\,i}$ to represent the code block of the first line, and $C_{k-2,\,i}$ for the second line. Then the X code can be calculated by the following equations.

$$C_{k-2,i} = \sum_{t=0}^{k-2} D_{t,(i+t-2)\bmod k} \left(i = 0,1,...,k-1\right) \tag{6}$$

$$C_{k-1,i} = \sum_{t=0}^{k-3} D_{t,(k-t-2)\bmod k} \left(i = 0,1,...,k-1\right) \tag{7}$$

Both the X code and the EVEN-ODD code belong to MDS Code. Tab. 1 shows the encoding process of X code when k = 5.

Table 1 X code at k = 5

$D_{0.0}$	$D_{0.1}$	$D_{0.2}$	$D_{0.3}$	$D_{0.4}$
$D_{1.0}$	$D_{1.1}$	$D_{1.2}$	$D_{1.3}$	$D_{1.4}$
$D_{2.0}$	$D_{2.1}$	$D_{2.2}$	$D_{2.3}$	$D_{2.4}$
$D_{0.2} + D_{1.3} + + D_{2.4}$	$D_{0.3} + D_{1.4} + D_{2.0}$	$D_{0.4} + D_{1.0} + + D_{2.1}$	$D_{0.0} + D_{1.1} + + D_{2.2}$	$D_{0.2} + D_{1.3} + D_{2.4}$
$D_{0.3} + D_{1.2} + + D_{2.1}$	$D_{0.4} + D_{1.3} + + D_{2.2}$	$D_{0.0} + D_{1.4} + + D_{2.3}$	$D_{0.1} + D_{1.0} + + D_{2.4}$	$D_{0.2} + D_{1.1} + + D_{2.0}$

C. RS code

For EVEN-ODD code and X code, k must be a prime number, k > 2 and m = 2. What's more, the minimum column distance $d_{\min}$ is 3. This is why it can only recover the data when the loss of the columns is less than or equal to 2. So, the Array Codes are not suitable for the data stored in the distributed system files. However, the k and m can be given arbitrarily in the RS code.

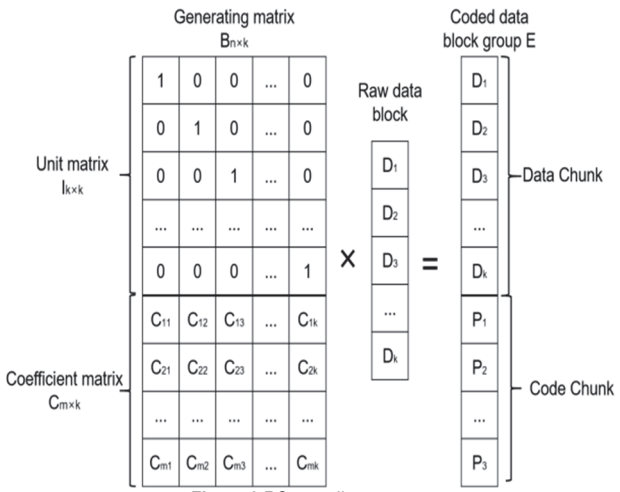

Figure 3 RS encoding process

The RS code encoding process as shown in Fig. 3, B is the generator matrix with $((k+m) * k)$-dimensional whose first k rows are called the unit matrix I, and the last m rows are called the coefficient matrix C.

Then, the column vector E with n-dimensional is obtained by the multiplication of B and data column vector D with K-dimensional. The matrix E includes k data blocks and m code blocks.

RS code is a type of MDS code. For RS (k, m), the minimum column spacing $d_{\min} = m + 1$, and the locality $r = k$. Therefore, unlike the Array codes, RS codes can still recover data when no more than k blocks are damaged, even if k is much greater than 2. When any single block fails, the data with the size of k blocks must be pulled for data recovery.

The RS decoding process is shown in Fig. 4. The matrix E' is composed of K blocks selected from the matrix E in Fig. 3. The matrix B' has the size of $k \times k$ and is composed of corresponding k rows from the matrix B. Then, the original data matrix D can be got from the following equation.

$$B'D = E' \Leftrightarrow B'^{-1}B'D = B'^{-1}E' \Leftrightarrow D = B'^{-1}E' \tag{8}$$

In Eq. (8), the matrix B' must satisfy the reversible condition.

