Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5574255, 14 pages
https://doi.org/10.1155/2021/5574255

Research Article

A New Method for Reconstructing Data on a Single Failure Node in the Distributed Storage System Based on the MSR Code

Miao Ye,[1,2] **Ruoyu Wei**,[1] **Wei Guo**,[3] **Qiuxiang Jiang**,[4] **Hongbing Qiu**,[1,2] **and Yong Wang**[3,5]

[1]*School of Information and Communications, Guilin University of Electronic Technology, Guilin Guangxi 541004, China*
[2]*Key Laboratory of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin 541004, China*
[3]*School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin Guangxi 541004, China*
[4]*School of Electronic Engineering and Automation, Guilin University of Electronic Technology, Guilin Guangxi 541004, China*
[5]*Guangxi Education Collaborative Innovation Center for Big Data and Network Security, Guilin University of Electronic Technology, Guilin Guangxi 541004, China*

Correspondence should be addressed to Hongbing Qiu; qiuhb@guet.edu.cn

Received 1 February 2021; Revised 22 February 2021; Accepted 19 March 2021; Published 1 April 2021

Academic Editor: Chi-Hua Chen

As a storage method for a distributed storage system, an erasure code can save storage space and repair the data of failed nodes. However, most studies that discuss the repair of fault nodes in the erasure code mode only focus on the condition that the bandwidth of heterogeneous links restricts the repair rate but ignore the condition that the storage node is heterogeneous, the cost of repair traffic in the repair process, and the influence of the failure of secondary nodes on the repair process. An optimal repair strategy based on the minimum storage regenerative (MSR) code and a hybrid genetic algorithm is proposed for single-node fault scenarios to solve the above problems. In this work, the single-node data repair problem is modeled as an optimization problem of an optimal *Steiner tree* with constraints considering heterogeneous link bandwidth and heterogeneous node processing capacity and takes repair traffic and repair delay as optimization objectives. After that, a hybrid genetic algorithm is designed to solve the problem. The experimental results show that under the same scales used in the MSR code cases, our approach has good robustness and its repair delay decreases by 10% and 55% compared with the conventional tree repair topology and star repair topology, respectively; the repair flow increases by 10% compared with the star topology, and the flow rate of the conventional tree repair topology decreases by 40%.

1. Introduction

Distributed storage first refers to a storage system built on large-scale and low-cost commercial hardware (network-attached storage (NAS)). It uses a hardware and software cooperation mechanism to encapsulate the interface to provide external programmable calls and command access [1]. Compared with the traditional centralized storage system, the design framework of a distributed storage system brings many advantages. For example, the storage capacity of a cluster can be expanded by constantly adding new underlying hardware devices or upgrading old hardware devices, and the adoption of large-scale and inexpensive commercial hardware reduces the cost of cluster construction. The advantages of a distributed storage system enable it to cope with the various demands of mass data storage. Therefore, distributed storage systems are widely favored worldwide.

With the increase in the number of storage nodes and the diversity and complexity of storage nodes in distributed storage systems, nodes often fail [2]. There are two ways to improve the reliability of a storage system: the copy mode [3] and erasure code [4]. The copy method stores the data in multiple copies, and when the storage node where one copy is located fails, the storage system can switch the service to the remaining copies. This method does not involve special coding and reconstruction algorithms; it has good fault-tolerant performance but very low storage utilization [5]. The erasure code method uses the data encoding method to store the coded blocks obtained after data encoding. When a storage node where a coded block is located fails, the

storage system can calculate the coding block and recover the fault node. Erasure code technology has the advantage of low storage costs; however, its repair cost is high, and it generates considerable repair traffic that occupies considerable bandwidth.

Although the copy strategy has advantages over the erasure correction code in terms of the speed of reading surviving data and recovering lost data, its high storage space cost is contrary to the original intention of distributed storage systems. The erasure code can provide almost the same reliability as the multireplica strategy while effectively improving the utilization of storage space and providing higher data writing efficiency, making it widely used in distributed storage systems.

Traditional erasure codes, such as the *Vandermonde* and Reed-Solomon (RS) codes [6] $(k+r, k)$, divide the original data of size m into k data blocks with the size of M/k. By applying the *Vandermonde* matrix in the *Galois* domain $\mathrm{GW}(2^w)$, the loss within any r block in $k+r$ blocks can be obtained by obtaining all the data from any k nodes from the remaining living nodes through the replacement node, and the codec operation can be performed to obtain the original data and repair the lost data. Traditional coding usually needs to obtain the data with the size of M from k nodes to recover the failed nodes and requires other nodes to transmit much repair traffic; this is the main reason for a large amount of repair traffic in the correction code method. The minimum storage regenerative (MSR) code can recover the fault node by performing more complex encoding and decoding operations on the original data and can recover the fault node when the total amount of data transmitted by the node providing the data is M/k, which greatly reduces the repair flow and repair delay so that the stored data can be repaired faster.

When the erasure code is applied to repair a failed node, the storage system provides a new node in the cluster topology to replace the failed node. This node is called the *newcomer*. The *newcomer* downloads data from several surviving nodes and performs the data recovery operation. The nodes that provide data to the *newcomer* are called the *provider* nodes. In the traditional data repair process, the repair topology in which *provider* nodes transmit the repair data to a *newcomer* is of the star type. The repair only starts when all *provider* nodes transmit their data to the *newcomer*, so the delay in repair depends on the bottleneck bandwidth. To reduce the repair delay, researchers proposed a tree repair topology based on the star repair topology that creates a tree repair topology with the *newcomer* as the root and then selects links with higher available bandwidth to join the topology to reduce the repair delay.

Compared with the star repair topology, the traditional tree repair topology does improve the data repair rate, but it brings more traffic overhead to the storage cluster network. In the construction of the repair tree, the optimal repair topology with the best repair delay can be obtained. This also causes more node selection problems, while the traditional greedy algorithm causes more computing overhead and increases the delay in repairing failed nodes due to the increase in nodes. Additionally, compared with the conventional star repair topology, the robustness of the conventional tree repair topology is very low when the repair operation of a failed node in the storage cluster is in progress. If another node fails in the repair topology, the traditional star topology just picks a live node to continue the repair. Nevertheless, when node losses occur in a traditional tree repair topology, all nodes in a subtree with a lost node as the root are impacted and the repair topology needs to be reconstructed, which results in a reduction in the repair rate.

In recent years, some improvements of the data construction for a single failure node have emerged, but the performances of the time delay, the transmitting traffic, and the robustness of the repair topology still need to be improved significantly. As the *Steiner tree* with constrained conditions for the data construction is a kind of NP-hard problem, it is hard to find the global optimal solution. For the above problem, in this paper, some nodes with excellent bandwidth properties in the storage cluster are added to the repair node set to participate in the construction of the repair tree as alternative intermediate nodes, and based on the characteristics of the MSR coded intermediate node computation, a *Steiner tree* model with constraints is established to optimize the repair time delay and the traffic flow in the repair topology. Then, by designing a hybrid genetic algorithm to calculate the approximate global optimal solution, finally, an optimal repair topology construction method based on the MSR code and a hybrid genetic algorithm is proposed. The simulation results show that under the same size of MSR codes, the repair delay is 70%-90% of that of the traditional tree repair topology and only 35%-45% of that of the traditional star repair topology. Although the repair traffic in the topology is slightly improved compared to the traditional star topology, the repair traffic is only 45%~60% of that of the traditional tree repair topology, and the constructed repair topology has higher robustness.

