IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 December 2023, accepted 21 December 2023, date of publication 25 December 2023,
date of current version 11 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3346881

RESEARCH ARTICLE

ACPR: Adaptive Classification Predictive Repair Method for Different Fault Scenarios

YING SONG[1,2,3], PEISEN ZHENG[1,2], YINGAI TIAN[1], AND BO WANG[4]

[1]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 (ZZULI), 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; in part by the State Key Laboratory of Computer Architecture [Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS)] under Grant CARCHA202103; and in part by the Key Scientific and Technological Projects of Henan Province under Grant 232102211084.

ABSTRACT Erasure codes are widely used in large-scale distributed storage systems due to their high efficiency and reliability, but they also face extremely high repair penalties when data corruption occurs. At present, machine learning methods can accurately predict the next failure time and type of machine nodes. Based on this, in order to solve the problem of unnecessary repair traffic caused by temporary failures, as well as the more degraded reads of high-frequency accessed data due to longer failure time of such data in existing repair methods, we propose an Adaptive Classification Predictive Repair method (ACPR) for different fault scenarios. By categorizing the failed blocks into high-risk and low-risk based on the failure type of the soon-to-fail (STF) node and the access heat of STF blocks, ACPR can perform adaptive predictive repair. By quickly repair high-risk blocks to ensure data availability while delaying the repair of low-risk blocks, a large amount of unnecessary repair traffic caused by temporary node failures in the cluster is avoided. Alibaba Cloud Elastic Compute Service (ECS) experiments results show that compared with FastPR and ECPipe, ACPR can shorten the repair time per data block by up to 15.2% and 33.5%, respectively. Moreover, ACPR can reduce repair traffic by up to 74.1% and 84.4%, respectively.

INDEX TERMS Distributed storage system, data recovery, erasure coding.

I. INTRODUCTION

With the explosive growth of data, the availability and reliability of distributed storage system have always been the focus of research. The distributed storage system uses data redundancy technologies such as replication and erasure codes to provide fault tolerance so as to ensure the reliability and availability of the cluster [1]. Replication is a simple data redundancy storage method, but it will bring massive storage redundancy [2], [3]. Erasure codes generate limited redundant data through calculation, which provides lower redundancy at the same fault tolerance level compared with replication [4], [5]. Reed Solomon (RS) code is the most widely used erasure code. In RS (k, m), k data blocks are encoded into m parity blocks by calculation, and $k + m$ blocks are placed on different machine nodes, which is called horizontal stripe. Other blocks can be reconstructed from any k available blocks on the stripe [6], so as to tolerate the loss of m blocks.

Although erasure codes achieve efficient storage, they face high repair penalties. Repairing a single block requires reading k available blocks, it means that the bandwidth and I/O costs of such repair have been magnified by k times. In order to alleviate the high repair cost of erasure code, the existing related research can be divided into passive repair and predictive repair according to whether the fault actually occurred during data repair. Passive repair is to quickly repair damaged data blocks by optimizing repair technology [7], [8], [9], [10], after discovering data block damage. If the damaged data is accessed, degraded data reading may occur, resulting in the increased response time. The predictive repair is to process the possible damaged or lost data blocks in advance before the actual failure based on the prediction of the soon-to-fail

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

(STF) node through machine learning or other methods [11], [12], [13], [14], [15], so as to shorten the unavailable time of data blocks and improve data reliability [16], [17], [18], [19], [20], [21].

However, the above researches focus on how to optimize the repair process and ignore whether it is necessary to immediately repair faults. In storage systems, temporary failures of machines are more common than permanent failures [22], and the nodes that have temporarily failed will automatically return to the cluster after repair. Therefore, storage systems usually wait for a period of time, e.g. 15 minutes [9] or 30 minutes [23], before starting repair to avoid unnecessary repair from temporary failures. However, there are still some temporary failures that exceed the fixed waiting time, repairing such failures will result in a large amount of unnecessary repair traffic. There are also studies [24] that enhance the reliability of remaining available nodes while delaying the repair of all data blocks to avoid unnecessary repair caused by temporary failures. However, they ignore the problem that delaying the repair of frequently accessed hot data will reduce the availability of data in the storage systems [25]. Recent research has shown that in distributed clusters with prediction mechanisms, the next failure of machine nodes and the type of failure can be accurately predicted, including Immediate-Reboot (IR), Slow-Reboot (SR), and Forcible-Decommission (FD) [26]. In order to solve the problem of unnecessary repair traffic caused by temporary failures, as well as the more degraded reads of high-frequency accessed data due to longer failure time of such data in existing repair methods. We propose an Adaptive Classification Predictive Repair method (ACPR) for different fault scenarios. By categorizing the failed blocks into high-risk and low-risk based on the failure type of the STF node and the access heat of STF blocks, ACPR can perform adaptive predictive repair. The main goal is to reduce unnecessary repair traffic and shorten repair time.

The main contributions of this paper are as follows:

- An adaptive classification predictive repair method ACPR for different fault scenarios is proposed. According to the failure type of STF node and the access heat of STF block, ACPR can perform adaptive classification predictive repair.
- In ACPR, in order to solve the problem of the more degraded reads of high-frequency accessed data due to longer failure time of such data, a repair method by coupling repair pipelining reconstruction and migration is proposed to minimize the data repair time.
- In ACPR, in order to solve the problem of a large amount of unnecessary repair traffic caused by temporary node failures, it is proposed to quickly repair high-risk blocks by dividing the predicted STF blocks in STF nodes according to the type of failure and access heat, while delaying the repair of low-risk blocks. Such method can ensure data availability as well as avoid the generation of a large amount of unnecessary repair traffic.
- In order to evaluate ACPR, we deployed it on Hadoop Distributed File System (HDFS) and conducted several groups of experiments on 9 instances on Alibaba Cloud ECS. The experimental results show that compared with Fast Proactive Repair (FastPR) and Repair Pipelining for Erasure-coded Storage (ECPipe), ACPR can shorten the repair time per data block by up to 15.2% and 33.5%, respectively. Moreover, ACPR can shorten the total repair time compared to FastPR and ECPipe by up to 83% and 86.2%, respectively, and can reduce repair traffic by up to 74.1% and 84.4%, respectively. In the event of node failure during the repair process, ACPR can shorten the data repair time by up to 41.9% compared to FastPR. After the completion of predictive repair for high-risk blocks, ACPR reduces the average unavailable time of data blocks by up to 89.1% compared to ECPipe.
- We also conducted theoretical analysis on ACPR in larger scale environments to demonstrate its effectiveness. By analyzing and calculating the results in larger scale environments, ACPR can shorten data block repair time by up to 69.8% and 25%, respectively, compared to FastPR and ECPipe, and can reduce repair traffic by up to 78.4% and 84.2%, respectively.

The rest of this paper is arranged as follows. Section II introduces the related work. Section III introduces the research methods of ACPR are elaborated in detail. Section IV analyzes the advantages and disadvantages of ACPR through experiments and theoretical analysis. Finally, Section V concludes the paper.

