


Minimizing Performance Degradation of RAID Recovery Through Pre-Failure Prediction

Jialin Liu², Yujiong Liang², Yunpeng Song², Yina Lv⁴,  Liang Shi^{1,2,3}

¹Software/Hardware Co-design Engineering Research Center, Ministry of Education, Shanghai, China

²School of Computer Science and Technology, East China Normal University, Shanghai, China

³Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China

⁴Department of Computer Science, City University of Hong Kong, Hong Kong, China

Abstract—Redundant Array of Independent Disks (RAID) is one of the mostly adopted storage systems to achieve fault tolerance capability. For parity-based RAID systems, which have high capacity utilization and good fault tolerance, multiple storage devices are arranged in the array to store upper-level data and parity. The RAID system recovers data on a failed storage device using the remaining data and parity on other devices and writes them to a new replacement device. The RAID system suffers from degraded performance and fault tolerance ability during an intolerably long recovery process which can last for several hours.

In this paper, we propose that existing storage device fault prediction mechanisms can be adopted to avoid such intolerably long degradation time. The prediction mechanism can identify faulty devices before they actually fail and start creating mirror devices for them, which can immediately replace the original device with no degradation time. Current prediction models usually focus more on precision rate than recall rate to avoid replacing too many healthy storage devices and sacrifice their ability to find more potentially faulty storage devices. Based on the above observations, we propose a two-stage backup mechanism that enables the models to have a higher recall rate to find more faulty storage devices. Instead of directly replacing the predicted faulty device, the mirror device will be synchronously updated with the predicted faulty device for a certain period of time until it actually fails. By avoiding directly removing the predicted faulty device, the disadvantage of a low precision rate is reduced. Numeric modeling analysis shows that our mechanism can reduce the degradation time of the whole fleet in the data center.

I. INTRODUCTION

Servers with multiple storage devices (e.g., hard disk drive (HDD), solid-state drive (SSD)) are widely deployed in data centers to provide large storage capacity to hold the ever-growing amount of data. Multiple storage devices also present multiple risks of device failure for these servers. Failure on a storage device will make all data on the device inaccessible. As data loss is unacceptable under scenarios like data center applications, data recovery capability is necessary under these scenarios. To provide data recovery capability, the storage system that manages these storage devices stores redundancy information along with the user data. When data recovery is triggered, the redundancy information is used for rebuilding

the lost data and writing them to the storage device that replaces the faulty one.

Because of its simplicity and high capacity utilization, the parity-based Redundant Array of Independent Disks (RAID) system [1] has become one of the widely deployed storage systems that provide such data recovery ability. It has been widely used in data analytics [2], machine learning [3], enterprise storage [4], and cloud environments [5]–[7]. In parity-based RAID systems, parity of user data is used as redundancy information. Lost data on a faulty device can be calculated with parity and the remaining user data and written to the replacing device. When a tolerable number of storage devices fail, a parity-based RAID system will enter a degraded mode. The degraded mode duration should be shortened for at least two reasons. First, in a degraded mode, the parity-based RAID system still functions with no data loss but serious performance degradation due to the loss of original user data. Second, a degraded RAID system will have weaker or no fault tolerance ability due to the redundancy reduction. To exit the degraded mode, the faulty storage device must be replaced by a normal one, and a recovery process must be started. However, the recovery process usually requires a long time (e.g., tens of hours), leaving the RAID degraded for a long period [8]–[10].

Existing works [11], [12] focus on reducing the duration of the recovery process. But with the limited I/O throughput of the replacing device, the recovery process is still too long. Some works [13]–[16] try to predict the failure of certain storage devices, utilizing the statistical (i.e., Self-Monitoring Analysis and Reporting Technique (S.M.A.R.T.) [17]) information of these devices. Once the storage device is predicted as a faulty device, it can be replaced and the recovery process is triggered. Prediction can be adopted to recover the faulty storage devices in advance. A mirror device of the potentially faulty device can be immediately created once the prediction process identifies the device. The mirror device can replace the faulty device immediately after it is completely generated, causing a very short degradation time. However, the problem is that prediction is not always accurate. Considering the high price of enterprise-level SSDs, current prediction-based works tend to trade recall for precision. In simple terms, current works tend to have fewer healthy devices mispredicted and more potentially faulty devices undiscovered. The undiscovered faulty devices still cause degradation in