In summary, the innovations of this paper are as follows:

(1) We consider the heterogeneity of the nodes, and the node set participating in the repair joins the nodes with better bandwidth properties in the cluster and participates in the construction of the repair topology as alternative intermediate nodes to obtain the repair tree with the optimal repair characteristics and to enhance the robustness of the repair topology simultaneously. As far as we know, there is no work that considers time delay and robustness at the same time in the available literature we can find

(2) The MSR code with better storage characteristics is used to construct an optimal repair tree with constraints based on the calculation of the intermediate nodes, which reduces the repair traffic cost of erasure code tree repair

(3) To solve the NP-hard problem of the optimal repair tree, a hybrid genetic algorithm is designed according to the background characteristics of the problem to obtain the optimal repair *Steiner tree* based on the repair node set

The rest of this article is organized as follows. Section 2 introduces the research status of the distributed storage

system and erasure code. Section 3 analyzes the characteristics and topology structure of the MSR code, the influence of the repair flow, and the node performance during the repair process to deduce the definition of the repair topology problem that needs to be constructed in this paper, and a mathematical model is established. Section 4 describes in detail the design and algorithm flow of each operator of the genetic algorithm proposed in this paper. In Section 5, experiments are designed to compare and analyze the performance differences between the proposed algorithm and the traditional star and tree repair algorithms and to analyze the influence of the parameters in the algorithm. Section 6 summarizes the whole paper and proposes future research directions.

2. Related Work

There are two ways to guarantee the reliability of nodes. One is multireplica, and the other is the erasure code [7]. Erasure codes originated in the field of communication and were originally used to solve the problems of error detection and error correction in communication transmission. Later, erasure codes were gradually applied to data error detection and error correction in storage systems to improve the reliability of storage systems and were gradually improved and popularized with the characteristics of storage system applications [8].

With the increasing scale of storage systems, the multireplica method has difficulty meeting the requirements of mass storage systems for redundant backup mechanisms in disk utilization and fault tolerance [9]. The advantages of the erasure code method with respect to its low storage cost become increasingly obvious, and research on erasure code technology has become a research hotspot in the industry. At present, the research literature on the disaster recovery strategy of erasure code methods mainly focuses on improving the coding mechanism and the construction of topology in the repair process.

In terms of the improved coding mechanisms, Wang [10] fixed defects with excessive local characteristics in the traditional minimum bandwidth regenerated (MBR) code and variable fractional repetition (VFR) code according to the idea of different data heat and different repeatability. Lin et al. [11] proposed a packet repair code (PRC) to overcome the high cost of multimode failure repair. Teng et al. [12] proposed sparse random erasure codes based on the high probability of the full rank property of a random matrix with equal row weights and sparsity. Based on the RS code, the Microsoft team designed a local reconstruction code (LRC) [13]. Dimakis et al. [14] obtained a regenerated code by improving the basic maximum distance separable (MDS) erasure code. Compared with the traditional erasure code, the regenerated code reduced the repair flow rate. Jiekak et al. [15] proved that the regenerated code repair traffic in the process of repairing the data transmitted by a failed node is approximately equal to 16% of the RS code at the same level.

In terms of repair topology construction, Huang [16] proposed a pipeline repair scheme, but it was essentially a serial transmission mode and could not reach the optimal repair rate. Jia et al. [17] considered the heterogeneity of bandwidth in a storage cluster, constructed a mathematical model based on this, and solved the routing algorithm for repairing data provided by a data-providing node to a newborn node. Gong et al. [18] proposed selection criteria for nodes and tree repair topologies provided by data in the repair process for storage clusters with regenerative code mechanisms and analyzed their advantages. Wan [19] proposed a signal-to-noise ratio (STNR) algorithm for repairing traffic integration at intermediate nodes, but it did not attempt to further optimize the coding mechanism. Zhang et al. [20] considered the actual network topology. They used a field-programmable gate array (FPGA) to replace the switch for traffic integration, but due to the high price of FPGA, it is difficult to apply it in an actual environment.

In this paper, we consider constructing an optimal *Steiner tree* topology with constraints based on the MSR code with better storage characteristics and a hybrid genetic algorithm through a parallel repair network, which not only reduces the bandwidth and computational cost but also increases the robustness of the topology and effectively solves the problem of the low efficiency of a serial repair.

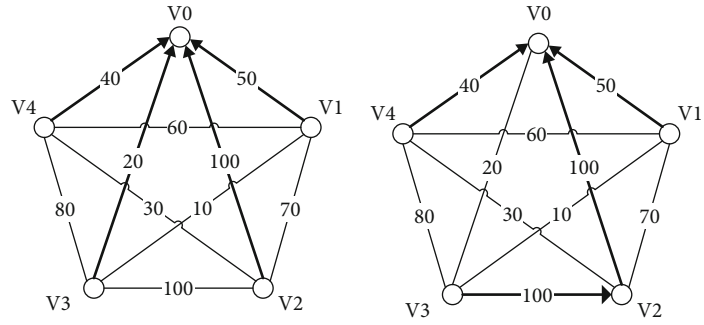
3. Modeling and Formulation of the Single-Node Failure Repair Problem

Given the problems mentioned above in the research of the erasure code repair process, this section analyzes the influence on the single-node fault repair process from three aspects: repair topology, topology repair traffic, and node processing capacity heterogeneity.

3.1. MSR Code. To further reduce the traffic generated by repair operations in clustered networks, Wu et al. proposed the minimal storage regenerating code based on regenerative code [21]. A more complex codec operation of the original data is used to reduce the size of the data block sent by each *provider* in the process of repair and further reduce the repair traffic in the overall topology.

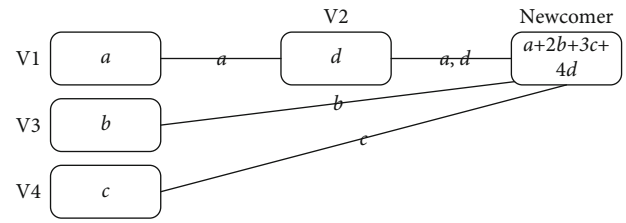
The MSR code with (n, k, d) means the original data block with a size of M is transformed into $k * (d - k + 1)$ encoded blocks which are stored in n storage nodes. Every storage node stores the encoded blocks with the same size of M/k. In general, when a storage node is selected as the provider node, the encoded block with a size of M/k will be transformed into a data block with a size of $M/k(d - k + 1)$ which is sent to the newcomer. When the *newcomer* receives $d * M/(k(d - k + 1))$ data blocks from d providers, the decoding operation can be performed to restore the data of size M/k stored on the failed node. As the data is transmitted by each *provider*, $\beta = M/(k(d - k + 1))$, under the MSR code disaster tolerance mechanism, the *newcomer* also needs to directly access the data from the $d - k + 1$ node during the repair process. Therefore, any repair topology that uses the MSR code must satisfy [22]

$$\text{degree}_{\text{newcomer}} \geq d - k + 1. \quad (1)$$

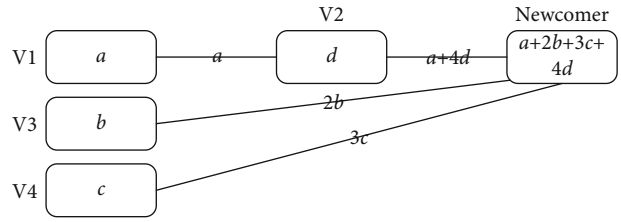


(a) Traditional star repair topology (b) Traditional tree repair topology

Figure 1: Traditional star repair topology and tree repair topology (unit Mbps).



