Received 25 October 2022, accepted 27 December 2022, date of publication 11 January 2023, date of current version 19 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3235787

RESEARCH ARTICLE

HV-SNSP: A Low-Overhead Data Recovery Method Based on Cross-Checking

YING SONG[1,2,3], TIANTONG MU[1,2], AND BO WANG[4]
[1]Beijing Key Laboratory of Internet Culture and Digital Dissemination, Beijing Information Science and Technology University, Beijing 100101, China
[2]Beijing Advanced Innovation Center for Materials Genome Engineering, Beijing Information Science and Technology University, Beijing 100101, China
[3]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100086, China
[4]Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou 450002, China

Corresponding author: Ying Song (songying@bistu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872043; and in part by the State Key Laboratory of Computer Architecture [Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS)] under Grant CARCHA202103.

ABSTRACT The failure of a single unreliable commodity component is very common in large-scale distributed storage systems. In order to ensure the reliability of data in large-scale distributed storage systems, many studies have emerged one after another. Among them, Erasure Codes are widely used in actual storage systems, such as Hadoop Distributed File System (HDFS), which provides high fault-tolerance with lower storage overhead. However, usually the recovery of erasure-coded storage system when encountering node failure results in severe cross-node and cross-rack bandwidth consumption, which affects the efficiency of failure recovery and wastes additional resources. In this paper, we improve the erasure coding storage strategy in distributed storage systems, and propose a low-overhead data recovery method based on cross-checking, namely HV-SNSP. In HV-SNSP, horizontal and vertical cross parity checking is realized by adding RS parity inside the data node, that is, $H^{RS(n,k)}$-$V^{RS(n',k')}$ storage architecture. Based on $H^{RS(n,k)}$-$V^{RS(n',k')}$, a low-cost supply node selection strategy, namely SNSP, is designed, and nodes with shorter network distance and lower load are selected to participate in recovery. This strategy can effectively reduce the amount of data transmission, shorten the recovery time, and improve the recovery efficiency. The experimental results show that compared with traditional RS, HV-SNSP can reduce the amount of cross-rack data transmission by 62.5% during data recovery, and can shorten the recovery time by up to 42.41%; Compared with D^3, HV-SNSP can reduce the occupation of cross-rack bandwidth by 25% and shorten the recovery time by 36.58%.

INDEX TERMS Data recovery, distributed storage system, erasure coding.

I. INTRODUCTION

Distributed storage systems are usually composed of many independent and unreliable commercial components, and component failures are common. In order to ensure the high reliability and availability of data in such a distributed storage system, two common methods are to use multiple copies and erasure codes to provide fault tolerance.

The multiple replicas method is to store data in a redundant way by making multiple copies of the stored data, such as Google File System (GFS) [1] and Hadoop Distributed File System (HDFS) [2]. By default, three copies of each data block are stored, to withstand a double component failure. The multi copy form is easy to deploy and recover from failures, but the storage overhead is too high, so it is not suitable for systems with large amount of data and small disk space.

As an alternative, erasure codes achieve the same fault tolerance as replication with much lower storage overhead, which has been deployed in some distributed systems. Reed-Solomon (RS) [3] coding is commonly used at present, a RS(n, k) code encodes n data blocks into k parity blocks, and all $n + k$ data/parity blocks form a strip. The loss of any block in the stripe can be recovered from the other n blocks, so RS can tolerate up to k data block failures. Erasure codes' fault-tolerant strategy will occupy less storage space than the multi-copy scheme. For example, RS code is deployed in HDFS-RAID [4], which reduces the storage redundancy from

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Gaggero.

the traditional 3x to 1.4x. However, using erasure codes, it is necessary to retrieve multiple available blocks to recover the failed blocks, which leads to high recovery costs. Although erasure codes improve storage efficiency, they significantly increase disk I/O and the occupation of network bandwidth for failure recovery.

The large-scale distributed storage system is deployed on multiple independent data nodes, which are located in multiple racks, that is, multiple data nodes are placed on each rack to store the original data and parity data of the erasure coding system, respectively. In order to maximize the data availability of distributed storage systems deployed using erasure codes, different erasure codes blocks are stored in nodes in different racks. This data layout allows the system to tolerate a certain number of node failures and rack failures. However, this data block placement method inevitably causes the repair of any faulty data block to retrieve available data blocks from other racks, which occupies a large amount of cross-rack bandwidth. In the cluster, the node that reads local data and then transmits it to other nodes through the network to participate in the recovery is called the supply node. The node that collects the required data from the supply node and calculates the lost data is called the newly born node. In a distributed storage system, the inner-rack bandwidth is considered sufficient but the cross-rack bandwidth is not. Generally, the available cross-rack bandwidth of each node is only 1/20 to 1/5 of the internal rack traffic [5]. Therefore, cross-rack bandwidth is generally regarded as a scarce resource [6], and excessive cross-rack bandwidth inevitably delays the recovery process.

When data blocks are obtained from each supply node, the state change of the node also have a certain impact on the recovery speed. In the RS default recovery process, all nodes are communicated in random order, and the surviving nodes are added to the success list so that the data blocks and check blocks can be obtained sequentially in the next step. If the status of the nodes added to the successful list suddenly changes during transmission and cannot respond in time, it is forced to end the current recovery procedure and reselect the nodes participating in the recovery, which wastes a lot of time. This situation is mostly due to too many tasks and heavy load in the supply node, and the data blocks cannot be transferred to the target node for recovery in time, which increases the recovery time and affects recovery efficiency.

In order to solve the above problems, we reconsider the recovery problem of single failure node in distributed systems and make improvements. The main contributions of this paper are as follows.