This work is supported by the NSFC 62072177, Shanghai Science and Technology Project 22QA1403300 and the Open Project Program of Wuhan National Laboratory for Optoelectronics NO.2023WNLOKF004. The corresponding author is Liang Shi (shi.liang.hk@gmail.com).

RAID, which restricts the effect of prediction.

In this paper, we propose a novel method that enables the prediction to have a higher recall rate, leaving fewer potentially faulty devices undiscovered during prediction. The challenge is that this strategy will cause more mispredicted healthy devices to be replaced, which may cause waste and increase overheads. To solve this challenge, we propose a mechanism that enables the original device to continue functioning. A mirror device will be created as the backup. Instead of directly removing the original one, we leave the original device to continue functioning till its failure. The mirror device can replace the original device immediately if the failure happens. Otherwise, the mirror device is released and can be used as other potentially faulty devices' mirrors. In this way, the RAID system obtains an increase in reliability and average performance.

The contributions of this paper are as follows.

- We propose a prediction-based pre-failure recovery mechanism to avoid long RAID degradation time caused by device failure.
- A two-stage backup mechanism is proposed to enable prediction models to have a high recall rate, i.e., find more potentially faulty devices.
- Four exceptional cases of the proposed mechanism and the corresponding countermeasures are discussed to improve the robustness of the proposed mechanism.
- Experimental results show that the total performance degradation time reduction in a data center can be improved by 150% with 0.06% more storage devices as overheads.

II. BACKGROUND AND MOTIVATION

A. Parity-based RAID

1) *Mechanism of Parity-based RAID*: Considering all forms of storage systems, the parity-based RAID systems provide storage device fault protection ability with low cost and complexity. Typical parity-based RAID methods include RAID5 and RAID6. The storage space of a parity-based RAID system is organized in stripes. Each stripe is composed of chunks, and every chunk in a stripe comes from a different storage device in the array.

The parity chunks used as redundancy are generated by performing the XOR operation on the user data chunks. Each stripe in RAID5 contains one parity chunk while each stripe in RAID6 contains two parity chunks. The parity enables RAID5 to tolerate up to one device failure with no data lost. RAID6 can tolerate up to two device failures with no data lost. The parity chunks of each stripe spread across all storage devices in the array to spread I/O pressure. To recover a lost user data chunk in the stripe, the rest of the user data chunks in the stripe and the parity chunk(s) must be read from the remaining storage devices. The lost user data chunk can then be calculated by performing the XOR operation on these chunks.

2) *RAID Degradation*: Although no data loss happens when a tolerable number of storage devices fails, the RAID system enters a degraded mode. Since obtaining the lost data chunk on the failed device incurs extra read operation and calculation, the total read performance degrades inevitably. To show the degradation problem of RAID, we use three SSDs to form a normal RAID5 array (platform information is shown in Section IV). For comparison, we set one SSD as a failure device to form a degraded RAID5 array. FIO is adopted to perform the evaluation under both circumstances to report 4KB-random-read throughput and 1MB-sequential-read throughput. As shown in Table I, the degraded RAID5 can only provide 21.8% random read throughput and 24.17% sequential read throughput compared to the normal RAID5. Besides performance degradation, the fault tolerance ability is also degraded, as the redundancy in the array reduces. For a degraded RAID5, another storage device failure will cause data loss in the array, which can hardly be recovered. Therefore, parity-based RAID must exit the degraded mode as soon as possible.

TABLE I: Performance degradation of RAID.