(a) The repair traffic of each link in the case of the traditional tree topology



(b) The repair traffic of each link in the case of the tree topology with an intermediate node aggregation mechanism

Figure 2: Repair flow with the traditional tree repair topology and tree repair topology adopted an intermediate node aggregation.

3.2. Optimal Bottleneck Bandwidth Repair Tree. To demonstrate the optimization of repair delay between the tree repair topology and the star repair topology under the same environment, a network topology with an erasable code disaster recovery cluster is assumed to have five nodes, V0, V1, V2, V3, and V4. The available bandwidth of the link between them is also provided (unit Mbps), and the data of $M = 600\,\text{MB}$ are stored in the nodes. In the case of (5, 3, and 4) MSR codes and the traditional star repair topology, V0 is adopted as a *newcomer*, as shown in Figure 1(a). Each node of the topology needs to pass $\beta = M/(k(d-k+1)) = 100\,\text{MB}$ during the repair process, and the repair delay is $t = \beta/20\,\text{Mbps} = 5\,\text{s}$. The tree repair topology is shown in Figure 1(b), the repair delay is $t = \text{Max}(\beta/40\,\text{Mbps}, 2\beta/100\,\text{Mbps}) = 2.5\,\text{s}$, and the bottleneck link changes from (V0, V3) to (V0, V4). The tree repair topology is more diversified in the choice of the optimal repair path than the star topology and can make full use of some relatively idle links in the cluster network to improve the overall repair efficiency.

Therefore, the problem of finding the optimal repair delay topology can be transformed into a classical minimum spanning tree problem. A tree repair topology T is found in the whole network topology, and the bottleneck repair delay is required to be the shortest which satisfies

$$T = \min(\max\{t\}), \tag{2}$$

where T represents the transmission delay on each link in the current repair topology.

3.3. Repair Traffic Overhead. Although the tree topology promotes the efficient repair of failure nodes, compared with a star topology, when the overall repair flow is large and when the scale of rectifying the deleted code increases, the topological structure of the repair becomes more complicated. Correcting the gap flow is also increasingly clear because the traffic of each link is the number of *providers* in the subtree times β. The links that are close to the *newcomer* experience more burden than the links close to the *provider*; hence, more network problems, such as congestion, occur.

Given the problem that the total amount of repair traffic in the tree repair topology is too large, considering the

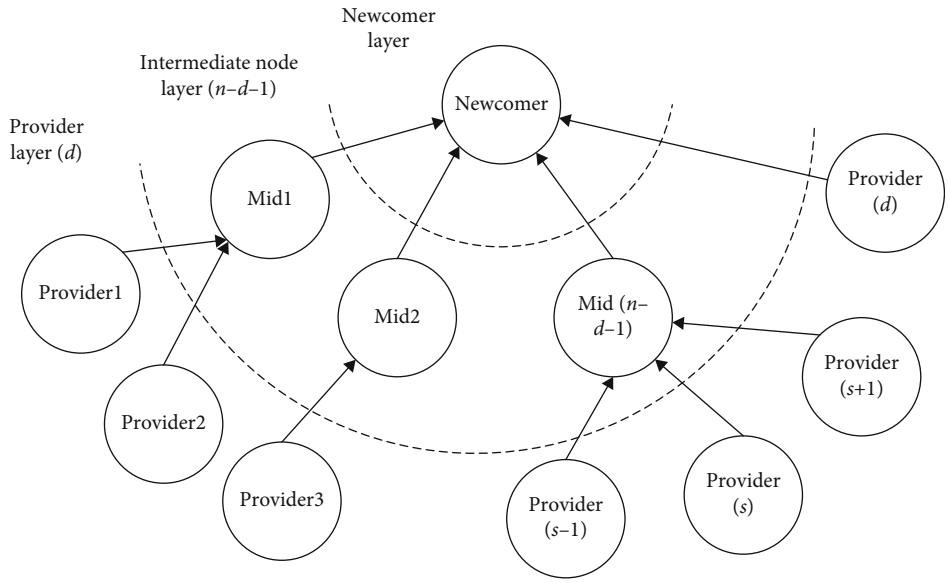


Figure 3: The three-tier repair structure of the topology.

encoding and decoding nature of the erasure code itself, the computing operation finally performed at the *newcomers* can be divided into several aggregation operations and then implemented at the intermediate nodes.

Figure 2(a) shows the repair traffic on each link in the topology when the traditional tree repair is performed. When the data of $a + 2b + 3c + 4d$ need to be repaired, the traffic in the overall repair topology is $a + b + c + d + a + b$, which is 6β in total. In Figure 2(b), the data are preliminarily calculated and processed at each node; then, the repair flow of the subtrees is consolidated at the middle node M1, and the complete decoding operation is finally completed at the *newcomer* node. At this time, the flow in the repair topology is $a + 2b + 3c + 4d + (a + 2b)$, which is 5β in total. Compared with the traditional tree repair topology, the repair flow is reduced by 1 unit.

After the traffic aggregation operation of the intermediate nodes is performed, the traffic saved by each intermediate node is 2 less than its degree in the repair topology. The repair traffic saved by the overall topology is shown in

$$\text{traffic}_{\text{save}} = \sum \text{degree}_{\text{mid}} - 2. \tag{3}$$

Therefore, the traffic saved by the topology is directly proportional to the number of intermediate nodes and nodes in the subtree of the intermediate nodes. However, too many intermediate nodes can generate much extra transmission overhead in the topology, which is contrary to the research goal of the erasure code repair process. Therefore, a balance must be struck between the topology repair rate and the traffic savings. The minimum spanning tree problem mentioned in the previous section is transformed into a *Steiner tree* problem with constraints when a mechanism for the traffic aggregation operation of the intermediate nodes is added to the traditional tree repair topology. In the network topology, a *newcomer* becomes the root of tree T to fix the topology,

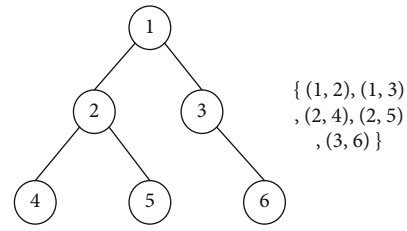


Figure 4: An example of tree coding.

and the repair time and the flow topology are minimized to satisfy

$$T = \min \left\{ c_{\text{time}} \max \left(\frac{\beta}{\omega} \right) + \frac{c_{\text{traffic}}}{\sum \text{degree}_{\text{mid}} - 2} \right\}, \tag{4}$$

where $c_{\text{time}} + c_{\text{traffic}} = 1$, c_{time} represents the repair delay weight coefficient, and c_{traffic} represents the repair traffic weight coefficient. Determining the constructed repair topology takes precedence over the repair delay or repair traffic and represents the available bandwidth of the link.

3.4. Heterogeneous Node Processing Capabilities. Although most research studies on the repair process of erasure codes ignore the processing delay of each node for data by default, the heterogeneity of nodes caused by the nature of the distributed storage system makes the processing speed of each node in the cluster to repair data vary. Compared with most erasure codes, MSR codes cost less in data repair operation at the encoding and decoding operation. In addition, the intermediate node processing mechanism in Section 3.3 will make massive data that are gathered in the middle of provider nodes for processing; so for this repair scenario, we cannot ignore the node processing data delay effect on the efficiency of the whole repair process.