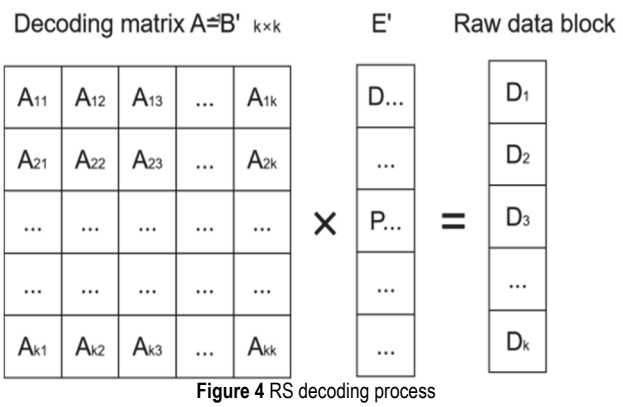

Figure 4 RS decoding process

In Fig. 3, matrix C is the RS code generation coefficient matrix which can be obtained from the Vandermonde matrix and Cauchy matrix. In which, the Vandermonde matrix is defined as:

$$G = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha_0 & \alpha_1 & \dots & \alpha_{k-1} \\ \alpha_0^2 & \alpha_1^2 & \dots & \alpha_{k-1}^2 \\ \dots & \dots & \dots & \dots \\ \alpha_0^{m-1} & \alpha_1^{m-1} & \dots & \alpha_{k-1}^{m-1} \end{bmatrix}$$

where $\alpha_0, \alpha_1, \dots, \alpha_{k-1}$ is an element on the Galois field G. From the above analysis, the RS code encoding and decoding process by using the Vandermonde coefficient matrix contains a large number of Galois field multiplication operations. The complexity of coding time is O (km), and the complexity of decoding time is O (m^3). To reduce the amount of calculation, the Cauchy coefficient matrix is used to convert the multiplication operation in the $GF(2^W)$ domain into an addition operation. The Cauchy coefficient matrix is defined as:

$$G = \begin{bmatrix} \frac{1}{x_1 + y_1} & \frac{1}{x_1 + y_2} & \dots & \frac{1}{x_1 + y_k} \\ \frac{1}{x_2 + y_1} & \frac{1}{x_2 + y_2} & \dots & \frac{1}{x_2 + y_k} \\ \dots & \dots & \dots & \dots \\ \frac{1}{x_m + y_1} & \frac{1}{x_m + y_2} & \dots & \frac{1}{x_m + y_k} \end{bmatrix}$$

2) ERASURE CODE LOCALIZATION ALGORITHM CLRC DESIGN

A. Optimization plan

In the previous section, we pointed out that the RS (k, m) code achieves data redundancy through the coding and decoding process, but the cost of data reconstruction is extremely high. The fundamental reason is that the locality r is too large, which increases the load of the CPU and the network. Based on the above analysis, we propose a new localization algorithm CLRC to optimize the RS code. The basic idea is to group the data blocks based on the RS code and perform an intra-group addition operation to obtain an additional code block. Thus, it can localize the coding block group and reduce the locality r of the algorithm. Therefore, it can reduce the cost of the data reconstruction process. To distinguish the additional blocks from the RS code blocks, we refer to the former as the local code block and the latter as the global code block.

Firstly, we divided the matrix E calculated by RS (k, m) into l groups and each group has the number of t blocks. In which, the first $l - 1$ group is the data block and the last group is the global code block. Then, we perform a summation operation in every group respectively and get the local code blocks with the number of l. The relationship between l, k, m, and t is Eq. (9).

$$l = \frac{k + m}{t} \tag{9}$$

Then, the r calculation process is Eq. (10).

$$p = t \tag{10}$$

Based on comprehensive consideration of the availability and redundancy of the data, we select RS codes with the parameters of $k = 12$, $M = 4$ for encoding, which can recover the data when the damaged block is less or equal to 4. So, the value of the redundancy p is just 33.33%. The process of p calculation can be expressed in Eq. (11).

$$p = \frac{m}{k} = 33.3\% \tag{11}$$

Fig. 5 shows the structure of the CLRC (12, 4, 4) code. In which, P_i is four global code blocks when $i \in \{a, b, c, d\}$ calculated from RS (12, 4) codes in $GF(2^W)$ domain. P_i can also be four local code blocks when $i \in \{1, 2, 3, 4\}$. P_i can be obtained by XOR operation in Eq. (12).