	Normal RAID5	Degraded RAID5
Random Read Throughput	5347 MB/s	1142 MB/s
Sequential Read Throughput	10600 MB/s	2563 MB/s

3) *Disadvantage of Post-failure Recovery*: Considering modern storage devices' high capacity and parity-based recovery's complexity, the recovery process of parity-based RAID systems will take hours to recover, leaving the RAID systems degraded for a long time. As shown in Figure 1a, a traditional measure against device failure under parity-based RAID is a post-failure recovery mechanism. To handle device failure in time, a RAID system is usually equipped with a spare device, which stores no data and is used for replacing faulty devices. The spare device will replace the faulty device after the failure happens. Before the spare device is applicable in the array, the data on the faulty device must be recovered to the spare device. As mentioned in Section II-A1, chunks in the remaining devices are read to calculate the lost data. This complicated process and the large volume of data can cause an intolerably long degradation period, which can be worsened with a certain amount of foreground workload pressure.

To avoid the long degradation time issue, one simple solution is through a pre-failure recovery scheme, as shown in Figure 1b. The basic process is as follows: First, a predictor is used to predict potentially faulty devices. Then a spare device is used for preparing a mirror device of the potentially faulty device that is still functioning. The mirror device can replace the faulty device with no degradation time since it already carries the data on the faulty device. The advantage of the pre-failure recovery mechanism comes from two parts. First, data written to the spare device in advance eliminates the need to recover data under degraded mode, thus eliminating the degradation period. Second, with the original device still functioning, the mirror preparation step is much

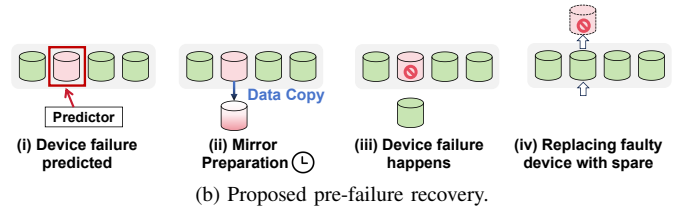
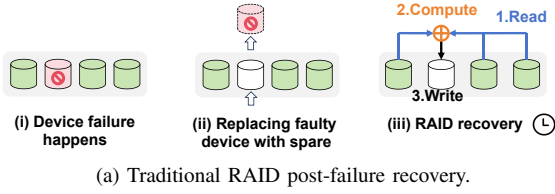


Fig. 1: Comparison between post-failure recovery mechanism and pre-failure recovery mechanism.

less complex than the parity-based RAID recovery process. These advantages significantly reduce the recovery's influence on foreground workloads. In this paper, a pre-failure prediction based recovery method will be proposed to solve the long degradation time issue.

B. Storage Device Fault Prediction

1) *Typical Prediction Process:* As compared to passive recovery of RAID systems, methods for predicting failures are increasingly being investigated. Predicting failures can leave enough time for operation staff to protect the data, such as migrating them to a completely new device. Since device failure is often related to many factors, such as workload, environment, age, etc., this leads it to be a complex statistical problem. So, predictive methods tend to use machine learning algorithms to help find the patterns of the failed devices. Overall, the machine learning based prediction process consists of four steps as shown in Figure 2.

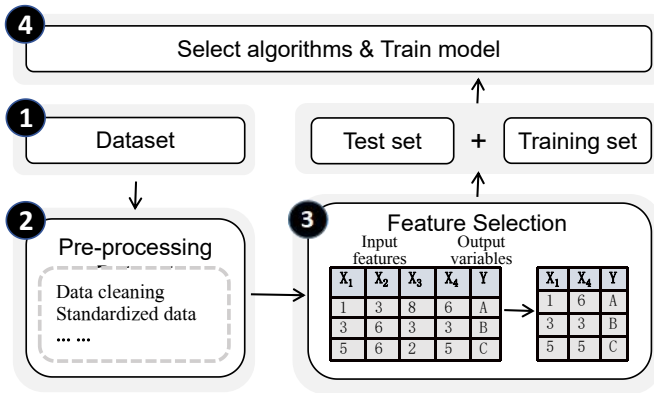


Fig. 2: The machine learning based prediction process.