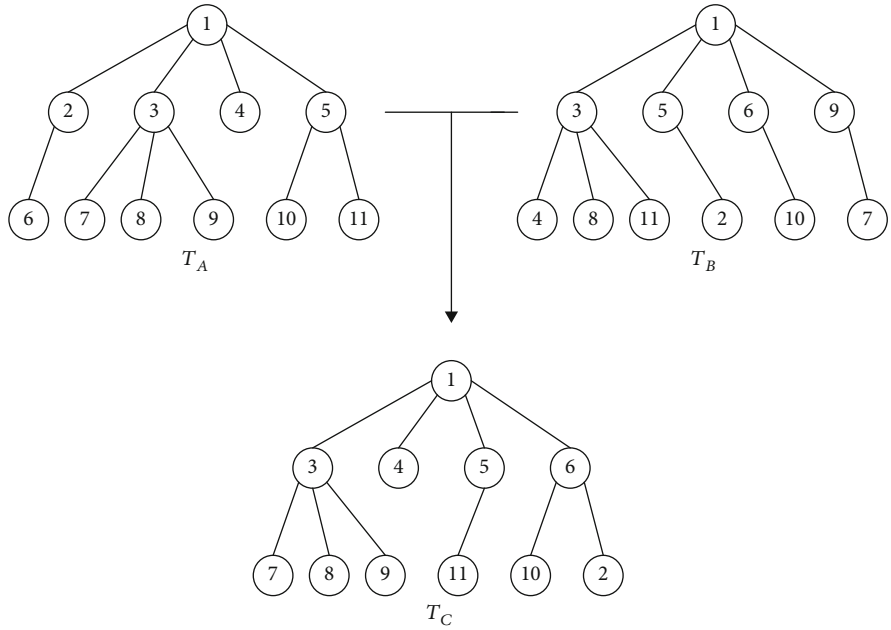


Figure 5: Example of a crossover operator.

This section adopts the definition of node processing capacity in the repair topology proposed by Qi et al. [23]; uses the sequence $x_1, x_2, \cdots, x_k, \cdots, x_m$ to represent the factors that affect the processing capacity of a node, such as a disk I/O, CPU cores, and main frequency; and assigns the corresponding weight factors $\gamma_1, \gamma_2, \cdots, \gamma_k, \cdots, \gamma_m$, where M represents the corresponding serial number. Therefore, the processing capacity of each storage node in the repair topology can be expressed as

$$\text{process}_i = \sum_{k=1}^{m} x_k r_k. \tag{5}$$

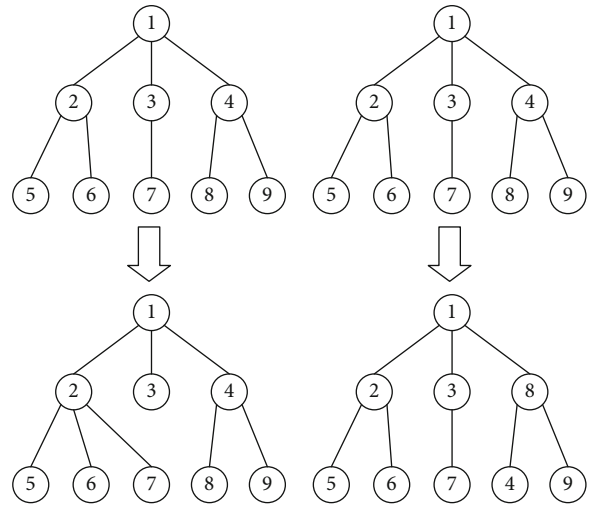
It is assumed that the repair traffic on node V_i is represented by D_i, and the processing time of node V_i for reading, coding, and forwarding the repair traffic is expressed as

$$t_{V_i} = \alpha \frac{D_i}{\text{process}_i}, \tag{6}$$

where α is the capacity conversion coefficient. When considering the processing delay of the intermediate nodes, the repair delay $t(V_i, V_j)$ of the link (V_i, V_j) is converted from β/ω to

$$t_{(V_i,V_j)} = \alpha \frac{D_i}{\text{process}_i} + \frac{\beta}{\omega(V_i, V_j)} + \alpha \frac{D_j}{\text{process}_j}. \tag{7}$$

3.5. The Proposed Multiobject Optimal Model of the Single Failure Node. Based on the above analysis of the MSR code coding mechanism and the three main problems in the repair process of the tree repair topology, the optimal repair topology model based on the MSR code and a genetic algorithm is obtained in this paper.

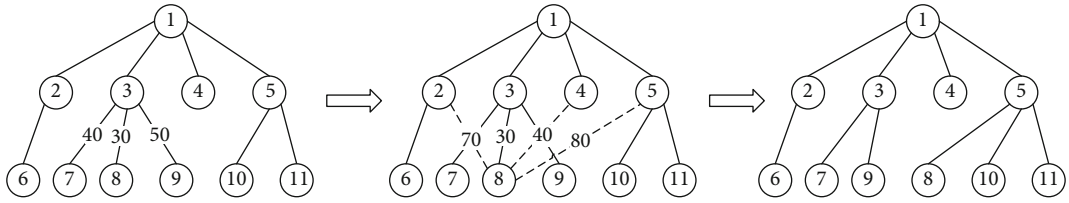


(a) Mutation operator 1 (b) Mutation operator 2

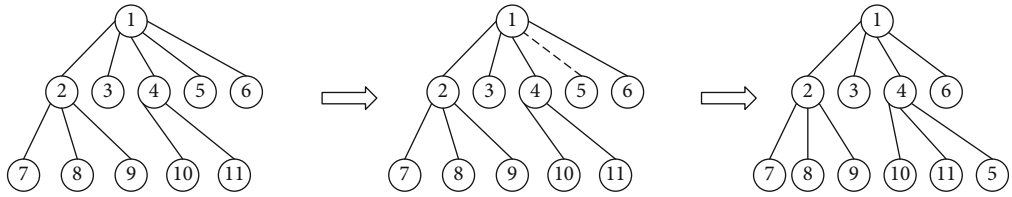
Figure 6: Example of the mutation operator.

This section simplifies the storage cluster topology based on the MSR code disaster recovery mechanism represented by (n, k, d) and uses a completely undirected connectivity weight graph $G = (V, E, W)$ to represent the topology structure of the distributed storage system, where $V = \{v_1, v_2, \cdots, v_n\}$ represents the collection of n nodes in the distributed storage system; $E = \{e_1, e_2, \cdots, e_m\}$, where $e_s(1 \le s \le m) = (v_i, v_j)$ represents the links that exist between nodes v_i and v_j in the topology; and $W = \{\omega(v_i, v_j) \mid (v_i, v_j \in E)\}$ represents the available bandwidth on the link and the processing capacity of the v_i node itself when $v_i = v_j$.