- HV-SNSP, a low-overhead data recovery method based on cross-checking, is proposed to reduce the amount of data transmission, shorten the recovery time, and improve the recovery efficiency.
- In HV-SNSP, the horizontal and vertical cross-checking is realized by adding RS parity inside the data nodes, that is, an improved erasure code storage layout $H^{RS(n,k)}$-$V^{RS(n',k')}$ [7], which is suitable for distributed systems with erasure coded storage, such as HDFS. $H^{RS(n,k)}$-$V^{RS(n',k')}$ is able to decode parity blocks from the node itself when the node fails, thus reducing the cross-rack traffic generated during recovery.
- Based on the $H^{RS(n,k)}$-$V^{RS(n',k')}$ storage architecture, a low-cost supply node selection strategy, namely SNSP, is designed. First, SNSP selects nodes with shorter distance, then selects nodes with lower load according to the load of each node, and selects the appropriate node as the supply nodes by combining the two factors of distance and load. It aims to improve the transmission rate of data blocks, thereby improving the efficiency of data recovery and the reliability of the system.
- By analyzing the experimental results, compared with the traditional RS erasure code storage, HV-SNSP can reduce the amount of cross-rack data transmission by up to 62.5%, and can shorten the recovery time by up to 42.41% when a single data block is lost. Compared with D^3, HV-SNSP can reduce the amount of cross-rack data transfer by up to 25% and can shorten the recovery time by 36.58%. Although the data transfer and disk space occupation within a part of the rack is increased, the reduction of the amount of cross-rack data transfer and the shortening of the recovery time are remarkable, and the recovery efficiency is improved.

The rest of this article is organized as follows. Section II introduces the related work on fault recovery of distributed systems. Section III introduces HV-SNSP in detail. Section IV analyzes the advantages and disadvantages of HV-SNSP through comparative experimental evaluation. Finally, Section V concludes the paper.

II. RELATED WORK

At present, there have been a lot of research achievements on the data recovery problem of distributed storage system. Before this, most recovery methods considered the frequency of data access, such as [8], [9], and [10], among which literature [10] proposed an initiative based on popularity in the HDFS-RAID system. PP tracks popular data and immediately restores lost popular data when Hadoop nodes fail. When these popular data are accessed, it takes no time to recover the data, which significantly improves the efficiency of foreground work. The data recovery time and the execution time of the foreground tasks are reduced, and the impact on other tasks in HDFS-RAID are minimized.

Recently, researches focus of distributed system data recovery has gradually changed from multi-copy recovery strategy to erasure codes-based data recovery. The existing data recovery work using erasure coding and data layout can be roughly divided into the following scheme: a combination of various erasure code methods, a change in the existing data layout in erasure codes, and a new recovery strategy in erasure codes.

A. RESEARCH ON DATA RECOVERY FOR PROPOSING NEW ERASURE CODE CODING SCHEME

Zhou [11] used the BASIC framework and ZigZag decodable code design to implement a new storage code, and built it into a STORE system. In the data strip layout, the upper triangular matrix is arranged, and the lower triangular part is reflected and filled along the diagonal. When there is data loss, one part could be recovered by diagonal copying, and the other part could be recovered by calculating the erasure code with the BMBR code. The evaluation on a 21-node HDFS cluster shows that the recovery bandwidth and disk I/O utilization are almost reduced to the minimum, and degraded read throughput is improved significantly. However, this layout leads to the repeated storage of some data blocks, which leads to a certain degree of redundancy. For single node failure, Li [12] constructed the CORE system, strengthened the existing optimized regeneration code, and supported single-node and concurrent failures. CORE proposed two coding symbols, which can recover the failed nodes by reconstructing virtual symbols and real symbols. CORE was implemented and evaluated on a Hadoop HDFS cluster test platform with up to 20 storage nodes. The results show that compared with the traditional erasure codes-based recovery method, the CORE prototype saves recovery bandwidth and improves recovery efficiency. Although the above recovery method can save bandwidth and improve the recovery efficiency of the distributed system, the complexity of calculation and derivation is greatly increased, and the occupation of storage space is also higher than that of traditional erasure codes.

B. RESEARCH ON DATA RECOVERY BASED ON THE COMBINATION OF MULTIPLE ERASURE CODES

At present, there are many kinds of erasure codes in use, such as RS [13], LRC [14], PC [15], MSR [16], [17], [18], [19], etc. Each erasure code has its own advantages and insufficient. In order to make up for these shortcomings, some scholars use a combination of multiple erasure codes (mostly a combination of two erasure codes) to improve the recovery efficiency of distributed storage systems. RS is one of the most common erasure codes, it satisfies the properties of MDS [20] and can achieve the highest fault tolerance, but the recovery read data volume is large. LRC reduces occupation of recovery bandwidth by reducing disk I/O, but sacrifices storage efficiency [21]. MSR is designed to reduce recovery cost with high storage efficiency, but its computation is too complicated. To solve these shortcomings, literature [22] proposed an erasure code (called AZ-Code) for multiple availability areas, which was a hybrid code that combines the advantages of MSR codes and LRC codes. AZ-Code used a specific MSR code as a local parity check, and used a typical RS code to generate a global parity check, to keep the recovery cost low and the reliability high. Literature [21] proposed to mix RS and LRC. Although the simultaneous application of these two codes in the same distributed system increased the complexity of encoding and decoding, the LRC-RS mixed code has obvious advantages in data downloading (recovery bandwidth). It not only reduced the occupation of recovery bandwidth in the network, but also brought considerable economic benefits. There were also some studies combined with other erasure codes. Literature [23] introduces a new erasure code storage system HACFS, which used two different erasure codes and dynamically adapts to the change of workload. It uses fast code to optimize recovery performance, and compresses code to reduce storage overhead. A novel conversion mechanism was used to efficiently upcode and downcode data blocks between fast codes and compact codes. Compared with the three popular single-code storage systems, the HACFS system always keeps the storage overhead low, significantly improves recovery performance, and effectively reduces the latency, recovery time and disk/network bandwidth of degraded read. However, the above solution increases the calculation amount of encoding and decoding, prolongs the recovery time, and the decision of switching between the two encoding modes also affects the recovery speed.