II. RELATED WORK

Data repair in distributed storage systems can be divided into passive repair and predictive repair methods based on whether a fault actually occurred during the data repair operation; According to the delay of data repair initiation after identifying faults, it can be divided into two types of methods, i.e., immediate repair and delayed repair.

The passive repair method is reactive, that is, the repair operation is only triggered after detecting a node failure, and its repair operation needs to be performed through degraded reads. In order to improve the speed of degraded reads, Partial-Parallel-Repair (PPR) [8] improves the utilization of network bandwidth by splitting the repair process of a single block into multiple sub processes that can be executed in parallel. Repair Pipelining for Erasure-coded Storage (ECPipe) [7] uses repair pipelining reconstruction to further improve parallelism and reduce the repair time of a single failed block. Ultimately, the repair time of each block can be reduced to a time similar to the normal reading of a block. Although ECPipe can significantly reduce the repair time of each block, it cannot reduce the high repair penalties caused by erasure codes, that is, repairing each block still requires transferring k times the size of the block, which consumes a large amount of network resources. Research [26] improves the reliability

of storage systems by accelerating the identification of data blocks with high risk of loss and prioritizing the repair of high-risk data with different fault monitoring times. If the number of data blocks lost exceeds the fault tolerance of system, passive data repair methods can cause permanent data loss.

The predictive repair method performs predictive repair before actual faults occur, so it can further improve the reliability of data compared to passive repair methods. Some studies [16] and [19] predict the storage of data in STF disks in multiple copies, improving system reliability and availability by increasing replica redundancy. However, the redundancy of multiple copies can significantly increase storage overhead. Other studies predict repair or transfer in advance before a node fails [28]. Fast Proactive Repair (FastPR) [17] carefully couples the migration and reconstruction of STF node blocks, scheduling in a parallel manner, and making more full use of available bandwidth resources in the cluster. Although FastPR predictive repair STF nodes by coupling migration and reconstruction methods, significantly reducing the high repair penalties of erasure codes, its reconstruction method does not fully utilize the bandwidth resources between nodes, and there is still room for improvement in repair time.

Most of the existing methods for passive and predictive repair also belong to the immediate repair method. When a fault or STF node is detected, the lost or about to be lost data can be quickly repaired to ensure the availability and reliability of the system. However, temporary faults in the storage system account for the majority. Even though the storage system usually sets a data repair start delay [22], there are still some temporary faults that exceed the set waiting range, causing a large amount of unnecessary repair traffic.

The method of delayed repair is more concerned with whether immediate repair is needed than how to optimize repair. Therefore, the method of delayed repair generally delays the repair of failed data by enhancing the reliability of some stripes. For example, Risk-Aware Failure Identification (RAFI) [29] and Lazy Repair with Temporary Redundancy (LRTR) [24], where RAFI enhances reliability by accelerating the repair of stripes with more failed blocks and delaying the repair of stripes with fewer failed blocks, reducing unnecessary repair traffic. LRTR performs low-cost temporary redundancy operations on the surviving blocks of the affected stripes and delays the repair of all failed data, reducing unnecessary repair traffic. However, neither RAFI nor LRTR considers the issue of data access heat. If the repair of frequently accessed data is delayed, it will reduce the reliability of the storage system. There are also studies that combine predictive repair technology with delayed repair, such as Lazy Fast Predictive Repair (LFPR) [18]. By delaying the repair of parity blocks, the parity blocks are repaired together with data blocks in the same stripe, reducing the repair time of each block. However, it still cannot avoid a large amount of unnecessary repair traffic issues.

Algorithm 1 The Main Strategy of ACPR

```
Input: List of nodes, list;
begin
 while true do
  predict the failure and failure type on the nodes of list
  STF nodes → STFlist //Insert STF nodes into STFlist
 for each node ∈ STFlist do
  //Immediate-Reboot, Slow-Reboot, Forcible-Decommission are sets of
  failure types
  if STF node ∈ Immediate-Reboot then
   riskIdentification() // Insert block into corresponding risk list
  end if
  if STF node ∈ Slow-Reboot then
   riskIdentification() // Insert block into corresponding risk list
  end if
  if STF node ∈ Forcible-Decommission then
   each block → HighRiskList // Insert each block into HighRiskList
    end if
  end for
  for each block ∈ HighRiskList do
   highRiskBlockRepair() // Execute ACPR repair method
  end for
  for each block ∈ LowRiskList do
   lowRiskBlockDynamicDetect() // Lazy repair low risk block
 end for
end while
end
```

In order to quickly repair blocks to ensure data availability while reducing a large amount of unnecessary repair traffic caused by temporary node failures. Our idea is to take advantage of both predictive repair and lazy repair to categorize the failed blocks, maximizing the performance of repairing important data while minimizing unnecessary repair traffic.

III. ACPR DESIGN

Based on the above analysis, we designed the ACPR method to address the problem of unnecessary repair traffic caused by temporary failures, as well as the more degraded reads of high-frequency accessed data due to longer failure time of such data in existing repair methods. The main idea of ACPR is to categorize the failed blocks into high-risk and low-risk based on the failure type of the STF node and the access heat of STF blocks, ACPR can perform adaptive predictive repair. For high-risk blocks, in order to fully utilize the bandwidth resources between nodes and minimize the repair time, ACPR couples the migration and repair pipeline reconstruction to achieve rapid predictive repair. For low-risk blocks, in order to further avoid a large amount of unnecessary repair traffic, ACPR dynamically detects the number of failed blocks in the affected stripe, while delaying the repair of low-risk blocks until the number of failed blocks reaches the upper limit of stripe fault tolerance. The specific steps are given in Algorithm 1. We will provide a more detailed description in the three aspects, i.e., risk identification, high-risk block repair, and low-risk block dynamic detection as follows.

TABLE 1. Risk levels of STF blocks in three scenarios.

	Hot data block	Hot parity block	Cold STF block
IR	high	low	low
SR	high	low	low
FD	high	high	high

A. RISK IDENTIFICATION

When the STF node is detected, the blocks in the STF node are classified for risk identification. And the access heat of the data blocks stored on the STF node can affect the risk of whether to degrade reads after the data block fails. Therefore, data blocks with different heat levels may adopt different repair strategies. To address this, ACPR proposes risk classification methods for hot data blocks, hot data parity blocks, and cold blocks under three fault scenarios, i.e., Immediate-Reboot (IR), Slow-Reboot (SR), and Forcible-Decommission (FD). This is explained in detail as follows and summarized in Table 1.

1) HOT DATA BLOCK

As to the frequently accessed hot data block, the normal access to such data block is very important, so once the hot data block fails, the risk of degraded read in the three failure scenarios illustrated above is very high, and it needs to be repaired as soon as possible.

2) HOT PARITY BLOCK

The failure of the hot parity block will not affect the normal reading of other data, and in the IR and SR scenarios, the failed node will automatically return to the cluster for normal operation within a period of time, so the risk of degraded read of the failed hot data parity block is low in these two scenarios. In order to reduce unnecessary repair traffic, the strategy of delayed repair is adopted. However, in the FD scenario, the hot data parity block will permanently fail, so in this scenario, the risk of irreparable blocks is high and fast repair is necessary.