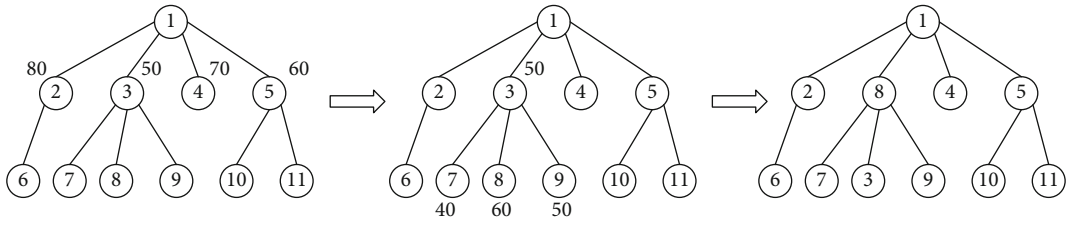
For the single-node failure scenario, to ensure the minimum repair delay and minimum repair flow at the same



(a) Local search for bandwidth



(b) Local search for traffic



(c) Local search for node processing power

Figure 7: Examples of the proposed local search operators.

time, a *Steiner tree* with constraints is constructed and satisfies

$$\min \quad \left\{ c_{\text{time}} \max \left[\alpha \frac{D_i}{\text{process}_i} + \frac{\beta}{\omega(V_i, V_j)} + \alpha * \frac{D_j}{\text{process}_j} \middle| (V_i, V_j) \in E \right] + \frac{c_{\text{traffic}}}{\sum_{\text{mid}-1}^{n-d-1} \text{degree}_{V_{\text{mid}}} - 2} \right\}$$

$$\text{s.t.} \quad \text{degree}_{\text{newcomer}} \geq d - k - 1. \tag{8}$$

The *Steiner tree* topology has a three-tier structure, as shown in Figure 3. At the top, the *newcomer* node is taken as the root, and the constraint in formula (1) is met to ensure that the amount of data required to restore the data block of size M/k is available. The intermediate layer consists of $n - d - 1$ intermediate nodes, which are responsible for the aggregation of the repair flow. The bottom layer contains d *provider* nodes connected to the intermediate node and the root node and provides the data needed to repair the failed node. The advantage of the three-layer structure is that it can avoid some weak links in the cluster topology by leveraging the advantage of the tree repair topology to reduce the repair delay. Second, the aggregation mechanism of the intermediate nodes can reduce the repair flow in the topology. In addition, the three-layer topology also limits the extra transfer overhead imposed by the intermediate nodes to the greatest extent so that the repair traffic in the whole topology can be reduced as much as possible. Finally, compared with the traditional tree topology, which cannot tolerate a secondary node failure, this paper designed a three-layer structure. It can avoid reconstructing the topology when the number of nonintermediate failure nodes is less than $n - d - 1$. The intermediate node can also be used as a substitute when a *provider* node fails; this allows the repair flow to complete the repair process. In exchange for a more robust topology, there is a small increase in the flow repair cost.

4. The Proposed Hybrid Genetic Algorithm

As shown in the model in Section 3.5, the goal of this paper is to improve the efficiency of repairing failed nodes in a distributed storage cluster topology with heterogeneous available bandwidth to increase the node processing performance and to reduce the repair traffic generated during the repair process as much as possible. After adding intermediate nodes to realize the aggregation mechanism of repair traffic, the topological problem of constructing a tree repair structure has been transformed from the traditional minimum spanning tree problem to the *Steiner tree* problem. However, under the repair scenario in this paper, the problem differs from the traditional *Steiner tree* problem in the following aspects. First, due to the nature of the MSR code, the degree constraint exists in the root node of the tree. Second, the root node of the tree has a degree constraint. Third, the *d provider* nodes used as leaves and the $n - d - 1$ nodes used as intermediate nodes are not fixed, and the nodes themselves have heterogeneous processing capabilities. These differences prevent us from using the traditional *Steiner tree* algorithm to solve this problem. In this paper, we design an optimal repair topology construction method based on a hybrid genetic algorithm.

In contrast to the classical mathematical approaches, stochastic optimization methods (such as an evolutionary

Encoding and decoding mode and population initialization
1: Assuming that the fault node number is r, add r to A and add else $n-1$ node to B, $s=0$
2: Choose $b \in B$, connect r and b as an edge of tree T, $A = A \cup \{b\}$ and $B = B - \{b\}$, $s = s+1$;
3: **if** $s < d-k+1$ **then** go to step 2
4: **else** go to step 6
5: **end if**
6: Take any element from set A and random element b from set B, join a and b as an edge and add it to tree T, and $B = B - \{b\}$, $k = k+1$, $s = s+1$;
7: **if** $s = n-1$ **then** the construction of individual T is completed;
8: **else** go to step 6.
9: **end if**

Algorithm 1

Crossover operator
1: Merge the intermediate node set $m\text{Node}_A(2,3,4,5)$ and $m\text{Node}_B(3,5,6,9)$ of T_A and T_B, obtain the intermediate node candidate $c\text{Node}(2,3,4,5,6,9)$, randomly select $d-k+1$ intermediate node for the intermediate node set $m\text{Node}_C(3,5,6,9)$ of the crossover descendant.
2: Execute Algorithm 1 based on the intermediate node set $m\text{Node}_C(3,4,5,6)$ to obtain the crossover subgeneration T_C.

Algorithm 2

Framework of ensemble learning for our system
1: p = random [0, 1].
2: **if** $p < 0.5$ **then** go to step 5
3: **else** go to step 7
4: **end if**
5: Function (mutation operator 1)
6: Select any leaf node and add it to another intermediate node or root node at random to obtain the mutation subtree, go to step 11
7: Function (mutation operator 2)
8: Select any leaf node and an intermediate node.
9: **if** links exist between the leaf node and all leaf nodes of the intermediate node **then** swap the positions of the two nodes in the repair topology to obtain the mutation subtree.
10: **end if**
11: EndFunction

Algorithm 3

algorithm) only utilize the objective function value with lower requirements for the continuity and differentiability of the objective function, which are suitable for the most complex engineering optimization problems [24–26]. As one of the well-developed stochastic optimization methods, a genetic algorithm [27–29] is adopted to design the solving algorithm to resolve our problem defined in Equation (8).

4.1. Encoding and Decoding Mode, Population Initialization, and Fitness Function. Since the subsequent crossover and mutation operations of a genetic algorithm are carried out based on population coding, the advantages and disadvantages of the population coding mode directly affect the search efficiency of the genetic algorithm. This section uses edge coding [30] and takes the edge set of the tree repair topology itself as the code, as shown in Figure 4. In a tree topology scene with a large number of constraints for the algorithm designed in this paper, edge coding can avoid a large number of infeasible solutions in this scene by adopting the crossover and mutation design, which is based on the characteristics of the tree topology and makes the execution of genetic operators more efficient.

According to the following, the tree repair topology constructed by the algorithm proposed in this paper should satisfy multiple constraints, including the constraint that all *provider* nodes are leaf nodes. The number of intermediate nodes is $n-d-1$, and the degree constraint condition is given in formula (1). Since the generation of an appropriate initial population is crucial to the genetic algorithm, this paper designs an initial population generation algorithm based on random methods and avoids infeasible solutions to ensure population diversity. Its pseudocode is as follows.

The fitness function is the main index for evaluating and describing the quality of an individual population, and it is

Local search operator
1: Function (bandwidth local search)
2: An intermediate node and its subtree are randomly selected from the parental topology to find the leaf node connected by the link with the least available bandwidth.
3: **if** there is more available bandwidth between the node and the other intermediate nodes **then** the node is connected to that intermediate node.
4: **end if**
5: EndFunction
6: Function (traffic local search).
7: **if** the parental repair topology meets the $\text{degree}_{\text{newcomer}} > d - k + 1$**then** randomly select a node from the leaf node directly connected to the root node and connect it to a random intermediate node $(d - k + 1 = 4)$;
8: EndFunction
9: Function (local search of node processing capacity)
10: A node with stronger processing capacity is randomly selected from the leaf node of the subtree to replace the worst processing capacity intermediate node, the positions of these two nodes in the topology are exchanged.
11: EndFunction

Algorithm 4

Local search operator
1: Randomly select individuals from the current population to join the next-generation population;
2: **if** the population of the next generation reaches pop $*$ 0.9**then** go to step 5
3: **else** go to step 1
4: **end if**
5: Arrange the current population individuals in ascending order according to fitness value, and select the former pop $*$ 0.1 individuals to inherit directly to the next-generation population.