First, data for the device is collected (step 1). This is because the essence of the model is to make judgments about the future by learning the trends of various attributes in large amounts of data. The S.M.A.R.T. is the most used information. This technique intermittently records information about the current state of the device as well as other factors such as environment and workloads. Second, since the S.M.A.R.T. mechanism is not completely reliable, the collected data may often be missing. Also, different attributes have different types of values, so the data needs to be pre-processed (step 2), containing operations such as padding and normalization. Third, feature selection is particularly important (step 3). This is because the S.M.A.R.T. mechanism contains too many

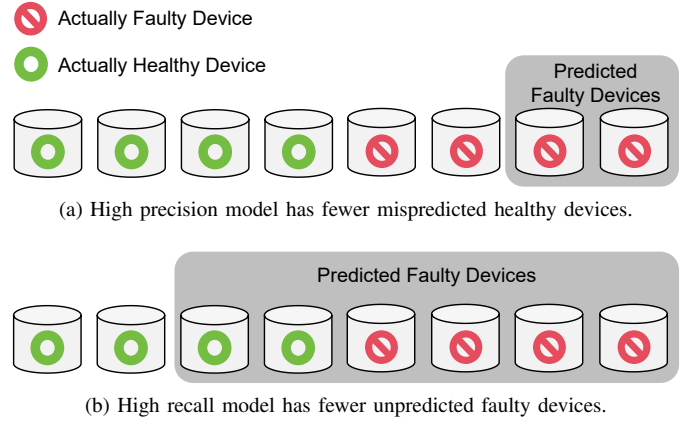


Fig. 3: Comparison between high recall model and high precision model.

types of attributes, but not all of them contribute to failure prediction. On the contrary, irrelevant attributes can lead to longer model training time and even affect the prediction performance. Therefore, in this step, data with some irrelevant attributes will be eliminated. Finally, the data is split into a training set and a test set. The training set is used to train the model while the test set is used to evaluate the performance of the model. Then, the appropriate machine learning algorithm is selected and the model is being built using the sliced and diced data set (step 4).

To better explain our prediction results and design philosophy, subsequent descriptions will classify the samples into four categories i.e., True Positive (actually faulty, predicted to be faulty), False Positive (actually healthy, predicted to be faulty), True Negative (actually healthy, predicted to be healthy) and False Negative (actually faulty, predicted to be healthy).

2) *Challenges of prediction methods:* For the performance of a model, we usually characterise it with the following two metrics:

- **Precision Rate (PRE):** The ratio of the number of correctly predicted failure devices to the number of devices predicted to fail.
- **Recall Rate (RECALL):** The ratio of the number of correctly predicted failed devices to the total number of failed devices.

For both metrics, higher is better. However, for well trained models, there is a trade off between the two metrics, i.e., improving one metric usually reduces another metric. We

briefly compare the difference between high precision models and high recall models, as shown in Figure 3. For the high precision model in Figure 3a, fewer misidentified failures exist but fewer failure devices are identified compared to that in Figure 3b. And for the high recall model in Figure 3b, more failure devices can be identified, but they also misidentify healthy devices as positive. In most cases, high precision models are usually chosen to avoid incorrectly discarding healthy devices as faulty devices, which may waste all those healthy devices. However, such an approach would also make the model weak in identifying failure devices. It becomes a challenge to use predictive models effectively. But if we do not discard the positive devices immediately, we may have a chance to see whether the devices are actually faulty, thus enabling us using a high recall model.

III. DESIGN

A. Overview

To fully utilize prediction to help shorten the degradation period, we propose a two-stage pre-failure recovery method to make a backup for the predicted positive devices in advance. The first stage is the mirror generation stage, which generates a mirror device for the positive devices given by prediction. This mirror device can be treated as a backup of the original device and replace the original device at any time to avoid the original device causing failure. The second stage is the mirror maintenance stage, keeping both original device and mirror device functioning and waiting for failure. Mirror maintenance is adopted so that the positive devices are not discarded immediately, enabling the mechanism to see whether the devices are actually faulty. This enables our mechanism to adopt a high recall model, as mentioned in Section II-B2.