C. RESEARCH ON DATA RECOVERY BY CHANGING THE LAYOUT OF EXISTING ERASURE CODED DATA

For situations involving bandwidth occupancy between racks, Literature [24] defined Deterministic Data Distribution (D^3), placing multiple data blocks of the same strip on different nodes in the same rack to reduce the cross-rack traffic for single-node failure recovery, and proposed an efficient recovery method based on D^3 to achieve load balancing of recovery traffic under single node failure. Not only between the nodes in the racks, but also between the racks to balance the recovery flow. The research team at D^3 also proposed PDL [25], which was constructed based on pairwise balanced design, and the lost blocks are recovered by uniformly selecting replacement nodes and retrieving the determined available blocks, which effectively reduces the cross-rack traffic, and can provide nearly balanced cross-rack traffic distribution. If data nodes are distributed in a very large geographic area, that is, storage across data centers, the recovery performance of the code are affected by more factors, including network and computing-related factors. Pablo et al. [26] proposed a code based on XOR, which supplemented the idea of parity copy and rack awareness, and expanded it to MXOR (Multiple XOR), generated several vertical parity check blocks and horizontal parity check blocks for the data block, which could effectively deal with the failure of a node. Through the implementation and evaluation of the erasure code module in the XORBAS version of HDFS, the amount of data to be read during recovery was reduced and the recovery time was shortened. This method can effectively solve single-node failures and reduce cross-rack traffic during recovery by arranging data blocks, but it takes up a lot of storage space, takes a long time to find blocks during recovery, and the recovery performance across data centers needs to be improved.

D. RESEARCH ON DATA RECOVERY BASED ON A NEW ERASURE CODE RECOVERY STRATEGY

Zheng et al. [27] proposed a low-cost multi-node failure repair method oriented to erasure codes, using a serial repair method to complete the repair work of multiple failed nodes in turn, taking the network distance as the basis for node selection, they believed that nodes with shorter network distance have higher bandwidth occupation, and vice versa, have less bandwidth occupation. This method of selecting the supply node and the central node based on the network distance can improve the available bandwidth between the nodes and reduce the burden of measuring the available bandwidth between nodes for the system. In addition, the calculation and transmission of data were organized by the multi-threaded computing method and the pipeline data transmission method, and the data recovery method based on the central node was used to restore multiple failed data blocks at the same time, which greatly reduced the bandwidth overhead. Li et al. [28] proposed a new pipeline repair technique, which could be applied in both homogeneous and heterogeneous environments. In an isomorphic environment, the recovery time for pipeline recovery reaches O(1). In a heterogeneous environment, if the system adopts the classic Reed-Solomon code, its recovery time can reach O(k); the recovery time of the recently proposed PPR repair method can reach O(log k). In many cases, compared with traditional recovery methods and PPR recovery methods, the recovery time of pipeline recovery of a single block can be reduced by 80% to 90%, which improves the recovery performance of HDFS-based systems [29]. Most of the above new erasure code recovery strategies are based on network and pipeline transmission methods, but if the node network or load conditions change instantaneously, transmission are interrupted, and it takes a lot of time to resume transmission.

III. HV-SNSP METHOD DESIGN

HV-SNSP realizes the cross-checking of RS parity with different parameters inside the data node from the horizontal and vertical directions, that is, the $\mathrm{H}^{\mathrm{RS(n,\,k)}}$-$\mathrm{V}^{\mathrm{RS(n',k')}}$ storage architecture. This architecture can use vertical parity to decode within the node during data recovery, reducing the amount of data transfer across nodes and across racks. Based on the $\mathrm{H}^{\mathrm{RS(n,k)}}$-$\mathrm{V}^{\mathrm{RS(n',\,k')}}$ storage architecture, SNSP further optimizes the inevitable cross-node and cross-rack transmission, selects nodes with shorter network distance and lower load as supply nodes to participate in recovery, reduces the transmission time as much as possible, so as to improve the recovery efficiency. Next, we describe HV-SNSP in detail.

A. $H^{RS(n,k)}$-$V^{RS(n',k')}$ CODING LAYOUT METHOD

RS(n, k) coding is a commonly used coding method in current erasure code storage systems, and it realizes the so-called Maximum Distance Separable (MDS) characteristic. Given two positive integers n and k, a RS(n, k) code encodes n data blocks into k additional parity blocks, all $n + k$ data/parity blocks form a stripe, $n + k$ is called the strip size. The MDS property guarantees that any of the $n + k$ blocks can be recovered from the other n blocks. Thus, an RS(n, k) code can tolerate the failure of any k blocks in a strip without data loss.