Algorithm 5

also an important reference factor for the genetic algorithm to obtain high-quality solutions. The objective of the algorithm optimization problem in this paper is to improve the repair efficiency of the online repair process of the failure nodes while reducing the repair flow as much as possible. Therefore, in combination with the above mathematical model and formula (8), the fitness function is formulated as follows:

$$f = c_{\text{time}} \max \left[\alpha \frac{D_i}{\text{process}_i} + \frac{\beta}{\omega(V_i, V_j)} + \alpha * \frac{D_j}{\text{process}_j} \middle| (V_i, V_j) \in E \right] + \frac{c_{\text{traffic}}}{\sum_{\text{mid}-1}^{n-d-1} \text{degree}_{V_{\text{mid}}} - 2}. \tag{9}$$

The smaller the individual fitness function value is, the better the tree repair topology performance is.

4.2. Crossover Operator. The crossover operator ensures the diversity of the population based on inheriting the parental gene fragment to improve the searching ability of the algorithm. For the two trees T_A and T_B under the three-layer topology in this paper, the crossover operation is shown in Figure 5. The scheme is as follows.

4.3. Mutation Operator. The mutation operator maintains the search range of the algorithm in the individual neighborhood of the population. To ensure that every solution in the variation parental neighborhood is reachable, two different variation operators are designed and executed with the same probability.

Mutation operator 1 makes it possible for every leaf node to be connected with any intermediate node or root node, while mutation operator 2 makes it possible for every leaf node to be an intermediate node. Here, in combination with Figure 6, the mutation operator scheme is as follows.

4.4. Local Search Operator. Due to the large-scale random search mechanism of the genetic algorithm, the convergence speed of the algorithm is too slow. The algorithm in this paper designed some operators to perform a local search for three targets in the fitness function: bottleneck bandwidth, topology saving traffic, and node processing capacity, as shown in Figure 7. These three operators are executed in turn, and the pseudocode is as follows.

4.5. Select Operator. This paper adopts the method of random selection combined with elite retention for selection. The random selection operator allows every individual in the current population to inherit the next generation through a random selection of the current population, maintains the diversity of the population, and avoids premature convergence to the local optimal algorithm. Elite retention selects some of the best individuals in the current population and passes them directly to the next generation. Although some population diversity is lost, elite retention can guarantee the convergence of the whole algorithm. The combined selection

The optional topology construction method based on a hybrid genetic algorithm
1: Function (parameter initialization)
2: Initialize the population size parameter pop, maximum evolutionary generation G, and cross variation probability P_C, P_M;
3: Function (population initialization)
4: Repeat Algorithm 1 pop times to get the initial population $P(0)$ with the number of pop. Calculate the fitness value of each individual according to formula (9) and set $t = 0$.
5: Function (crossover operation)
6: Randomly select two individuals from the current population $P(t)$
7: **if** probability $P_C < 0.5$**then** perform Algorithm 2 to generate crossover descendants
8: **else** parental individuals directly join the crossover descendants
9: **end if**
10: Repeat the crossover descendants until the number of crossover descendants is equal to the pop, and the individual set of all these crossed individuals is denoted as a O_1;
11: Function (mutation operation)
12: From a random individual in the current population $P(t)$,
13: **if** probability $P_M < 0.5$**then** Algorithm 3 is executed to generate mutation progeny
14: **else** parental individuals are directly added to cross progeny
15: **end if**
16: Repeat the mutation operation until the number of descendants of the mutation is equal pop, denoted as O_2 in the topology.
17: Function (local search)
18: Perform Algorithm 4 successively for each individual in set $P(t) \cup O_1 \cup O_2$, perform a local search, and get three equally large subpopulations of L_1, L_2, and L_3.
19: Function (select operation)
20: Perform Algorithm 5 and set $P(t) \cup O_1 \cup O_2 \cup L_1 \cup L_2 \cup L_3$. Select individuals of pop as the next-generation population $P(t+1)$, and let $t = t + 1$;
21: EndFunction
22: **if** the generations = G**then** the individual with the lowest fitness function value is the solution
23: **else** go to step 3
24: **end if**

Algorithm 6

operator inherits the advantages of the two selection methods so that the algorithm does not fall into local optima based on good convergence. Assuming that the population size is pop, the specific steps are as follows.

4.6. Optimal Topology Construction Method Based on a Hybrid Genetic Algorithm. Based on the operators designed above, the whole genetic algorithm proposed is as follows.

5. Experiment and Evaluation

5.1. Experimental Environment. We used PyCharm 2019.1.3 and Python 3.7 to code the genetic algorithm simulation program on an Intel® Core™ i7-9750H @ 2.60 GHz CPU, 16 GB of memory, and 512 GB SSD hardware to conduct the contrast experiment. The experiment simulated the performance of the storage nodes in the cluster topology parameter selection for the I/O, CPU, memory, chips; the weighting of the corresponding parameters of 40%, 30%, 20%, and 10%; and the corresponding values range for [21,265], [1, 90], [0.3,76], and [0.5,23] [31]. The heterogeneity of the available bandwidth of the links in the cluster is governed by the parameters in the literature. Its distribution is obtained in PlanetLab [32], which is shown in Table 1. In the experiment, the size of the original data M was 1 GB, and the crossover probability and mutation probability of the algorithm in this paper were set as 0.9 and 0.1, respectively, after repeated experiments.

Table 1: Bandwidth distribution of the virtual storage nodes (conversion results and statistics).

Bandwidth C (KB/s)	The percentage (%)
$C < 30$	2.7
$30 \le C < 200$	11.3
$200 \le C < 500$	26.2
$500 \le C < 1000$	56.1
$1000 \le C$	3.7

5.2. Repair Efficiency. The optimal repair topology construction method based on the MSR code and the genetic algorithm is proposed in this paper.

The repair delay of all links traversing the *newcomer* direct connection is sorted, and the traditional star repair topology of the optimal solution is selected. The prim algorithm based on the minimum spanning tree algorithm PrimMST is used to construct the traditional tree repair topology with four MSR codes of different sizes, $(10, 6, 8)$, $(20,12,16)$, $(40,24,32)$, and $(80,48,64)$, considering the storage cluster scenario with heterogeneous node processing performance and link available bandwidth. The optimal repair delays of the three topologies were compared. The results are shown in Figure 8.

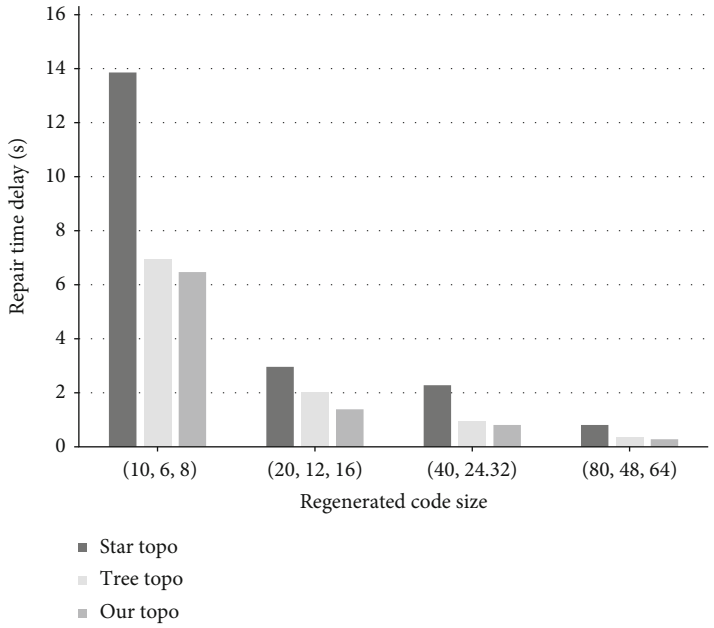


Figure 8: Optimal topology repair delay of each algorithm.