$$P_i = D_{4i-1} + D_{4i-2} + D_{4i-1} + D_{4i}, (i = 1, 2, 3, 4) \quad (12)$$

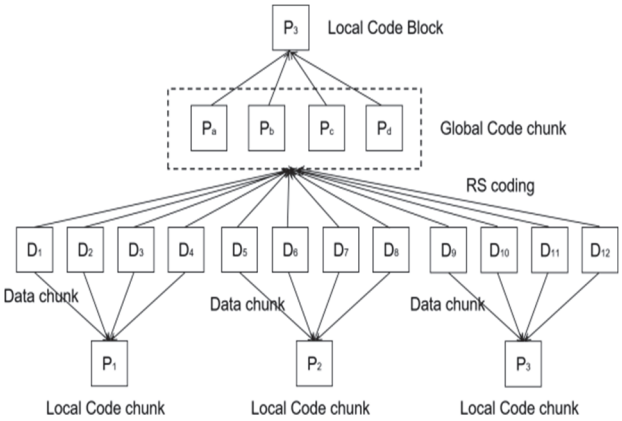

Figure 5 The structure of the CLRC (12, 4, 4) code

CLRC (12, 4, 4) has the same minimum column spacing $d_{\min} = m + 1 = 5$ as RS (12, 4), but the locality of the former $r_{CLRC} = 4$, redundancy $p_{CLRC} = 66.66\%$, locality of the latter $r_{RS} = 12$, redundancy $p_{RS} = 33.33\%$. In other words, the CLRC algorithm sacrifices 33.3% of disk space, in exchange for three times the efficiency of data reconstruction.

B. CLRC coding

Compared with the RS code, the CLRC code algorithm adds a code block to improve the locality of the algorithm and reduce the data block required for data reconstruction, thereby improving the efficiency of reconstruction. Compared with the encoding process of the two EC codes, it can be found that the CLRC code and the RS code are consistent in the coding of the global code block, and can be obtained by multiplying the original matrix and generating matrix B. The only difference is that the CLRC code needs to group the encoded data block group E and calculate the local code block separately. By analyzing the composition of the local code block, the first $l - 1$ blocks are calculated by the data block in the encoded data block group E, and the last block is obtained by the global code block in E. It is worth noting that the data block in the encoded data block group E is identical to the original data block group, which is obtained by multiplying the unit matrix I and the original data block. Therefore, the encoding process of the local code block can be divided into two parts:

(1) Adding the corresponding coefficient matrix C' to matrix B to form B'', the first $l - 1$ local code block can be obtained by multiplying B'' with the original data block group.

(2) The last local code block is obtained by the summation of the calculated global code block.

Where the coefficient matrix C' is defined as: it has l groups, and each group has the CLRC code of t blocks. For $i \in [1, l-1]$, the elements from $[1 + t(i - 1)]$ to $t \times i$-th is 1 in the i-th row of Coefficient matrix C', and the values of the rest of elements are 0.

The coding process of CLRC (12, 4, 4) is shown as follows:

(1) Initializing the values of k, m, l by the CLRC algorithm. For CLRC (12, 4, 4), it is necessary to establish a data buffer and block list. The former is used to store the data block and code block in the process of calculation, and the latter is used to record the block information in the same block group after grouping.

(2) The original data is grouped and the grouping information is recorded into the block group list for CLRC decoding.

(3) For calculating the group of the coded data block, the calculation process is similar to the RS code. The difference is that the generating matrix B contains not only the unit matrix I and global coefficient matrix C but also the local coefficient matrix. The calculated coded data block group E contains not only the data block and global code block but also the local code block.

(4) For the local code block, it is calculated by the global code block group.

(5) The calculation results of the block group are stored, and the principle of storage is that the coded data block in the same group should be stored as close as possible to the nearest node in the network topology.

In fact, in step (3), all data blocks and global code block calculations have been completed, and the local code block corresponding to all data block groups has been calculated. Therefore, the efficiency of coding can be further optimized by advancing from step (4) to step (5). The local code block corresponding to the global code block group can be calculated through the MapReduce jobs. MapReduce is assigned by Name node to the node where the global code block group is located.

C. CLRC decoding

When a data block is lost or damaged, the CLRC algorithm can recover the lost data block by pulling a certain number of data blocks from the file system. According to the number of data block exceptions, the reconstruction process can be divided into three following types:

(1) An exception occurs in only one block. In this case, it is only necessary to determine which group the block belongs to and pull other blocks from the group to perform an XOR operation to reconstruct the data.

(2) If there are multiple blocks with exceptions and each block belongs to a different group, then each block can be reconstructed by other blocks in the same group. The reconstruction process is similar to the first case.