Figure 4 shows the whole process of our mechanism. Firstly, a recall-enhanced model is trained offline for the prediction job. Secondly, the S.M.A.R.T. information is periodically fetched as data input of the model. The model then uses the accumulated S.M.A.R.T. information to predict which storage devices are going to fail. Thirdly, for these predicted faulty devices, our method starts to generate a mirror for them. A spare device will be used for generating the mirror device (i.e., data of the original device will be synchronized to the spare device). Finally, the original device will be monitored for a certain period of time as the observation period. The mirror device will be updated synchronously with the original device during the observation period (in a RAID1 manner). The mirror device can replace the original device immediately after the original device fails. If the original device survives during the observation period, the mirror device will be released and can be used for mirroring other devices.

B. Model Tuning

As previously stated, our design relies on a recall-enhanced version of the model. Therefore, how the performance of the model is tuned is fundamental to realizing our design. Failure prediction can be viewed as a binary classification task where the model simply outputs 1 or 0, with 1 representing predicted

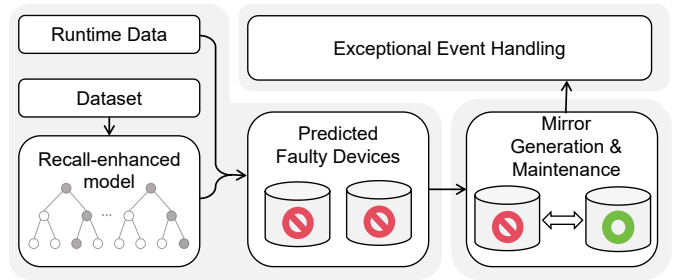


Fig. 4: Process overview.

failure and 0 representing predicted health. In real practice, we can take from the machine learning model the probability that the sample will be identified as a 1. This means that each sample will have a probability value distributed between 0 and 1, as shown in Figure 5. Typically, the probability value of a device that is actually faulty will be closer to 1, while a device that is actually healthy will be closer to 0. However, there will always be situations where the distribution of these two types of devices intersects, which is also why the prediction cannot be perfect. In this case, we can set the threshold. For the predicted value above this threshold, we consider this sample to be predicted as faulty. Conversely, it is considered to be predicted as healthy. By adjusting the threshold, we can easily tune the model, setting it to a high recall model (or a high precision model if needed). As shown in Figure 5, we show three thresholds as examples. Threshold 3 identifies all faulty devices and identifies some healthy devices as faulty as well, at which point the model becomes a typical high recall model. Similarly, threshold 1 represents a high precision model and the threshold 2 represents a balanced model.

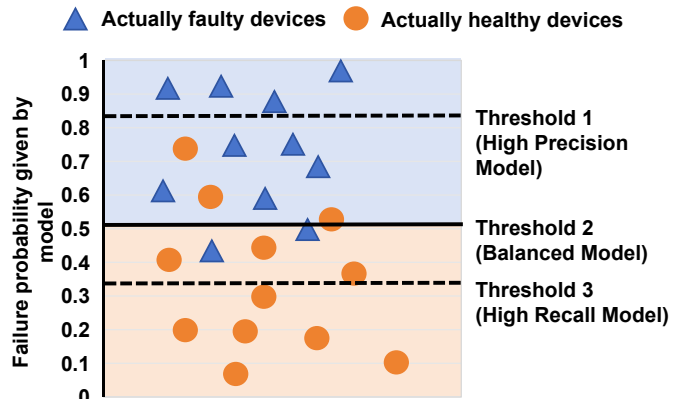


Fig. 5: Schematic overview of model performance tuning.

It should be emphasized that in our design, we do not restrict what model to use, we only focus on the tuning of the model. In fact, tuning model performance by thresholds is a very pervasive method that is applicable to almost all machine learning algorithms used for classification tasks. In the subsequent design, in this paper, we take Random Forest (Random Forest [18]), which is currently used widely, as an

example, and build a real model with real data to verify the reasonableness of our design.