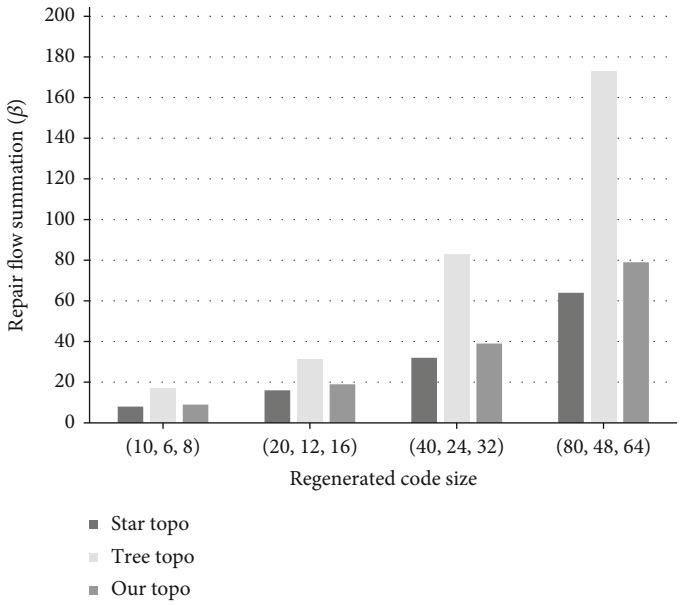


Figure 9: Optimal topology repair flow of each algorithm.

It can be seen that the star topology cannot avoid some links with poor available bandwidth in the selection of topology links, leading to a long repair delay. However, the PrimMST algorithm based on a greedy mechanism can no longer obtain the global optimal solution after the introduction of the processing capability of node heterogeneity. Under the same storage scale, the repair delay of the approximate global optimal topology constructed by the algorithm proposed in this paper is only 70%~90% and 35%~45% of the traditional tree and star repair topologies.

5.3. Repair Flow. As mentioned in Section 3.3 of this paper, the size of the repair traffic in the repair topology is also an

Table 2: Influence of four $(c_{\text{time}}, c_{\text{traffic}})$ values on MSR code repair delays of different sizes.

$(c_{\text{time}}, c_{\text{traffic}})$ Size of MSR	(0.25,0.75)	(0.5,0.5)	(0.75,0.25)	(1, 0)
(10, 6, 8)	6.47	6.47	6.47	6.47
(20,12,16)	1.39	1.39	1.39	1.39
(40,24,32)	0.82	0.82	*0.81*	*0.80*
(80,48,64)	*0.29*	*0.28*	*0.27*	*0.26*

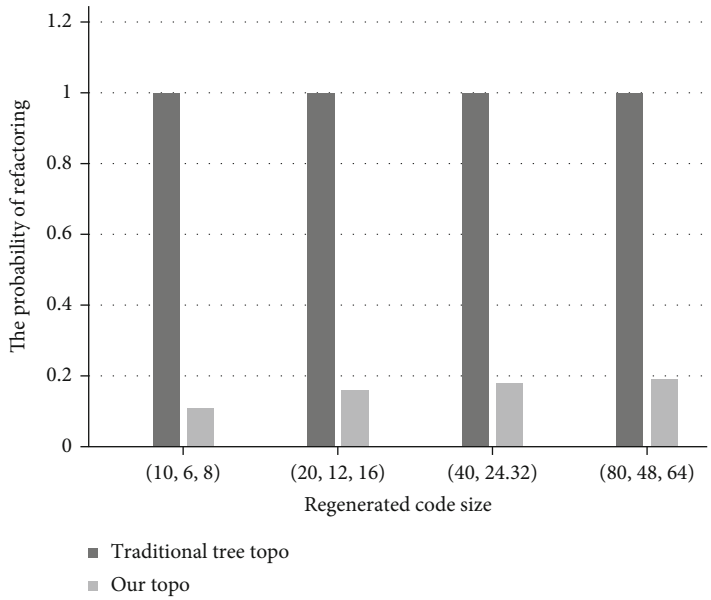


Figure 10: Probability of topology reconstruction of the single-node failure in the repair topology under four MSR codes.

important indicator for judging the merits of the repair topology. Taking the β-size traffic sent by each *provider* in the topology as a unit, we calculated the sum of the repair traffic passed on each link in each topology in the same experimental environment as in the previous section. The results are shown in Figure 9.

Although star topologies have the lowest repair traffic, the previous section has shown that their repair efficiency is too low for practical use. Combined with the experimental results in the previous section, it can be found that compared with the traditional tree repair topology construction method, the proposed method has a higher repair efficiency and the traffic in the topology is only 50% to 60% of that in the traditional tree topology. Although the extra path brought by the intermediate nodes increases the traffic overhead slightly compared with the star topology, the algorithm in this paper can use links with larger available bandwidth in the topology to transmit the repair traffic, so the repair efficiency is far higher than that of the star topology. Considering the two factors of repair efficiency and repair flow, the method in this paper surpasses the two traditional topological algorithms in terms of comprehensive performance, which is consistent with the original design goal.

5.4. Parameter Analysis. We adjust c_{time} and c_{traffic} to test and analyze the relationship between the repair rate and the repair flow in the topology during the overall repair process. The larger the c_{time} is, the greater the algorithm focuses on the repair rate, and the larger the value is, the less the repair delay of the algorithm construction topology is. When $c_{\text{time}} = 1$, the topology reduces the repair delay as much as possible at the cost of improving the repair traffic, and the topology repair delay is the shortest. However, the effect of adjusting the weight parameters on the whole topology structure can be almost ignored because of the small amount of traffic saved when the size of the MSR code is small. Table 2 shows how the repair rate of the topology constructed by the algorithm in this paper changes with the repair delay weight coefficient c_{time} and the repair flow weight coefficient c_{traffic} under different sizes of the regenerated code. When $(c_{\text{time}}, c_{\text{traffic}})$ is $(0.25,0.75)$ and $(0.5,0.5)$, the repair delays of the corresponding topologies in the scales of $(40,24,32)$, $(20,12,16)$, and $(10, 6, 8)$ do not change. When $(c_{\text{time}}, c_{\text{traffic}})$ is $(0.75,0.25)$ and $(1, 0)$, although the repair delays corresponding to the scales $(20,12,16)$ and $(10, 6, 8)$ remain unchanged, the repair delays of the topologies $(80,48,64)$ and $(40,24,32)$ are reduced to a certain extent, which is consistent with our analysis.

5.5. Robust Analysis of the Repair Topology. To demonstrate the robustness of the repair topology constructed by the proposed method, we also analyzed the probability that the repair topology in the MSR code storage cluster with scales of $(10, 6, 8)$, $(20,12,16)$, $(40,24,32)$, and $(80,48,64)$ is reconstructed when nodes in the topology fail during the repair process. The results of the proposed method and the traditional tree repair topology are compared. The results are shown in Figure 10.