The RS(n, k) coding method enables the system to tolerate a certain number of node failures and rack failures. However, the original data blocks placement inevitably causes the recovery of any faulty data block to retrieve available data blocks from other racks, which occupies a large amount of scarce cross-rack bandwidth resources and inevitably delays the recovery process. In order to reduce the cross-rack bandwidth occupation of the erasure code storage system during data recovery, and thus reduce the recovery delay, we make improvements on the basis of RS(n, k) encoding and propose a new encoding method, namely $\mathrm{H}^{\mathrm{RS(n,\,k)}}$-$\mathrm{V}^{\mathrm{RS(n',k')}}$. The basic idea of $\mathrm{H}^{\mathrm{RS(n,\,k)}}$-$\mathrm{V}^{\mathrm{RS(n',k')}}$ is to add the parity calculation inside the node (called vertical coding in this paper) on the basis of the traditional RS erasure code storage (called horizontal coding), also using RS coding, and store the parity calculation result in the current node. Among them, $\mathrm{H}^{\mathrm{RS(n,k)}}$ means that the horizontal direction adopts RS(n, k) coding, and $\mathrm{V}^{\mathrm{RS(n',k')}}$ means that the vertical direction adopts RS(n', k') coding. Although $\mathrm{H}^{\mathrm{RS(n,k)}}$-$\mathrm{V}^{\mathrm{RS(n',\,k')}}$ may cause data redundancy, in the event of a node failure, the parity block can be decoded from the node itself, thereby reducing the cross-rack traffic generated during recovery. Next, a specific example is given to introduce $\mathrm{H}^{\mathrm{RS(n,\,k)}}$-$\mathrm{V}^{\mathrm{RS(n',k')}}$.

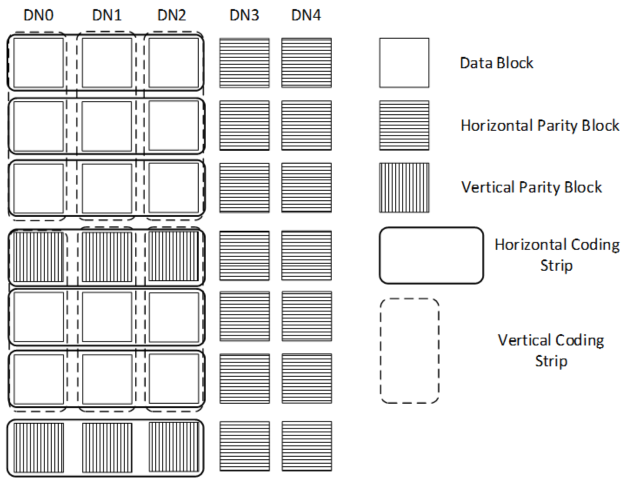

FIGURE 1. $\mathbf{H}^{RS(3,2)}$-$\mathbf{V}^{RS(3,1)}$ **storage architecture.**

Consider a distributed storage system, such as HDFS [2]. The system is composed of a collection of nodes, and each node stores many blocks. The nodes are divided into data nodes and parity nodes. The data node stores the data block and a little local parity blocks generated by encoding $\mathrm{V}^{\mathrm{RS(n',k')}}$ within the node, and the parity node stores the parity blocks generated by the cross-node RS encoding, as shown in Figure1. The n and k values of horizontal RS can be changed according to the number of nodes. Taking $H^{RS(3,\ 2)}$-$V^{RS(3,1)}$ as an example, namely, horizontal RS(3, 2) and vertical RS(3, 1), which use a total of five nodes. DN_0, DN_1, and DN_2 are data nodes, and DN_3 and DN_4 are parity check nodes. Three data blocks in each row are encoded to generate two parity check blocks and stored in the parity check node. These five blocks form a horizontal strip. If any two blocks in a row are lost, they can be reconstructed by $H^{RS(3,\ 2)}$. Inside the node, RS coding is also used. Different from horizontal, RS(3, 1) coding is used for vertical fixed, namely $V^{RS(3,1)}$. Every three data blocks are coded to generate one parity block and stored in this node. The above four blocks form a vertical stripe. In this way, when a small amount of data is lost in a node, $V^{RS(3,1)}$ can be used to decode and restore without using a parity node, which reduces the amount of cross-node and cross-rack data transmissions, thereby reducing the bandwidth occupation between racks.

B. NODE SELECTION STRATEGY BASED ON NETWORK DISTANCE AND LOAD BALANCING(SNSP)

When applying the RS-based erasure coding strategy for data recovery, the default data block selection algorithm uses sequential reading from the random node list, and then communicates with the nodes in turn. If the node survives, it will be used, and it will be added to the success list, and determine the nodes participating in the recovery in the order of the list. In the process of recovery, whether it is to recover a block, multiple blocks or even the whole node data block, the decoding computation time is negligible compared to the time of selecting the data node participating in the recovery and transmitting the data. Therefore, one of the more effective ways to shorten the recovery time is to reduce the transmission time, so as to avoid a large increase in the recovery time caused by the nodes being too busy and errors in transmission. At this time, the selection of blocks becomes particularly important. The coding layout method we propose in Section III-A increases the internal verification of nodes and reduces the amount of cross-node and cross-rack transmission of some data blocks, but there is still the problem of the node access delay mentioned above in the lateral verification. Therefore, we propose a node selection strategy based on network distance and load balancing for the $H^{RS(n,k)}$-$V^{RS(n',\ k')}$ architecture.