3) COLD STF BLOCK

Due to the low access frequency of cold data block, in the IR and SR scenarios, the failed node will automatically return to the cluster for normal operation within a period of time, so the risk of degraded read of the failed cold data block is low in these two scenarios, and the strategy of delayed repair is also adopted. However, in the FD scenario, cold data blocks will permanently fail, so in this scenario, the risk of irreparable blocks is high and fast repair is necessary.

B. HIGH-RISK BLOCK REPAIR

For high-risk blocks, ACPR needs to repair them efficiently as soon as possible. Therefore, ACPR aims to minimize the repair time for each round, such that it minimizes the overall repair time. When blocks retrieved from healthy nodes in each round are distributed across different nodes, the reconstruction can be executed in parallel. However, conventional reconstruction methods fail to fully utilize the bandwidth resources between nodes. To further shorten repair time, ACPR adopts a pipeline approach for reconstruction. Additionally, as data migration operations from STF nodes can be executed in parallel with the repair pipelining reconstruction to further reduce the repair time, ACPR adopts a parallel approach of data migration and pipeline reconstruction for data repair, which requires determining which data blocks need to be repaired through data migration or repair pipelining reconstruction. Therefore, ACPR needs to divide all data blocks to be repaired into two sets, namely, reconstruction set and migration set, and performs repair pipelining reconstruction and migration respectively. As long as the reconstructed set is determined, the remaining data blocks belong to the corresponding migration set, thus the primary goal is to determine the reconstructed set.

The idea of constructing the reconstruction set is to first construct the initial reconstruction set, then optimize and adjust it, and move some stripes from the reconstruction set to the migration set to determine the final reconstruction set.

1) CONSTRUCTING THE INITIAL RECONSTRUCTION SET

First, the initial reconstruction set is constructed, and each one of all high-risk blocks in the STF node is added to one subset of the reconstruction set according to the following subset division principle. ACPR divides G or less than G stripes into a subset in the reconstruction set. As to each stripe in the same subset, all k health blocks scattered across k different nodes. And all blocks in each subset can retrieve k blocks from k surviving nodes for reconstruction and repair at the same time. Where G is the maximum number of stripes that can be reconstructed in parallel, and $G \leq (N-X)/k$. That is, each node can concurrently retrieve at most one block from $N-X$ surviving nodes, where N is the number of cluster nodes and X is the number of STF nodes. In this way, at most $G \leq (N-X)/k$ stripes can be reconstructed in parallel each time, so each subset of the reconstruction set can have at most G stripes. If the parallelism of the current subset is less than G (the number of stripes reconstructed in parallel is less than G), it means that the current maximum subset has not been reached. It is necessary to optimize the current subset through the replacement strategy to make its parallelism as close as possible to G until all high-risk blocks in the STF node are divided into the reconstruction set.

Figure 1 shows an example of repairing high-risk blocks to build a reconstruction set. The storage system uses RS (3,2) coding. After the $N1$ node temporarily fails, only the blocks in the stripes $S1, S2, S3$ and $S4$ on the node $N1$ are determined to be high-risk blocks, so it is only necessary to build a reconstruction set on the $S1$ to $S4$ stripes. Since the stripes with k health blocks scattered across k different nodes are first divided into a subset of the reconstruction set, $S1$ and $S2$ are first divided into a subset $R = \{S1, S2\}$ in this example. However, at this time, the subset is smaller than the maximum

parallelism G =3, so the subset needs to be optimized, but $S3$ and $S4$ cannot join the subset at this time. Therefore, the reconstructed set $R = \{\{S1, S2\}, \{S3\}, \{S4\}\}$, and it requires three rounds of repair. However, try to replace $S2$ with $S3$. After replacement, $S4$ can also join the subset. At this time, $R = \{S1, S3, S4\}$ is equal to the maximum parallelism G, and the remaining $S2$ is divided into another subset. Therefore, the final reconstruction set is $R = \{\{S1, S3, S4\}, \{S2\}\}$, and it only requires two rounds of repair.

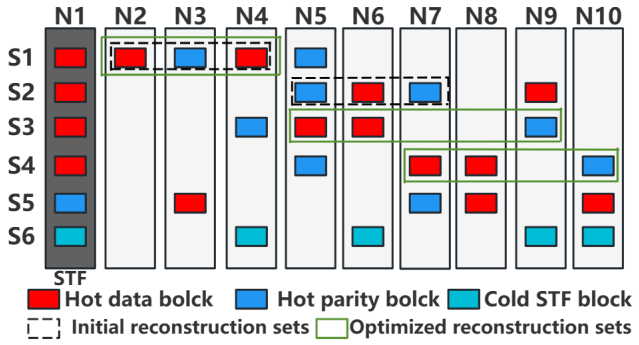

FIGURE 1. Example of constructing and optimizing reconstruction sets.

2) ADJUSTING THE RECONSTRUCTION SET

The goal of ACPR is repairing more blocks in each round, thus minimizing the total number of repair rounds and consequently reducing repair time. We conduct migration repair in parallel with each round of repair pipelining reconstruction. This entails converting a portion of the blocks slated for repair from using reconstruction to migration repair. Hence, we need to make adjustments to the reconstruction set, removing stripes from certain subsets of the reconstruction set and adding them to the migration set, to determine the final reconstruction set and migration set. The idea for adjusting the reconstruction set is as follows. ACPR sorts the subsets in descending order based on the number of data blocks to be repaired contained in each subset of the initial reconstruction set. It allows larger subsets to undergo parallel repair pipelining reconstruction, and moving stripes from smaller subsets to the migration set for repair through migration to reduce repair time.

How many stripes from the smaller subsets should be moved to the migration set? In order to answer this question, firstly, we need to calculate the time T_{recon} required for each round of high-risk blocks repaired through pipelining reconstruction and the time T_{migra} to repair a high-risk block through migration to determine the number of blocks repaired through migration during each round of repair. The calculation methods for T_{recon} and T_{migra} are shown in (1) and (2), respectively. Where s refers to the number of small data packets that the entire block is divided into for transmission, c refers to the block size, b_n refers to network bandwidth, and b_d refers to disk bandwidth. Due to the fact that in practice, computation operations can be executed in parallel with disk I/O and network transmission, and require less time, we ignore the overhead generated by computation here. Due to the fact that the number of packets s is generally much greater than $k-1$, the calculation of T_{recon} and T_{migra} shows that during each round of repair pipelining reconstruction, one high-risk block can be migrated and repaired in parallel. Therefore, each round of repair will select one subset from the subset arranged in descending order of the reconstruction set for repair pipelining reconstruction. At the same time, one stripe will be selected from the subset at the end of the reconstruction set and moved to the migration set for migration repair.

$$T_{recon} = \frac{c}{b_d} + \left(1 + \frac{k-1}{s}\right) * \frac{c}{b_n} + \frac{c}{b_d} \tag{1}$$

$$T_{migra} = \frac{c}{b_d} + \frac{c}{b_n} + \frac{c}{b_d} \tag{2}$$

Figure 2 shows an example of adjusting of the reconstruction set, which is $R = \{\{S1, S2, S3\}, \{S4, S5\}, S6\}\}$. And suppose that in this example, the time of transmitting a block $t = \frac{c}{b}$, $\frac{k-1}{s} = 0.1t$. Since the reconstruction set is arranged in descending order, the first subset $\{S1, S2, S3\}$ in R is reconstructed by repair pipelining, which takes $1.1t$. At the same time, we extract stripes from the end terminal set of R and add them to migration set M, that is, $S6$ is added to M, and the migration repair time is $1t(< 1.1t)$. Since $0.1\ t$ is not enough to migrate and repair a whole block, the first round of migration set M only has $S6$. The first round of repair is completed, and the reconstruction set $\{R\} = \{\{S4, S5\}\}$. Since there is only one subset left in the reconstruction set at this time, the second round only needs to use repair pipelining reconstruction subsets $\{S4, S5\}$.