Due to the nature of the MSR code itself, when a node loss occurs during repair, the repair traffic must be supplied from the nonprovider node, but the traditional tree topology only has two types of nodes, *provider* and *newcomer*; even with the spare nodes in the structural topology, the newly built repair topology does not have global optimality and there is no guarantee that a failed node can be repaired. Therefore, the reconstruction probability of the proposed method is always 100% regardless of the storage size. The intermediate nodes of the repair topology constructed by this method are also selected from n coding nodes, so they are qualified to be *providers*. When one of the *provider* nodes fails in the repair topology, it is only necessary to specify an intermediate node to make it a *provider*. Therefore, unless the intermediate node fails, there is no need to reconstruct

the repair topology, and the repair topology is more robust than the traditional tree repair topology.

6. Summary and Prospects

In the practical application of a distributed storage system, the heterogeneity of the processing capacity of each node and the influence of the repair traffic on the state of the cluster cannot be ignored in the process of repairing the data on the failed nodes. Aiming at reconstructing data in a single failure node scenario, this paper establishes a repair model and proposes an optimal repair topology construction method based on a genetic algorithm to simultaneously optimize the repair rate and flow of the repair topology. Experiments show that the proposed repair topology can improve the efficiency of node repair and can optimize the flow of repair topology, and the proposed topology is more robust than the traditional tree repair topology. The method proposed in this paper is only applicable to the repair scenario of single-node failure in a distributed storage system. The multimode scenario that considers optimization objectives is worth further research.

Data Availability

The heterogeneity of the available bandwidth of the links in the cluster is governed within the article (Ref. [27]). Its distribution is obtained in PlanetLab (Ref. [28]) which is shown in our paper (Table 1) to support the findings of our study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. 61662018 and 61861013), the Science and Technology Major Project of Guangxi (No. AA18118031), the Guangxi Natural Science Foundation of China (No. 2018GXNSFAA050028), the Doctoral Research Foundation of Guilin University of Electronic Science and Technology (No. UF19033Y), and the Director Fund Project of Key Laboratory of Cognitive Radio and Information Processing of Ministry of Education (No. CRKL190102).

References

[1] Wikipedia, "Distributed file system for cloud[EB/OL]," 2019, https://en.wikipedia.org/wiki/Distributed_file_system_for_cloud.

[2] N. Lakshmi, G. R. Bairavasundaram, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35no. 1, pp. 289–300, 2007.

[3] Y. Wang and S. Li, "Research and performance evaluation of data replication technology in distributed storage systems," *Computers & Mathematics with Applications*, vol. 51, no. 11, pp. 1625–1632, 2006.

[4] J. S. Plank, "Erasure codes for storage applications," in *Tutorial Slides Presented at FAST-2005: The 4th USENIX Conference on File and Storage Technologies*, San Francisco, USA, 2005.

[5] Z. Fengyan, W. Yan, and L. Nianshuang, "Survey of heterogeneous-based data repair strategies for erasure codes," *Application Research of Computers*, vol. 36, no. 8, pp. 2241–2249+2255, 2019.

[6] L. Rizzo, "On the feasibility of software FEC," Univ di Pisa, Italy, 1997.

[7] Z. Yao, C. Jiajia, and W. Chuliang, "Survey on data updating in erasure-coded storage systems," *Journal of Computer Research and Development*, vol. 57, no. 11, pp. 2419–2431, 2020.

[8] Q. Zheng, "Research on erasure code for secure storage system," Shanghai Jiao Tong University, Shanghai, 2009.

[9] X. Luo and S. Jiwu, "Summary of research for erasure code in storage system," *Journal of Computer Research and Development*, vol. 49, no. 1, pp. 1–11, 2012.

[10] S. Wang, "Study on fast repair of failed nodes in distributed storage systems," Chang'an University, 2019.

[11] X. Lin, Y. Wang, P. Xiaoqiang, F. Xu, and F. Yongquan, "GRC: a high fault-tolerance and low recovery-overhead erasure code for multiple losses," *Journal of Computer Research and Development*, vol. 51, Supplement 2, pp. 172–181, 2014.

[12] P. Teng, L. Chen, D. Yuan, and X. Wang, "Sparse random erasure code: a fault tolerance scheme for large-scale storage system," *Journal of Xi'an Jiaotong University*, vol. 51, no. 5, pp. 48–53, 2017.

[13] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Transactions on Information Theory*, vol. 12, no. 5, pp. 102–117, 2014.

[14] A. G. Dimakis, K. Ramchandran, Yunnan Wu, and Changho Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[15] S. Jiekak, A. M. Kermarrec, N. le Scouarnec, G. Straub, and A. van Kempen, "Regenerating codes," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 2, pp. 23–32, 2013.

[16] H. Jianzhong, *Design and Optimization of Erasure-Coded Clustered Storage Systems*, Science Press, Beijing, 2016.

[17] C. Jia, J. Wang, Y. Zhu et al., "On the optimal provider selection for repair in distributed storage system with network coding," *International Conference on Algorithms and Architectures for Parallel Processing*, 2015, pp. 506–520, Springer, Guangzhou, China, 2015.

[18] Q. Gong, J. Wang, D. Wei, J. Wang, and X. Wang, "Optimal node selection for data regeneration in heterogeneous distributed storage systems," in *I ICPP '15 Proceedings of the 2015 44th International Conference on Parallel Processing (ICPP)*, pp. 390–399, Washington DC, USA, 2015.

[19] X. Wan, "The research on optimization of topology sensitive repair technology in distributed storage system," Nanjing University, 2015.

[20] H. Zhang, H. Li, and S. Y. R. Li, "Repair tree: fast repair for single failure in erasure-coded distributed storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1728–1739, 2017.

[21] Y. Wu, R. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Allerton Conference on Control, Computing, and Communication*, IEEE Press, 2007.

[22] J. Li, S. Yang, X. Wang, and B. Li, "Tree-structured data regeneration in distributed storage systems with regenerating

codes," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, San Diego, CA, USA, March 2010.

[23] Q. Fenglin, G. Qingyuan, Z. Yangfan, and X. Wang, "Heterogeneity-aware node selection for data repair in distributed storage systems," *Journal of Computer Research and Development*, vol. 52, Supplement 2, pp. 68–74, 2015.

[24] H. Liu, Y. Wang, and N. Fan, "A hybrid deep grouping algorithm for large scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 6, pp. 1112–1124, 2020.

[25] X. Xue, "A compact firefly algorithm for matching biomedical ontologies," *Knowledge and Information Systems*, vol. 62, no. 7, pp. 2855–2871, 2020.

[26] C.-H. Chen, "A cell probe-based method for vehicle speed estimation," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E103-A, no. 1, pp. 265–267, 2020.

[27] Y. Wang, H. Liu, F. Wei, T. Zong, and X. Li, "Cooperative coevolution with formula-based variable grouping for large-scale global optimization," *Evolutionary Computation*, vol. 26, no. 4, pp. 569–596, 2018.

[28] X. Xue and J. Chen, "Using Compact Evolutionary Tabu Search algorithm for matching sensor ontologies," *Swarm and Evolutionary Computation*, vol. 48, pp. 25–30, 2019.

[29] C.-H. Chen, "An arrival time prediction method for bus system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4231-4232, 2018.

[30] G. R. Raidl and B. A. Julstrom, "Edge sets: an effective evolutionary coding of spanning trees," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 225–239, 2003.

[31] S. J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between PlanetLab nodes," *International Conference on Passive & Active Network Measurement*, 2005, pp. 292–305, Springer, Boston, USA, 2005.

[32] "Planet-Lab a platform for developing, deploying, and accessing planetary-scale services [EB/OL]," https://www.planet-lab.org/.