This strategy comprehensively considers the network distance between nodes and the current load of the node. The main purpose of considering the network distance is to use the data blocks in the same rack to participate in the recovery as much as possible, which can reduce a large amount of cross-rack data transmission. If the network distance is the same or similar, the node with lower load is selected to participate in the recovery. Such selection may avoid selecting the node with a large amount of tasks, so as to avoid a long response time and affecting the recovery efficiency. We still take $H^{RS(3,\ 2)}$-$V^{RS(3,1)}$ in Section III-A as an example, as shown in Figure 2. Assuming that the DN_0 node fails, all data blocks are lost. In such situation, the internal check block of the node cannot be recovered, so that the data blocks need to be transmitted from other nodes. Before the fault is detected and the data needs to be transmitted is determined, the nodes are sorted according to the network distance from other nodes of the same strip to the faulty node in ascending order, and the node distance list templist1 is obtained. The estimation method of the network distance is as follows, d represents data center, r represents racks, and n represents nodes:

Case I: the distance of the same node:
$\text{Distance}(/d_1/r_1/n_0, /d_1/r_1/n_0) = 0$
Case II: the distance of nodes on the same rack:
$\text{Distance}(/d_1/r_1/n_1, /d_1/r_1/n_2) = 2$
Case III: the distance of nodes on different racks in the same center:
$\text{Distance}(/d_1/r_1/n_1, /d_1/r_2/n_1) = 4$
Case IV: the distance of nodes in different center:
$\text{Distance}(/d_1/r_1/n_1, /d_2/r_1/n_1) = 6$

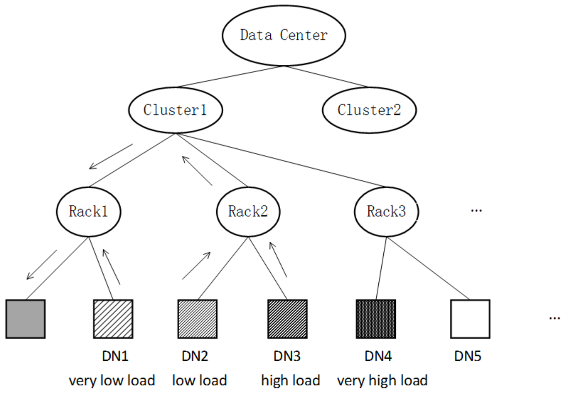

FIGURE 2. The principle of supply node selection strategy based on distance and load balancing.

Algorithm 1 Supply Node Selection Strategy Based on Distance and Load Balancing

```
   Input: List of nodes, list1
   Output: List of nodes that eventually participated in
   the recovery, list2
1  begin
2      Invalidate the specified DataNode
3      for i=0 to list1.size
4          Calculate network distance based on node location
5          Insert nodes into list templist1 in ascending
6      end
7      for j=0 to list1.size
8          Calculate the CPU usage of each node
9          Calculate the memory usage of each node
10         Calculate the disk usage of each node
11         Calculate the weight value
12         Insert nodes into list templist2 in descending order
13     end
14     Comparing templist1 and templist2 inserted into list2
15     return list2
16 end
```

Then, the load are calculated for other nodes in the same strip of the faulty node, respectively, according to the current

CPU usage abbreviated as cpu, memory usage abbreviated as *mem*, and disk usage abbreviated as disk of each node. Such three parameters are weighted according to the ratio of 0.5, 0.3, 0.2 [30] to calculate the weight, as shown in formula (1).

$$load = 0.5 * cpu + 0.3 * mem + 0.2 * disk \qquad (1)$$

Then, all the calculated loads are stored in the load list templist2 in ascending order. Synthesize the node order of templist1 and templist2 to obtain the final list of nodes participating in the recovery. The specific steps are shown in Algorithm III-B.

IV. EXPERIMENT

In order to verify the HV-SNSP method, we create five virtual machines with the CentOS system in the server, build a Hadoop cluster, and conducts comparative experiments according to the amount of data and the number of missing data. In our experiments, the data amounts are 10G, 50G, 100G, and 1T, and the amount of lost data is one data block, multiple data blocks, and all data blocks in a single data node. The comparison objects are traditional RS coding and D^3. We verify the amount of cross-node, cross-rack data transmission and the failure recovery time in the above scenarios. The comparative analysis results are listed in Section IV-B.

A. OVERVIEW OF EXPERIMENTAL DESIGN

We use one server with ten-core 2.40 GHz Intel(R) Xeon(R) Gold 5115 CPU, 256G memory, Dell PERC H730P integrated RAID hard disk, total storage space 3.3T, running CentOS 7.x. We configure KVM on this server and create five new virtual machines, each with 8G memory and running CentOS 7.x. Three of them are used to build Hadoop 3.3.1 cluster, all with 560G storage space, and one virtual machine runs as NameNode and DataNode at the same time, and the other two virtual machines only run as DataNode; The other two are not in the cluster, and only store parity blocks as parity nodes, each with 380G storage. In the actual system, the available cross-rack bandwidth of each node is only 1/20 to 1/5 of the internal rack bandwidth, and we regard the environment where the five virtual machines are located as the same rack in the real environment, so convert the speed difference to time delay processing when simulating a scene across racks.

In the above environment, we implement the encoding and decoding operations of HV-SNSP, RS(3, 2) and D^3 in Java respectively. For consistency of comparison, the same encoding parameters of RS(3, 2) are used in HV-SNSP and D^3, which are described as HV-SNSP(3, 2) and $D^3(3, 2)$ in the following. RS(3, 2) and $D^3(3, 2)$ are used as comparison terms and integrated into the recovery system. The master node arranges the overall code of the recovery program, and the slave node only arranges the jar package encoded by $V^{RS(3,1)}$, which is convenient for operation. The amount of test data is 10G, 50G, 100G, and 1T, and there are three test scenarios: five nodes are in the same rack, some nodes are in the same rack, and none of the five nodes are in the same rack. Each data block stored in Hadoop is 128MB by default. In order to evaluate the recovery performance, we randomly delete one data block, multiple data blocks and all data blocks of a single data node, and then measure the amount of data transmission during recovery and the total recovery time. In order to avoid the randomness of the results, we repeat each group of experiments three times, and take the average of the three results as the result.