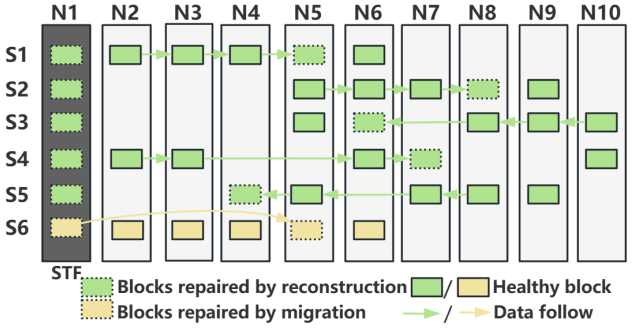

FIGURE 2. Example of adjusting the reconstruction sets.

Because there may be errors in the state prediction of nodes, it is inevitable to suddenly break down in the process of predictively repairing high-risk blocks. The migration method fails due to node failure, but the repair pipelining reconstruction method can still be used normally. At this time, the unrepaired high-risk blocks can only be repaired by repair pipelining reconstruction. Therefore, in this scenario, the ACPR repair method will degenerate into ECPipe.

C. LOW-RISK BLOCK DYNAMIC DETECTION

As to the low-risk blocks, their likelihood of being accessed is low, the risk of data irreparable is relatively low, and their hosted nodes may only temporarily fail and will automatically return to the cluster to work after a period of time. Thus, ACPR adopts a delayed repair strategy for such low-risk blocks to minimize the unnecessary repair traffic caused by temporary failures. When a node fails, ACPR will detect the fault tolerance of all low-risk block related stripes on that node. If the number of failed blocks in all related stripes has not yet reached the maximum number of failed blocks m in the stripe, ACPR will not handle it. If the number of failed blocks on any related stripe reaches m, ACPR will repair all failed blocks on that stripe, including hot data parity blocks and cold data blocks. Due to the fact that single node failure is the most common failure in distributed storage systems, accounting for 98% of the total [22], and the failure of low-risk blocks is temporary, the probability of repairing low-risk blocks is very low.

IV. PERFORMANCE EVALUATION

We create 9 cloud servers on Alibaba Cloud Elastic Compute Service (ECS), set up Hadoop clusters, and evaluate the performance of ACPR for RS codes on Hadoop Distributed File System (HDFS) (Section IV-A). Due to the limitations in experimental conditions, we also conduct theoretical analysis on ACPR in larger scale environments to demonstrate its effectiveness (Section IV-B). To evaluate the repair performance of ACPR, we compared it with FastPR and ECPipe in experimental and theoretical analysis.

A. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we evaluate ACPR for RS codes on Alibaba Cloud Hadoop clusters. First, we introduce the experimental design. Then, we analyze the performance of ACPR by four sets of experiments.

1) OVERVIEW OF EXPERIMENTAL DESIGN

We create 9 cloud server ECSs on Alibaba Cloud, built a Hadoop 3.1.1 cluster, and evaluate ACPR based on the RS code on HDFS. The instance is located in North China 2 (Beijing), and each instance has two 2.5GHz Intel Xeon Platinum vCPUs, 16GB of RAM, and 100GB of ESSD cloud disk storage, running CentOS 7.9. The disk bandwidth and network bandwidth of each instance are 128MB/s and 1 Gb/s, respectively. The ACPR master node and HDFS NameNode are deployed in one instance, while the ACPR slave nodes and HDFS DataNodes run in the remaining 8 instances as storage nodes.

We generate a 90GB file, which was is RS (3,2) encoded and written to HDFS, with randomly distributed stripes in the storage system. For consistent testing, we use the following default configuration: the erasure code parameter is RS (3,2), the data block size is 64MB, and the data packet size is 32KB. In each experiment, repair 60 blocks from STF nodes. To avoid randomness of the results, each group of experiments was repeated 10 times, and the average of the 10 results was taken as the final result.

To evaluate the performance of ACPR in different block numbers, block sizes, and node failures during predictive repair processes, and to evaluate the impact of the strategy of ACPR delaying low-risk block repair on data availability. In the above Alibaba Cloud ECS environment, we conduct four sets of experiments to analyze the impact of the number of repaired data blocks on repair time, the impact of block size on repair time and repair traffic, the impact of node failures on repair time during predictive repair process, and the impact of ACPR delayed repair of low-risk blocks on data availability.

2) EXPERIMENT 1: ANALYSIS OF THE IMPACT OF THE NUMBER OF REPAIRED DATA BLOCKS ON REPAIR TIME

In order to explore the impact of the number of repaired data blocks on repair time in our experiments, we first evaluate the average repair time of each block under different block numbers using ACPR, FastPR and ECPipe. As depicted in Figure 3, it can be seen that the repair time of each data block using the same method did not change significantly with the number of repaired data blocks. Therefore, for the convenience of subsequent evaluation, we use 60 as the default number of repaired data blocks.

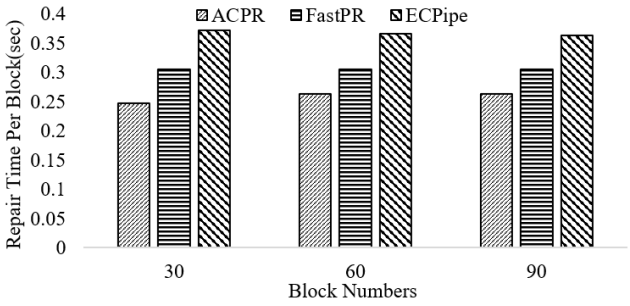

FIGURE 3. Experiment 1: Analysis of the impact of the number of repaired data blocks on repair time.

3) EXPERIMENT 2: ANALYSIS OF THE IMPACT OF BLOCK SIZE ON REPAIR TIME AND REPAIR TRAFFIC

We evaluate the impact of different block sizes on average repair time and average repair traffic per node, considering that ACPR only needs to repair high-risk blocks within STF nodes, while FastPR and ECPipe need to repair all blocks within nodes. Therefore, for better evaluation, we add the experiment of "ACPR-Repair all" (ACPR which repairs all blocks within nodes) to compare with FastPR, ECPipe and ACPR.