C. Mirrors for Predicted Failure Devices

To provide a substitute for the positive storage device in the failure prediction, our mechanism generates a mirror device. The mirror device has the same content as the original device and can immediately replace the original device. Then, once the predicted device fails, the mirror device can replace the faulty device quickly. However, such a straightforward strategy prefers a high precision of prediction to avoid replacing mis-predicted healthy devices, which increases the maintenance cost. This limits the amount of potentially faulty devices that the prediction process can discover.

To solve this issue, our mechanism also maintains the mirror device for an observation period after the mirror device is set up. False positive devices can survive an observation period. After the observation period, the related mirror devices can be released if the original device is still functional (will be further discussed in Section III-D3). The true positive devices are likely to fail during the observation period and the mirror device can replace the failed device immediately with little degradation time. The maintenance of the mirror device requires synchronous updating with the original device for consistency. Therefore our mechanism forms a RAID1 array of the original device and the spare device to make the spare device become the mirror of the original device.

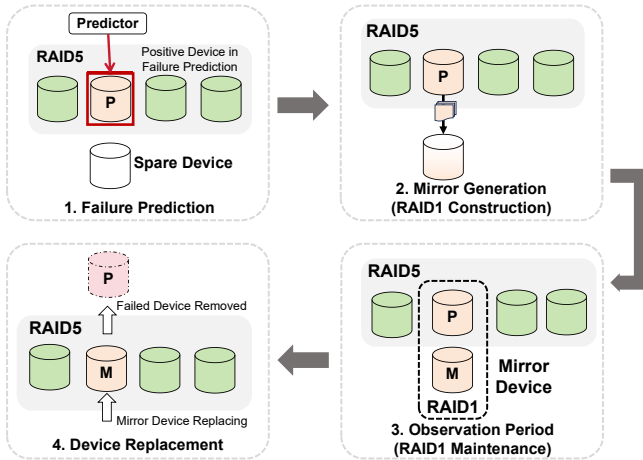


Fig. 6: Lifetime of mirror device.

The process is presented in Figure 6, which uses a 4-disk RAID5 as an example. Firstly, a positive device is detected by the prediction algorithm. Secondly, our mechanism begins to use a spare device to form a RAID1 array with the positive device, which is still functioning currently. Thirdly, the RAID1 functions as a single device in the upper-level RAID5 during the observation period. The RAID1 carries the data on the positive device and waits for the device failure. Finally, after the original device fails, the original device is removed and the mirror device remains to serve as the original device.

D. Exceptional Event Handling

The above design is simple and easy to avoid the long degradation period. However, there are several exceptional events not considered. In this subsection, we mainly discuss the handling of possible exceptional events (i.e., events outside the normal process) to ensure the robustness of our mechanism.

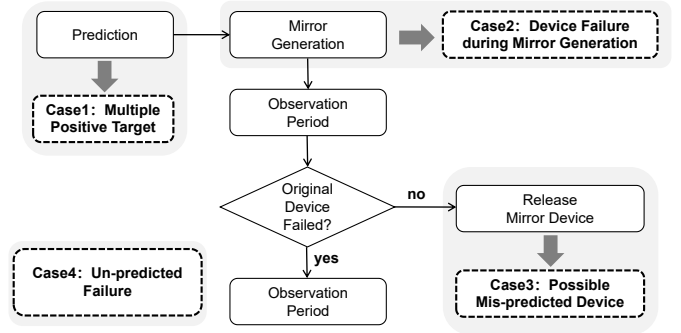


Fig. 7: Possible exceptional events during process.

As listed in Figure 7, four exceptional events are discussed below. The first event, multiple positive targets, happens when more positive devices are given by the predictor compared to the spare devices available on the server, forbidding our mechanism to provide a mirror device for every positive device. The second event refers to the case that a device fails during the mirror generation process, disrupting the mirror generation process. The third event, possible mispredicted devices, refers to the case that a healthy device is identified as positive, which may occupy the mirror device meaninglessly. The fourth event, un-predicted failure, refers to un-predicted failures (i.e., false negative devices), which cannot be handled by our pre-failure recovery mechanism. In the following, these four events are discussed in detail.