B. EXPERIMENTAL RESULTS AND ANALYSIS

We conduct extensive simulation tests to evaluate the performance of the recovery system. The difference of data transmission amount and recovery time when recovering lost data blocks is analyzed between HV-SNSP(3, 2) and RS(3, 2) as well as HV-SNSP(3, 2) and $D^3(3, 2)$ in three different scenarios. Each scenario is further divided into three cases: one data block is lost, multiple data blocks are lost, and all the data blocks in a single node are lost.

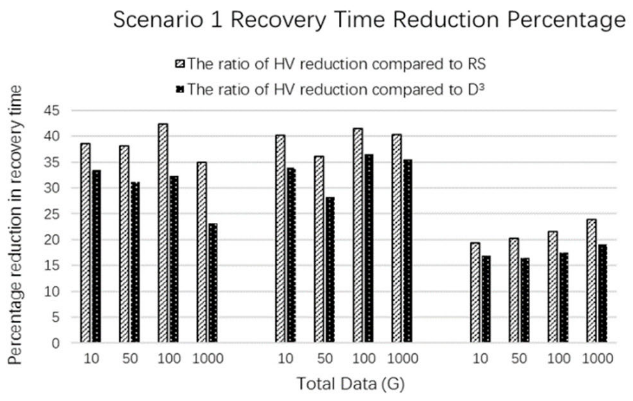

FIGURE 3. Scenario 1 Recovery time reduction ratio diagram.

1) SCENARIO 1: EACH NODE IS ON A DIFFERENT RACK

In scenario 1, when one or n data blocks are lost, using RS(3,2) or D^3 to recover the lost data requires accessing at least $3n$ blocks, and then transferring them across racks to the nodes that need to be recovered for computational recovery. While HV-SNSP(3, 2) can be recovered by vertical RS(3,1) decoding on the same node, which only needs decoding calculation and does not need transmission. As shown in Figure 3, the recovery time of HV-SNSP is greatly shortened compared with RS(3,2) and D^3, and the loss of a block can be reduced by more than 35% and 23% respectively. The lost blocks can be reduced by 36% and 28% respectively. In the case of the whole node failure, because each node is on a different rack, every transmission is cross-rack transmission. However, HV-SNSP reduces the cross-rack transmission by 25%, and selects the node with lower load as the supply node to improve the recovery efficiency, and reduces the recovery time by more than 20% and 16% compared with RS(3,2) and D^3.

2) SCENARIO 2: EACH RACK HAS MULTIPLE NODES (TWO ARE USED AS AN EXAMPLE IN THE EXPERIMENT)

In the second scenario, if one or *n* data blocks are lost, HV-SNSP still does not need to transmit data blocks, and it can be recovered directly by decoding the internal check block of the nodes. RS(3,2) requires at least 2*n* cross-rack transmissions, D^3 first transmits the data in the same rack to other nodes in the same rack, then calculates the linear combination of the data, and then transmits the linear combination result to a new node across racks, that is, the recovery needs to be transmitted across racks for *n* times, and the selection of supply nodes is random, so the recovery speed is greatly affected. As shown in Figure 4, HV-SNSP can shorten the recovery time by up to 40% and 34%. For the case that all the data of the nodes are lost, the recovery process of RS(3,2) and D^3 is similar to that of multiple data block failures, which requires a lot of data transmission across nodes and racks. Compared with RS(3,2), the data transmission of HV-SNSP(3,2) increases by 25% across nodes and decreases by 62.5% across racks; Compared with D^3, the data transmission of HV-SNSP(3,2) across nodes and racks is reduced by 25%. The recovery time is about 20% shorter than RS(3,2), and with the increase of data, the advantage is more obvious.

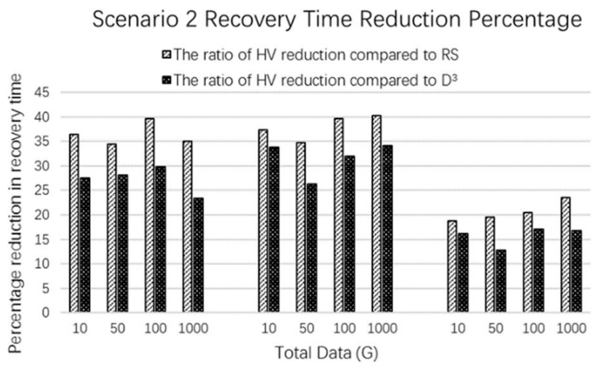

FIGURE 4. **Scenario 2 recovery time reduction ratio diagram.**

3) SCENARIO 3: ALL NODES ARE ON THE SAME RACK

In the recovery process of scenario 3, there is no cross-rack data transmission. When one or more data blocks are lost, it is similar to the first two scenarios, as shown in Figure 5, the recovery time of HV-SNSP(3, 2) is shortened by about 35% and 27% compared with RS(3, 2) and D^3, and the effect is lower than that of the first two scenarios. When recovering the whole node, HV-SNSP reduces the amount of cross-node data transmission by 25% compared with the other two methods, so the recovery time is shortened, and it increases with the increase of data, up to 23% and 16%, which greatly shortens the recovery time and improves the recovery efficiency.