As shown in Figure 4, the repair time of each data block increases with the increase of data block size, but "ACPR-Repair all" still reduces the repair time of each block compared to FastPR and ECPipe by 13.3% - 14.1% and 29.6% - 33.5%, respectively. And ACPR also reduces the repair time of each block compared to FastPR and ECPipe

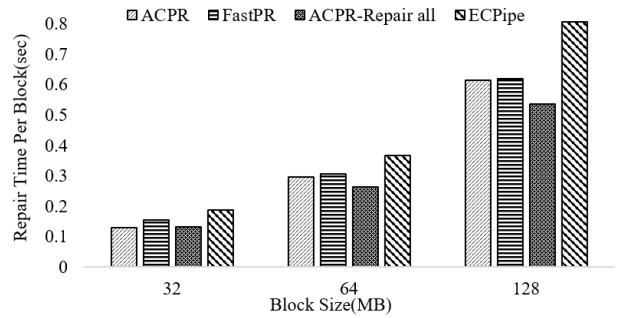

FIGURE 4. Experiment 2: Analysis of the impact of block size on per block repair time.

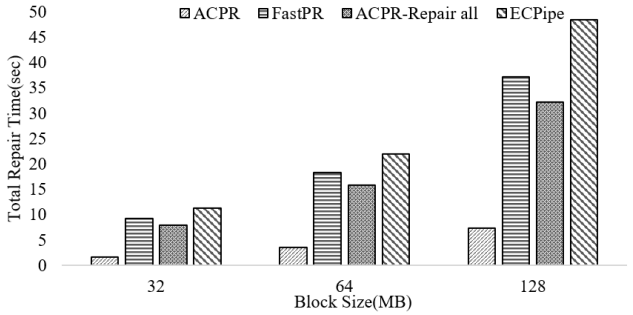

FIGURE 5. Experiment 2: Analysis of the impact of block size on total repair time.

by 0.6% - 15.2% and 23.7% - 30.8%, respectively, in all evaluated data block sizes. Additionally, as shown in Figure 5, ACPR reduces the total repair time compared to FastPR and ECPipe by 80.1% - 83% and 83.9% - 86.2%, respectively. And "ACPR-Repair all" also reduces the total repair time compared to FastPR and ECPipe by 13.3% - 14.1% and 28.2% - 33.5%, respectively, in all evaluated data block sizes. Since ACPR only needs to repair high-risk blocks, the number of blocks that ACPR needs to repair is much less than FastPR and ECPipe, so ACPR demonstrates a greater advantage in overall repair time. Therefore, it can be concluded that ACPR outperforms FastPR and ECPipe in terms of repair time per block and total repair time, whether repairing all blocks or only high-risk blocks.

Figure 6 shows that due to ACPR using the pipeline reconstruction method to repair data blocks, compared to FastPR, more data blocks will be repaired through reconstruction. Therefore, when all data blocks within the repair node are repaired, "ACPR-Repair all" generates 22.9% additional repair traffic, but still reduces 22.2% repair traffic compared to ECPipe. However, ACPR only needs to repair high-risk blocks within the node to ensure data availability, so in this case, ACPR will reduce repair traffic by 74.1% and 84.4% compared to FastPR and ECPipe, respectively.

4) EXPERIMENT 3: ANALYSIS OF THE IMPACT OF NODE FAILURES ON REPAIR TIME DURING PREDICTIVE REPAIR PROCESS

In real business scenarios, there may be STF node failures during the predictive repair process, resulting in incomplete data repair and degradation of repair methods. Such situation

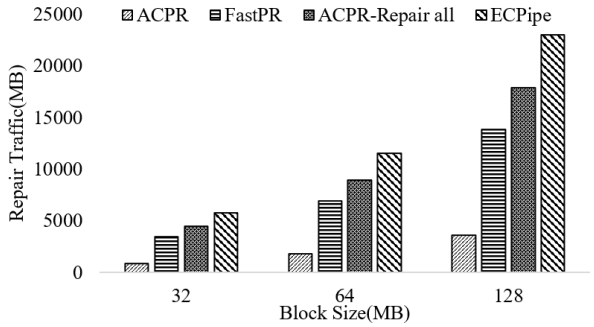

FIGURE 6. Experiment 2: Analysis of the impact of block size on repair traffic.

may lead to poor repair performance. Therefore, in order to evaluate ACPR in the situation of the node failure during the predictive repair, we compare ACPR with FastPR in such situation. As ECPipe belongs to passive repair, no comparison need to be made in this experiment. When an STF node fails, the method of migrating data blocks fails, and the ACPR method of repairing data blocks will degenerate into ECPipe, which only repairs the rest data blocks through pipeline reconstruction. FastPR does not consider this situation, so it degenerates into a normal reconstruction method. We compare ACPR with FastPR in the situations that the STF node fails when the repair completed 25%, 50%, and 75%, respectively. In this experiment, the size of each data block is 64MB and the number of repaired data blocks is 60. As shown in Figure 7, despite the failure of the migration repair method, ACPR still reduced the repair time by 31.8% - 41.9% compared to FastPR in all test cases. Moreover, the earlier the STF node failed during the predictive repair process, the more obvious the advantage of ACPR. Because in the event of a failed predictive repair by ACPR, the remaining data blocks can still be repaired through pipeline reconstruction, while FastPR can only repair the remaining data blocks through ordinary reconstruction.

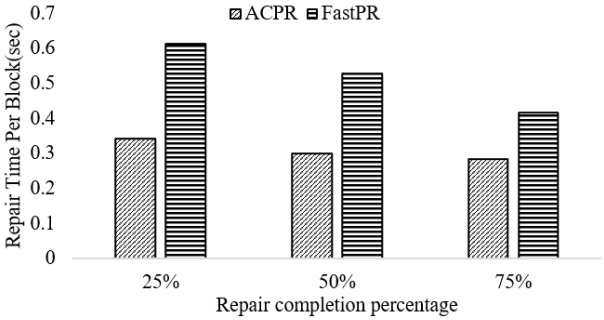

FIGURE 7. Experiment 3: Analysis of the impact of node failures on repair time during predictive repair process.

5) EXPERIMENT 4: ANALYSIS OF THE IMPACT OF ACPR DELAYED REPAIR OF LOW-RISK BLOCKS ON DATA AVAILABILITY

To evaluate the impact of delay low-risk block repair strategy adopted by ACPR on the availability of all data block, firstly, we completed the predictive repair of high-risk blocks

with ACPR. We assume that the STF node failed after the predictive repair. Then we test the average unavailable time of accessing all data blocks within the STF node. When the low-risk blocks are accessed, pipeline reconstruction is required to repair such data blocks, so the unavailable time of low-risk blocks is the average repair time of such data blocks in this scenario. Considering that FastPR predictively repairs all data blocks of STF nodes, we assume that the unavailable time of the data blocks after FastPR predictive repair is 0. ECPipe belongs to the passive repair method, so the unavailable time of the data blocks is the repair time of the data blocks. From Figure 8, it can be seen that the average unavailable time of data blocks increases with the increase of data block size. However, due to ACPR predictively repairing frequently accessed hot data blocks, the average unavailable time of all data blocks using ACPR is controlled within 0.09 seconds and reduced by 88.7% -89.1% compared to using ECPipe. Therefore, even if ACPR delays the repair of low-risk blocks, it does not have a significant impact on the data availability.