(3) Blocks in a group cannot be reconstructed by intra-group XOR operation when multiple blocks are abnormal and two blocks belong to the same group at least. In this case, it is necessary to calculate the total number w of the data block and global code block in the abnormal block. If $w < d_{\min} = m + 1$ is not satisfied, the data cannot be reconstructed. Otherwise, the data can be reconstructed through the RS decoding process shown in Fig. 3.

4 RESULTS AND DISCUSSION

Due to the limitations of the experimental environment, we chose to build a distributed HDFS cluster using VMware virtual machines. The cluster has a total of four nodes, one of which can be used as Namenode and Datanode at the same time. The basic configuration information is shown in Tab. 2.

Table 2 Node configuration information

Memory	Hard Disk	Hadoop Version	Linux Version
2 G	20 G	2.5.0	Centos6.4

A. Coding efficiency test

To avoid the influence of different file types and networks and other factors, a circular text file with the content "CLRC test" is used in the code test. The default block size in the HDFS configuration file is changed to 64 M. Three files of size 12 × 64 M, 24 × 64 M, and 32 × 64 M on the name node can be generated (Shown in Tab. 3), and the experimental results are generated from the average value of three tests. The algorithm parameters for each set of test files as CLRC (12, 4, 4) and RS (12, 4) are tested. The experimental results are shown in Fig. 5.

Table 3 The delay time of coding

file size	12 × 64 M	24 × 64 M	32 × 64 M
RS code	45.1 s	75.4 s	115.2 s
CLRC code	62.2 s	107.2 s	191.5 s

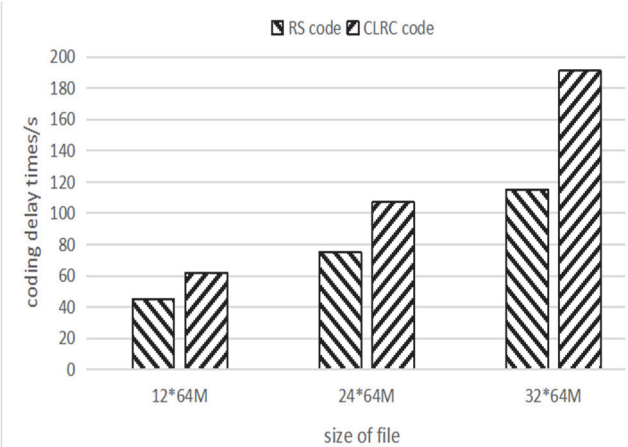

Figure 5 Comparison of coding delay time between RS code and CLRC code

Fig. 5 shows that compared with the RS code, the CLRC has a longer encoding delay time and a lower encoding efficiency. The reason is that CLRC code needs to generate four extra code blocks during the encoding process, and the encoding process is more complicated. However, the data files in HDFS are mostly uploaded once and read multiples times, and will not be modified after the file is uploaded. Therefore, it is acceptable to sacrifice storage efficiency in exchange for reading efficiency.

B. Reconstruction efficiency test

The refactoring efficiency can be measured by the size of the data read during the reconstruction process. By manually deleting the block in the HDFS we can simulate the loss of the data block. When the blocks from 1 to 4 are lost, two ways including RS and CLRC can be used to measure the size of the data and encoding time decoded.

Fig. 6 shows the comparison of the amount of data read by RS code and CLRC code during the data reconstruction. We can see that the amount of the data read is not related significantly to the number of lost blocks in the RS code. Tab. 4 shows the size of data needed to read for RS and CLRC from Fig. 6. From Tab. 4 we know that the average size of data read is 773 M for the RS code. This is because the RS code always needs to read k data blocks, no matter how many data blocks are lost. However, it also can be seen from Fig. 6 that the CLRC code only needs to read 302 M of data when one block is lost. Therefore, the CLRC code can save about 61% bandwidth and I/O consumption during the reconstruction of a single data block.