In conclusion, in the three scenarios mentioned above, compared with RS(3, 2) and D^3, HV-SNSP(3, 2) reduces the amount of data transmission across nodes and racks, thus effectively shortening the recovery time. As can be seen from Figure 3-5, due to the addition of the checksum supply node selection strategy inside the node, when the data block is lost, compared with RS(3, 2), it can reduce a lot of data transmission across nodes and racks, thus greatly shortening the recovery time. And with the increase of data volume, it basically shows an upward trend. Compared with D^3, the improvement effect of some test results does not strictly comply with the rising rule. The reason is that the load balancing policy of D^3 allocates some data blocks and parity blocks to nodes with lower load, which exactly reduces the transmission amount during recovery. Similarly, when other commonly used erasure codes (n, k) parameters are used, such as (6,3) and (10,4), HV-SNSP(n, k) can also play a similar role. When recovering the whole node, with the data volume increasing from 10G to 1T, the reduction ratio of recovery time gradually increases, showing an obvious upward trend; With the same amount of data, as shown in Figure. 6, taking 1T as an example, the recovery time reduction ratio is Scenario 1 > Scenario 2 > Scenario 3, that is, the more racks the nodes are distributed, the more effective HV-SNSP is. Therefore, due to the addition of internal node check and the selection of supply nodes with short network distance and low load to participate in recovery, HV-SNSP has better recovery efficiency, and is suitable for scenarios with large amount of data and more cross-rack transmission.

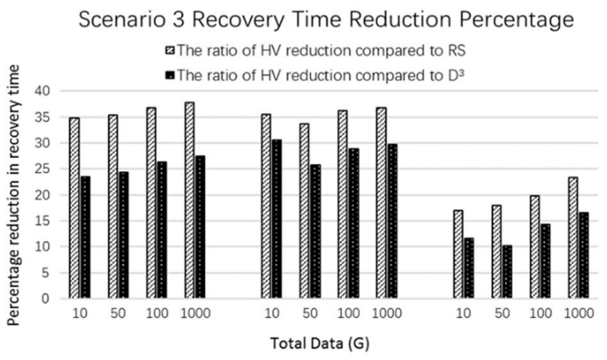

FIGURE 5. **Scenario 3 Recovery time reduction ratio diagram.**

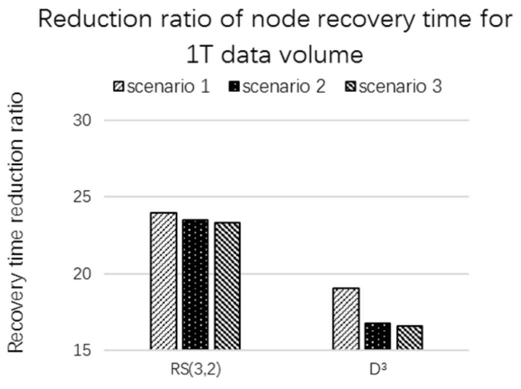

FIGURE 6. **1T data volume reduction ratio of the node recovery time.**

V. CONCLUSION

Aiming at the reliability problem of erasure-correction code system, this paper proposes HV-SNSP, a low overhead data recovery method based on cross check, which includes $H^{RS(n,k)}$-$V^{RS(n',k')}$ storage architecture and SNSP provider node selection strategy, which can significantly reduce the amount of cross-rack data transmission and the data recovery time. $H^{RS(n,k)}$-$V^{RS(n',k')}$ architecture adds the parity calculation inside the node on the basis of the traditional RS erasures code storage, uses RS codes with different parameters, and stores the parity calculation results in the current node. In this way, although some data redundancy is caused, parity check blocks can be decoded from the node itself in case of node failure, thus reducing the amount of cross-rack data transmission during recovery. Based on $H^{RS(n,k)}$-$V^{RS(n',k')}$ architecture, SNSP combines the two factors of network distance and load to select the nodes participating in the recovery, so as to avoid the large extension of the recovery time caused by the long response time of nodes. It can be seen from the experimental results that our method is more suitable for scenes with more nodes distributed in racks, more cross-rack transmission, and a large amount of data. Compared with the traditional RS erasure code storage, the amount of cross-rack data transmission of RS can be reduced by up to 62.5% when data is recovered, and the recovery time can be shortened by up to 42.41% when a single data block is lost. Compared with D^3, HV-SNSP can reduce the occupation of cross-rack bandwidth by up to 25% and shorten the recovery time by 36.58%. Although some intra-rack data transfer and disk space usage are increased, the inter-rack data transfer and the recovery time is greatly shortened, and the recovery efficiency of a single node is improved.

To further reduce the overhead of data recovery, HV-SNSP can be improved in the following three aspects in the future. 1) By adjusting the execution order of the calculation, unnecessary repeated calculation can be avoided, and the calculation amount and calculation cost can be reduced. 2) The same block of data can be used for the recovery of multiple blocks of data by rationally selecting the strip used for recovery. 3) By adjusting the recovery order of data blocks or stripes, files can be restored as soon as possible.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th ACM Symp. Operating Syst. Princ.*, New York, NY, USA, Oct. 2003, pp. 29–43, doi: 10.1145/945445.945450.

[2] K. Shvachko, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol.*, May 2010, pp. 1–10.

[3] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[4] Facebook. (2011). *HDFS-RAID*. [Online]. Available: https://wiki.apache.org/hadoop/HDFS-RAID

[5] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, 2010, pp. 267–280.

[6] R. Li, Y. Hu, and P. P. C. Lee, "Enabling efficient and reliable transition from replication to erasure coding for clustered file systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 9, pp. 2500–2513, Sep. 2017.

[7] T. Mu, Y. Song, M. Yang, B. Wang, and J. Zhao, "H-V: An improved coding layout based on erasure coded storage system," in *Database Systems for Advanced Applications, DASFAA International Workshops*, vol. 13248. Cham, Switzerland: Springer, 2022, pp. 203–213.

[8] S. Liu and D. Duan, "An improved method for HDFS replica recovery: Based on SVM algorithm," in *Proc. 4th Int. Conf. Cloud Big Data Comput.*, New York, NY, USA, Aug. 2020, pp. 76–80.