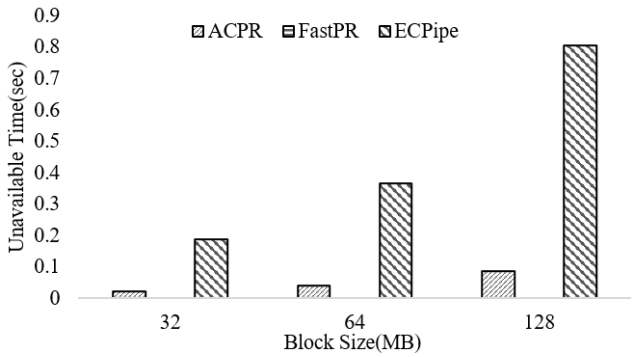

FIGURE 8. Experiment 4: Analysis of the impact of ACPR delayed repair of low-risk blocks on data availability.

B. THEORETICAL ANALYSIS OF ACPR

Due to the limitations in experimental conditions, we cannot evaluate ACPR in larger cluster sizes. Therefore, in this section, we use theoretical analysis to evaluate ACPR in larger scale environments to demonstrate its effectiveness.

1) OVERVIEW OF THEORETICAL ANALYSIS DESIGN

We use the following default settings, the total number of nodes $N = 100$, the number of blocks in the STF node $U = 1000$, the data block size $c = 64$MB, the network transmission bandwidth $b_n = 128$MB/s, disk transmission bandwidth $b_d = 1$Gb/s, the number of packets $s = 2048$, and use RS (6,3) as the default code, that is, $k = 6$, $m = 3$. The proportion of IR, SR, and FD nodes is configured to be the same as Google Trace, accounting for 67%, 32%, and 1% respectively. According to Zipf's law [30], the ratio of cold data block to hot data block (including hot data parity block) is set to 80% and 20%.

To evaluate the repair performance of ACPR under different RS encoding and different number of nodes in the cluster. We conduct theoretical analyses for ACPR in two aspects, i.e., the impact of different RS codes and the number of nodes in the cluster on repair time and the impact of different RS codes and the number of nodes in the cluster on repair traffic. And we verify the correctness of theoretical analysis methods through experiments on Alibaba Cloud.

2) ANALYSIS OF THE REPAIR TIME

Since ACPR only repairs high-risk blocks immediately, in order to better analyze the time spent on repair, all blocks in the STF node are set as high-risk blocks in this scenario. We assume that x blocks in the STF node are repaired by migration, and the number of blocks reconstructed by repair pipelining is $U - x$, so the total time spent on migration is $x*T_{migra}$, and the total time spent on repair pipelining reconstruction is $\frac{U-x}{G}*T_{recon}$. Because the operations of repair pipelining reconstruction and migration are parallel, the total repair time t predicted for repair is as follows.

$$T = max(x * T_{migra}, \frac{U - x}{G} * T_{recon}) \tag{3}$$

If the total repair time T is minimized, it is necessary to make $x*T_{migra} = \frac{U-x}{G}*T_{recon}$, that is, $x = \frac{U*T_{recon}}{G*T_{migra}+T_{recon}}$, so the minimum predicted repair time T_{min} is as follows.

$$T_{min} = \frac{U * T_{recon} * T_{migra}}{G * T_{migra} + T_{recon}} \tag{4}$$

We can calculate the ideal average repair time of data blocks for ACPR in the predictive repair process without node failures, as indicated by (4). To verify the feasibility of the theoretical analysis method, we first evaluated the average repair time of data blocks under the same parameters as the practical experiment. As shown in Figure 9, under different data block sizes, ACPR shortened the repair time by 27.2% and 33.3% compared to FastPR and ECPipe, respectively. This result is also close to practical EXPERIMENT 2 in Section IV-A, therefore the performance of ACPR can still be evaluated by the theoretical analysis.

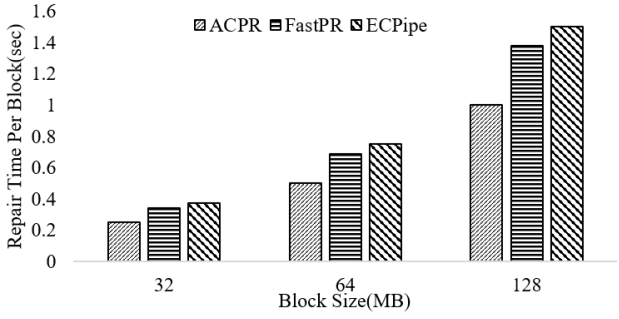

FIGURE 9. Theoretical analysis method validation.

Figure 10 and Figure 11 respectively show the repair times of ACPR, FastPR, and ECPipe under different RS encoding

parameters and different number of nodes. ACPR reduces the average repair time per block by 46.9% - 69.8% compared to FastPR and by 5.8% - 25% compared to ECPipe, respectively. This improvement is attributed to ACPR adopting a pipeline approach for the reconstruction method, which effectively optimizes bandwidth resources between nodes compared to reconstruction method of FastPR. Additionally, ACPR, can concurrently conduct migration repair while using repair pipelining reconstruction, leading to more efficient repair, compared to ECPipe.

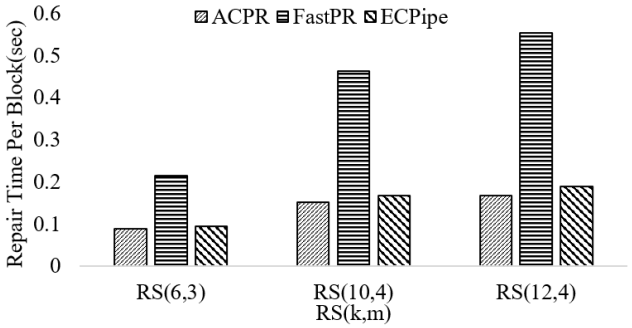

FIGURE 10. The impact of RS(k, m) on repair time.

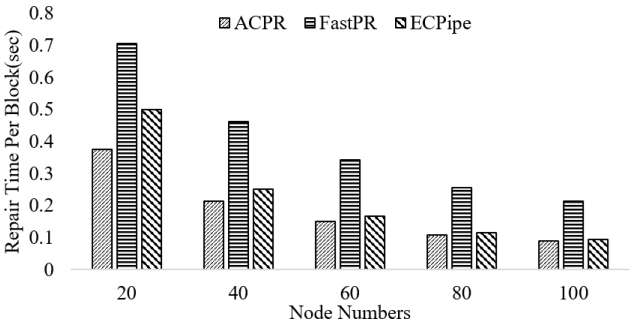

FIGURE 11. The impact of the number of nodes on repair time.