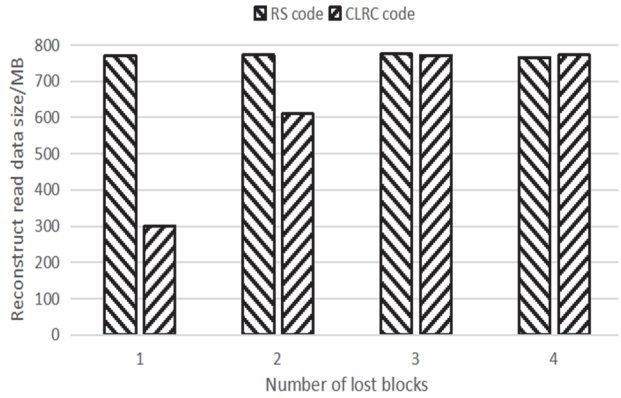

Figure 6 Comparison of reading data size between RS code and CLRC code

Table 4 the size of data needed to read

block number	1	2	3	4
RS code	771 M	773 M	775 M	765 M
CLRC code	300 M	610 M	771 M	772 M

Fig. 7 shows the difference in reconstruction time between the RS code and CLRC code. We can see that the CLRC code has no obvious advantage for the reconstruction time when multiple blocks are lost. But for the loss of a single block, the CLRC code can obviously shorten the decoding time which is only about 59% of the RS code (shown in Tab. 5). In the actual production environment, the condition of the loss of a single block is about 90% among all failures. Thus, the CLRC can effectively reduce the time of data reconstruction and improve the efficiency of the data recovery while the data fault tolerance is ensured.

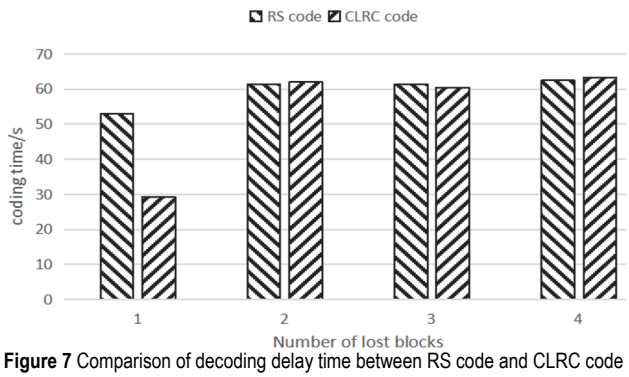

Figure 7 Comparison of decoding delay time between RS code and CLRC code

Table 5 The delay time of decoding

block number	1	2	3	4
RS code	53 s	61.5 s	61.5 s	62.5 s
CLRC code	29.3 s	62 s	60.5 s	63.3 s

5 CONCLUSION

As the amount of data continues to rise, HDFS has gradually failed to meet the demand for big data for storage capacity. In this paper, HDFS performance is optimized by improving storage space utilization. Firstly, it is found that the waste of storage space resources was caused by the replica strategy of HDFS. Additionally, the EC algorithm helps improve the rate of the storage space utilization of HDFS while ensuring data availability. Then two

commonly used EC codes are introduced with their advantages and disadvantages. On this basis, a localization algorithm CLRC based on RS code is designed and implemented. The experimental results show that the CLRC code has the same or even better fault tolerance than the RS code, and as well it also has higher reconstruction efficiency when a single block is unavailable.

Acknowledgment

In the preparation of this paper, we were supported by the project (No. 61462004, 61662002) of the National Natural Science Foundation of China, the training project (No. 201710405025, No. 201810405010, No. 201810405020) of the innovation and entrepreneurship for college student, and the open fund project (No. HJSJYB2016-4) of the research center of Nuclear Technology Application Engineering (Ministry of Education), and scientific and technological research project of Henan province (No. 202102110275), and key scientific research project of higher education institutions of Henan province (No. 20A520032), and Jiangxi Engineering Laboratory on Radioactive Geoscience and Big Data Technology (No. JELRGBDT201905).

6 REFERENCES

[1] Liu, X., Zhao, D., Xu, L. et al. (2015). A distributed video management cloud platform using hadoop. *IEEE Access*, *3*, 2637-2643. https://doi.org/10.1109/ACCESS.2015.2507788

[2] Yacchirema, D. C., Sarabia-Jácome, D., Palau, C. E. et al. (2018). A smart system for sleep monitoring by integrating IoT with big data analytics. *IEEE Access*, *6*, 35988-36001. https://doi.org/10.1109/ACCESS.2018.2849822

[3] Balaji, S. B., Krishnan, M. N., Vajha, M., Ramkumar, V., Sasidharan, B., & Kumar, P. V. (2018). Erasure coding for distributed storage: an overview. *Science China (Information Sciences)*, *61*(10), 7-51. https://doi.org/10.1007/s11432-018-9482-6

[4] Tang, D. (2016). Research of Methods for Lost Data Reconstruction in Erasure Codes over Binary Fields. *Journal of Electronic Science and Technology*, *14*(01), 43-48.