[9] W. Zhao and X. Cui, "A fast adaptive replica recovery algorithm based on access frequency and environment awareness," in *Proc. 4th Int. Conf. Cloud Big Data Comput.*, Aug. 2020, pp. 102–107.

[10] S. Wu, W. Zhu, B. Mao, and K.-C. Li, "PP: Popularity-based proactive data recovery for HDFS RAID systems," *Future Gener. Comput. Syst.*, vol. 86, pp. 1146–1153, Sep. 2018.

[11] T. Zhou, H. Li, B. Zhu, Y. Zhang, H. Hou, and J. Chen, "STORE: Data recovery with approximate minimum network bandwidth and disk I/O in distributed storage systems," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2014, pp. 33–38.

[12] R. Li, J. Lin, and P. P. C. Lee, "CORE: Augmenting regenerating-coding-based recovery for single and concurrent failures in distributed storage systems," in *Proc. IEEE 29th Symp. Mass Storage Syst. Technol. (MSST)*, May 2013, pp. 1–6.

[13] H. Dau, I. Duursma, H. M. Kiah, and O. Milenkovic, "Repairing Reed–Solomon codes with multiple erasures," *IEEE Trans. Inf. Theory*, vol. 64, no. 10, pp. 6567–6582, Oct. 2018.

[14] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 1814–1818.

[15] Y. Tang, F. Wang, and Y. Xie, "An efficient failure reconstruction based on in-network computing for erasure-coded storage systems," *J. Comput. Res. Develop.*, vol. 56, no. 4, pp. 767–778, 2017.

[16] Y. Hu, X. Zhang, P. P. C. Lee, and P. Zhou, "Generalized optimal storage scaling via network coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 956–960.

[17] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandic, "Opening the chrysalis: On the real repair performance of MSR codes," in *Proc. 14th USENIX Conf. File Storage Technol.*, 2016, pp. 81–94.

[18] M. Vajha, "Clay codes: Moulding MDS codes to yield an MSR code," in *Proc. 16th USENIX Conf. File Storage Technol.*, 2018, pp. 139–154.

[19] Y. Min and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[20] M. Ye and A. Barg, "Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1202–1206.

[21] H. Jin-Ping, L. Gui-Yang, and L. Hui, "Hybrid coding LRC-RS in heterogeneous decentralized storage," *Comput. Eng. Des.*, vol. 42, no. 2, pp. 301–308, 2021.

[22] X. Xie, C. Wu, J. Gu, H. Qiu, J. Li, M. Guo, X. He, Y. Dong, and Y. Zhao, "AZ-code: An efficient availability zone level erasure code to provide high fault tolerance in cloud storage systems," in *Proc. 35th Symp. Mass Storage Syst. Technol. (MSST)*, May 2019, pp. 230–243.

[23] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *Proc. 13th USENIX Conf. File Storage Technol.*, 2015, pp. 213–226.

[24] L. Xu, M. Lyu, Z. Li, Y. Li, and Y. Xu, "Deterministic data distribution for efficient recovery in erasure-coded storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2248–2262, Oct. 2020.

[25] L. Xu, M. Lv, Z. Li, C. Li, and Y. Xu, "PDL: A data layout towards fast failure recovery for erasure-coded distributed storage systems," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 736–745.

[26] P. I. S. Caneleo, L. J. Mohan, U. Parampalli, and A. Harwood, "On improving recovery performance in erasure code based geo-diverse storage clusters," in *Proc. 12th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Mar. 2016, pp. 123–129.

[27] L. Zheng and L. I. Xiaodong, "Low-cost multi-node failure repair method for erasure codes," *Comput. Eng.*, vol. 43, no. 7, pp. 110–118, 2017.

[28] R. Li, "Repair pipelining for erasure-coded storage," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2017, pp. 567–579.

[29] F. Zhong, "Survey of heterogeneous-based data repair strategies for erasure codes," *Appl. Res. Comput.*, vol. 36, no. 8, pp. 2241–2249, 2019.

[30] N. Wang, Y. Yang, Z. Mi, Q. Ji, and K. Meng, "A fault-tolerant strategy of redeploying the lost replicas in cloud," in *Proc. IEEE 8th Int. Symp. Service Oriented Syst. Eng.*, Apr. 2014, pp. 370–375.

YING SONG received the Ph.D. degree in computer engineering from the Institute of Computing Technology (ICT), Chinese Academy of Sciences. She is currently an Associate Professor with the Computer School, Beijing Information Science and Technology University. Her work has covered topics, such as performance modeling, resource management, cloud computing, and big data computing platform. She has been authored or coauthored more than 30 publications in these areas, since 2007. Her research interests include computer architecture, parallel and distributed computing, and virtualization technology. She served in various academic conferences.

TIANTONG MU received the B.S. degree in network engineering from Beijing Information Science and Technology University, Beijing, China, in 2019, where she is currently pursuing the master's degree with the Computer School. Her research interest includes distributed storage.

BO WANG received the B.S. degree in computer science from Northeast Forest University (NEFU), Harbin, China, in 2010, and the Ph.D. degree in computer science from Xi'an Jiaotong University (XJTU), Xi'an, China, in 2017. He was a Guest Student with the State Key Laboratory of Computer Architecture, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), from 2012 to 2016. He is currently a Lecturer with the Software Engineering College, Zhengzhou University of Light Industry (ZZULI). He has published more than ten research articles in these areas. His research interests include distributed systems, cloud computing, edge computing, resource management, and task scheduling. He served in various academic journals and conferences.