3) ANALYSIS OF THE REPAIR TRAFFIC

ACPR only generates cross node repair traffic when repairing high-risk blocks in STF nodes. Therefore, we assume that U' is the number of high-risk blocks identified as to be repaired in IR and SR scenarios, and the expression is as follows.

$$U' = U * \frac{k}{k+m} * 20\% \tag{5}$$

When a high-risk block is repaired using the repair pipelining reconstruction method, k data blocks need to be transferred, while only one data block needs to be transferred when using the migration method. According to the proportion of U, U' and the proportion of IR, SR and FD nodes, the average repair traffic of each node can be obtained by calculating the repair traffic of repair pipelining reconstruction and migration.

Figure 12 and Figure 13 show the average repair traffic of each node using ACPR, FastPR, and ECPipe under different RS encoding parameters and different numbers of nodes. Under various parameter conditions, ACPR reduces the average repair traffic per node by up to 72.9% - 78.4% and 81% - 84.2% compared to FastPR and ECPipe, respectively. This reduction is attributed to the strategy of ACPR delaying low-risk block repair. Consequently, all repair traffic is solely based on high-risk blocks repair, leading to an overall reduction of approximately 80% in repair traffic.

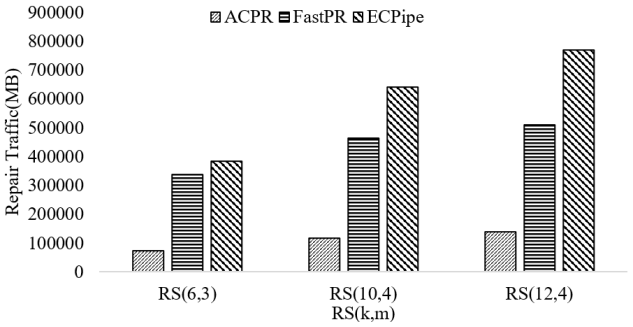

FIGURE 12. The impact of RS(k, m) on repair traffic.

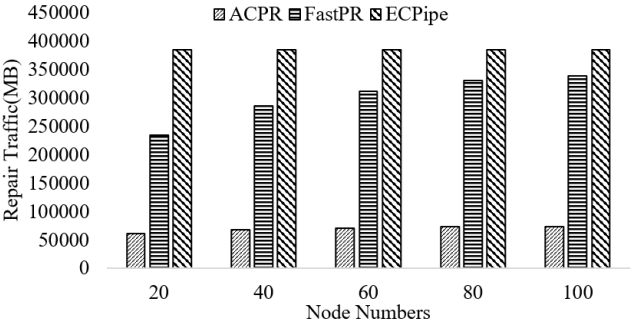

FIGURE 13. The impact of the number of nodes on repair traffic.

From the above theoretical analysis, it can be seen that even in larger cluster environments, the repair performance of ACPR is still superior to FastPR and ECPipe. Compared to FastPR and ECPipe, ACPR can shorten data block repair time by up to 69.8% and 25%, respectively, and reduce repair traffic by up to 78.4% and 84.2%, under different RS encoding parameters and different numbers of nodes in the cluster.

V. CONCLUSION

In this paper, we propose an Adaptive Classification Predictive Repair method ACPR for different fault scenarios. By categorizing the failed blocks into high-risk and low-risk based on the failure type of the STF node and the access heat of STF blocks, ACPR can perform adaptive predictive repair. The main idea is through coupling migration and pipeline reconstruction, high-risk blocks can be quickly repaired to ensure data availability, while delaying the repair of low-risk blocks can avoid unnecessary repair traffic. According to Alibaba Cloud ECS experimental evaluation, compared to FastPR and ECPipe, ACPR can shorten the repair time per

data block by up to 15.2% and 33.5%, respectively. Moreover, ACPR can shorten the total repair time compared to FastPR and ECPipe by 80.1% - 83% and 83.9% - 86.2%, respectively. Since ACPR couples the migration and repair pipeline reconstruction to achieve a reduction in repair time per block. And ACPR only needs to repair high-risk blocks within the node to ensure data availability, so ACPR demonstrates a greater advantage in overall repair time. And ACPR can reduce repair traffic compared to FastPR and ECPipe by up to 74.1% and 84.4%, respectively. Since ACPR only repairs high-risk blocks within the node, avoiding a large amount of unnecessary repair traffic. Therefore, ACPR outperforms FastPR and ECPipe in terms of repair traffic. In the event of node failure during the predictive repair process, ACPR can shorten the data repair time by up to 41.9% compared to FastPR. After the predictive repair is completed, ACPR reduces the average unavailable time of data blocks by up to 89.1% compared to ECPipe. We also conducted theoretical analysis of ACPR in larger scale environments to demonstrate its effectiveness. By analyzing and calculating the results in larger scale environments, ACPR can shorten data block repair time by up to 69.8% and 25%, respectively, and can reduce repair traffic by up to 78.4% and 84.2%, respectively, compared to FastPR and ECPipe. In future work, we will apply ACPR to larger scale real systems to further validate the performance of ACPR. We will also consider replacing RS encoding with other erasure codes with less repair traffic to further optimize ACPR.

REFERENCES

[1] Y. Song, T. Mu, and B. Wang, "HV-SNSP: A low-overhead data recovery method based on cross-checking," *IEEE Access*, vol. 11, pp. 5737–5745, 2023, doi: 10.1109/ACCESS.2023.3235787.

[2] Y. Song, W. Zhao, and B. Wang, "BPR: An erasure coding batch parallel repair approach in distributed storage systems," *IEEE Access*, vol. 11, pp. 44509–44518, 2023, doi: 10.1109/ACCESS.2023.3257404.

[3] X. Wang and Y. Liao, "New storage codes between the MSR and MBR points through block designs," *IEEE Access*, vol. 11, pp. 87120–87130, 2023, doi: 10.1109/ACCESS.2023.3299502.

[4] H. Zhou and D. Feng, "Boosting erasure-coded multi-stripe repair in rack architecture and heterogeneous clusters: Design and analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 8, pp. 2251–2264, Aug. 2023, doi: 10.1109/TPDS.2023.3282180.

[5] H. Qiu, C. Wu, J. Li, M. Guo, T. Liu, X. He, Y. Dong, and Y. Zhao, "EC-fusion: An efficient hybrid erasure coding framework to improve both application and recovery performance in cloud storage systems," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2020, pp. 191–201, doi: 10.1109/IPDPS47924.2020.00029.

[6] 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.

[7] X. Li, Z. Yang, J. Li, R. Li, P. P. C. Lee, Q. Huang, and Y. Hu, "Repair pipelining for erasure-coded storage: Algorithms and evaluation," *ACM Trans. Storage*, vol. 17, no. 2, pp. 1–29, May 2021, doi: 10.1145/3436890.

[8] S. Mitra, R. Panta, M.-R. Ra, and S. Bagchi, "Partial-parallel-repair (PPR): A distributed technique for repairing erasure coded storage," in *Proc. 11th Eur. Conf. Comput. Syst.*, Apr. 2016, pp. 1–16, doi: 10.1145/2901318.2901328.

[9] Z. Shen, J. Shu, Z. Huang, and Y. Fu, "ClusterSR: Cluster-aware scattered repair in erasure-coded storage," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, New Orleans, LA, USA, May 2020, pp. 42–51, doi: 10.1109/IPDPS47924.2020.00015.