[5] Shahabinejad, M., Khabbazian, M., & Ardakani, M. (2014). An efficient binary locally repairable code for Hadoop distributed file system. *IEEE Communications Letters*, *18*(8), 1287-1290. https://doi.org/10.1109/LCOMM.2014.2332491

[6] Dau, H. & Milenkovic, O. (2017). Optimal repair schemes for some families of full-length Reed-Solomon codes. *2017 IEEE International Symposium on Information Theory (ISIT)*, 346-350. https://doi.org/10.1109/ISIT.2017.8006547

[7] Yao, L., Lu, J., Liu, J. et al. (2018). A Secure and Efficient Distributed Storage Scheme SAONT-RS Based on an Improved AONT and Erasure Coding. *IEEE Access*, *6*, 55126-55138. https://doi.org/10.1109/ACCESS.2018.2872749

[8] Higai, A., Takefusa, A., Nakada, H. et al. (2014). A study of replica reconstruction schemes for multi-rack hdfs clusters. *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. IEEE Computer Society*, 196-203. https://doi.org/10.1109/UCC.2014.28

[9] Ishengoma, F. (2013). HDFS+: Erasure-Coding Based Hadoop Distributed File System. *International Journal of Scientific & Technology Research*, *2*(9), 11-12.

[10] Wylie, J. J. & Swaminathan, R. (2012). System and method for determining the fault-tolerance of an erasure code: US, US 8127212 B2. 33.

[11] Dimakis, A. G., Godfrey, P. B., Wu, Y. (2010). Network coding for distributed storage systems. *Information Theory, IEEE Transactions on*, *56*(9), 4539-4551. https://doi.org/10.1109/TIT.2010.2054295

[12] Rashmi, K. V., Shah, N. B., & Kumar, P. V. (2011). Optimal accurate-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. *Information Theory, IEEE Transactions on*, *57*(8), 5227-5239. https://doi.org/10.1109/TIT.2011.2159049

[13] Li, Y., Chan, W. Y., & Blostein, S. D. (2017). On design and efficient decoding of sparse random linear network codes. *IEEE Access*, *5*, 17031-17044. https://doi.org/10.1109/ACCESS.2017.2741972

[14] Shah, N. B., Rashmi, K., Kumar, P. V. et al. (2012). Interference alignment in regenerating codes for distributed storage: necessity and code constructions. *Information Theory, IEEE Transactions on*, *58*(4), 2134-2158. https://doi.org/10.1109/TIT.2011.2178588

[15] Cadambe, V. R., Jafar, S. A., Maleki, H. et al. (2013). Asymptotic interference alignment for optimal repair of mds codes in distributed storage. *Information Theory, IEEE Transactions on*, *59*(5), 2974-2987. https://doi.org/10.1109/TIT.2013.2237752

[16] Gopalan, P., Huang, C., Simitci, H. et al. (2012). On the locality of codeword symbols. *Information Theory, IEEE Transactions on*, *58*(11), 6925-6934. https://doi.org/10.1109/TIT.2012.2208937

[17] Su, Y. S. (2017). On constructions of a class of binary locally repairable codes with multiple repair groups. *IEEE Access*, *5*, 3524-3528. https://doi.org/10.1109/ACCESS.2017.2675878

[18] Sun, Y., Song, H., Jara, A. J. et al. (2016). Internet of things and big data analytics for smart and connected communities. *IEEE access*, *4*, 766-773. https://doi.org/10.1109/ACCESS.2016.2529723

[19] Miyamae, T., Nakao, T., & Shiozawa, K. (2014). Erasure code with shingled local parity groups for efficient recovery from multiple disk failures. *Proceedings of the 10th USENIX conference on Hot Topics in System Dependability*, 5-5.

[20] Tamo, I., Papailiopoulos, D. S., & Dimakis, A. G. (2013). Optimal locally repairable codes and connections to matroid theory. *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, 1814-1818. https://doi.org/10.1109/ISIT.2013.6620540

Contact information:

Ying FANG
Shangqiu Normal University, China

Hai TAN
(Corresponding author)
1) Nanjing Audit University, China
2) East China University of Technology, China
E-mail: ted.cyber@foxmail.com

Shuaifang WANG
(Corresponding author)
Shangqiu Normal University, China
E-mail: tongfangjuming@126.com

Xin ZHANG
East China University of Technology, China

Gejian LIAO
East China University of Technology, China

Jun ZHANG
East China University of Technology, China