[10] S. Lin, G. Gong, Z. Shen, P. P. C. Lee, and J. Shu, "Boosting full-node repair in erasure-coded storage," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2021, pp. 641–655.

[11] M. Ma, Y. Liu, Y. Tong, H. Li, P. Zhao, Y. Xu, H. Zhang, S. He, L. Wang, Y. Dang, S. Rajmohan, and Q. Lin, "An empirical investigation of missing data handling in cloud node failure prediction," in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Singapore, Nov. 2022, pp. 1453–1464, doi: 10.1145/3540250.3558946.

[12] Y. Zhou, F. Wang, and D. Feng, "A disk failure prediction method based on active semi-supervised learning," *ACM Trans. Storage*, vol. 18, no. 4, pp. 1–33, Nov. 2022, doi: 10.1145/3523699.

[13] J. Wang, W. Bao, L. Zheng, X. Zhu, and P. S. Yu, "An attention-augmented deep architecture for hard drive status monitoring in large-scale storage systems," *ACM Trans. Storage*, vol. 15, no. 3, pp. 1–26, Aug. 2019, doi: 10.1145/3340290.

[14] J. Yu, "Hard disk drive failure prediction challenges in machine learning for multi-variate time series," in *Proc. 3rd Int. Conf. Adv. Image Process.*, Chengdu, China, Nov. 2019, pp. 144–148, doi: 10.1145/3373419.3373437.

[15] H. Zhou, Z. Niu, G. Wang, X. Liu, D. Liu, B. Kang, H. Zheng, and Y. Zhang, "A proactive failure tolerant mechanism for SSDs storage systems based on unsupervised learning," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Tokyo, Japan, Jun. 2021, pp. 1–10, doi: 10.1109/IWQOS52092.2021.9521302.

[16] P. Li, J. Li, R. J. Stones, G. Wang, Z. Li, and X. Liu, "ProCode: A proactive erasure coding scheme for cloud storage systems," in *Proc. IEEE 35th Symp. Reliable Distrib. Syst. (SRDS)*, Budapest, Hungary, Sep. 2016, pp. 219–228, doi: 10.1109/SRDS.2016.039.

[17] Z. Shen, X. Li, and P. P. C. Lee, "Fast predictive repair in erasure-coded storage," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Portland, OR, USA, Jun. 2019, pp. 556–567, doi: 10.1109/DSN.2019.00062.

[18] Y. Wu, D. Liu, Y. Tan, M. Duan, L. Luo, W. Wang, and X. Chen, "LFPR: A lazy fast predictive repair strategy for mobile distributed erasure coded cluster," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 704–719, Jan. 2023, doi: 10.1109/JIOT.2022.3203415.

[19] P. Li, "Enabling low degraded read latency and fast recovery for erasure coded cloud storage systems," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2017, pp. 164–167, doi: 10.1109/DSN-W.2017.27.

[20] R. Nachiappan, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Optimized proactive recovery in erasure-coded cloud storage systems," *IEEE Access*, vol. 11, pp. 38226–38239, 2023, doi: 10.1109/ACCESS.2023.3267106.

[21] Y. Song, M. Yang, and B. Wang, "LEC-PR: Proactive recovery method in erasure-coded storage," in *Proc. IEEE/ACM 5th Annu. Workshop Emerg. Parallel Distrib. Runtime Syst. Middleware (IPDRM)*, Dallas, TX, USA, Nov. 2022, pp. 9–16, doi: 10.1109/IPDRM56689.2022.00007.

[22] D. Ford, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Vancouver, BC, Canada, Oct. 2010, pp. 1–14.

[23] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhani, "Erasure coding in windows Azure storage," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2012, pp. 15–26.

[24] L. Luo, Y. Tan, D. Liu, M. Duan, W. Wang, Y. Wu, and X. Chen, "Lazy repair with temporary redundancy(LRTR): Reducing repair network traffic in erasure-coded storage," in *Proc. 19th ACM Int. Conf. Comput. Frontiers*, Turin, Italy, May 2022, pp. 85–93, doi: 10.1145/3528416.3530240.

[25] Y. Song, Q. Zhang, and B. Wang, "FACHS: Adaptive hybrid storage strategy based on file access characteristics," *IEEE Access*, vol. 11, pp. 16855–16862, 2023, doi: 10.1109/ACCESS.2023.3243098.

[26] Y.-L. Lee, D.-C. Juan, X.-A. Tseng, Y.-T. Chen, and S.-C. Chang, "DC-prophet: Predicting catastrophic machine failures in D ata C enters," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 10536, Y. Altun, K. Das, T. Mielikainen, D. Malerba, J. Stefanowski, J. Read, M. Zitnik, M. Ceci, and S. Dzeroski, Eds. Cham, Switzerland: Springer, 2017, pp. 64–76, doi: 10.1007/978-3-319-71273-4_6.

[27] J. Fang, S. Wan, P. Huang, C. Xie, and X. He, "Early identification of critical blocks: Making replicated distributed storage systems reliable against node failures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2446–2459, Nov. 2018, doi: 10.1109/TPDS.2018.2833457.

[28] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Adaptive bandwidth-efficient recovery techniques in erasure-coded cloud storage," in *Euro-Par 2018: Parallel Processing* (Lecture Notes in Computer Science), vol. 11014, M. Aldinucci, L. Padovani, and M. Torquati, Eds. Cham, Switzerland: Springer, 2018, pp. 325–338, doi: 10.1007/978-3-319-96983-1_23.

[29] W. Huang, J. Fang, S. Wan, C. Xie, and X. He, "Design and evaluation of a risk-aware failure identification scheme for improved RAS in erasure-coded data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 16–30, Jan. 2021, doi: 10.1109/TPDS.2020.3010048.
[30] Z. Wang, H. Wang, A. Shao, and D. Wang, "An adaptive erasure-coded storage scheme with an efficient code-switching algorithm," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Edmonton, AB, Canada, Nov. 2020, pp. 1177–1178, doi: 10.1109/ICDCS47774.2020.00129.

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 platforms. Since 2007, she has been authored or coauthored more than 30 publications in these areas. She served in various academic conferences. Her main research interests include computer architecture, parallel and distributed computing, and virtualization technology.

PEISEN ZHENG received the B.S. degree in computer science and technology from the Beijing Information Science and Technology University, Beijing, China, in 2022, where he is currently pursuing the master's degree with the Computer School. His main research interest includes distributed storage.

YINGAI TIAN received the B.S. degree in computer science and technology from the Beijing University of Science and Technology, Beijing, China, in 2006. She is currently an Associate Professor with the Computer School, Beijing Information Science and Technology University. Her main research interests include big data analysis methods, machine learning algorithms, document information pressing technology, and document format standards.

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. From 2012 to 2016, he was a Guest Student with the State Key Laboratory of Computer Architecture, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He is currently a Lecturer with the Software Engineering College, Zhengzhou University of Light Industry (ZZULI). He has served in various academic journals and conferences. His research interests include distributed systems, cloud computing, edge computing, resource management, and task scheduling. He has published more than ten research articles in these areas.