1) *Multiple Positive Targets*: There is a possibility that multiple storage devices on a single server are predicted as positive. As mentioned above, each positive device requires a spare device to generate its own mirror device. Thus, in our scheme, the number of spare devices on the server must be no less than the number of positive devices to generate and maintain mirror devices. However, due to the limited number of storage device slots on a server, the available spare devices may not be enough.

An optimistic solution is to consider that not all devices require urgent backup since prediction precision is reduced in our mechanism. The devices' probability value given by the prediction model is compared to decide the devices that require the most urgent handling. Mirror devices will only be generated for the most urgent devices given by the prediction model. The remaining devices are left without any backup and recovered by a post-failure recovery process in the case of failure. The downside of the optimistic solution is that the unprotected remaining devices are still likely to become faulty and cause degradation in RAID.

To avoid the degradation issue, our mechanism adopts a pessimistic solution. The pessimistic solution is to consider all devices as faulty and directly replace positive devices that are beyond our ability to generate mirrors. Our mechanism begins with generating mirrors using the available spare devices for part of the positive devices. Figure 8 shows an example of our mechanism. At first, the first four devices are predicted as positive, while only two spare devices are available on the server. Then, our mechanism generates mirror devices for devices 3 and 4, assuming that they have higher probability value given by the prediction model than devices 1 and 2. As we adopt a pessimistic solution, devices 3 and 4 are thought to be faulty. So they are immediately replaced by their corresponding mirror devices. Their observation periods are omitted as they are thought to fail in the near future to make room for other devices' mirror devices. With devices 3 and 4 replaced, their slots can be used for new spare devices to be plugged in. The observation period is omitted until there are enough spare devices to generate a mirror device for every remaining positive devices. If the number of spare devices is enough, the remaining positive devices won't be immediately removed after their mirror devices are generated and will enter the observation period. In the given example, the spare devices are enough after devices 3 and 4 are replaced, so devices 1 and 2 are each provided a spare device for generating a mirror device and entering the observation period. Note that if a mirror device exists before the number of positive devices exceeds the number of spare devices, the existing mirror devices' observation period will be omitted first.

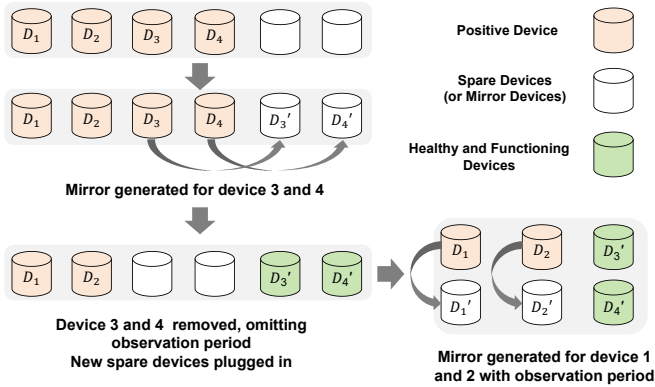


Fig. 8: Handling multiple positive targets.

Such a pessimistic solution may cause healthy devices (false positive devices) to be replaced compared to the optimistic solution. However it reduces the probability of data loss and array degradation, which is worthy. Beside, the situation where the number of positive devices detected is greater than the number of spare devices is a rare case. So this pessimistic solution won't cause too much waste of healthy devices.

2) *Device Failure during Mirror Generation:* It's possible that a device fails before the mirror generation process completes. This event can be handled in an intuitive way. In the case that the original device breaks before the mirror

generation process completes, the data on the mirror device can be abolished and the mirror device can be directly used as the device for RAID recovery. In this case, the RAID enters degraded mode and a post-failure recovery is executed. With the prediction enabled design, this will be a rare case. Then, it's acceptable to use such a naive policy.

3) *Handling False Positive Devices:* In our mechanism, the precision of prediction is supposed to be reduced to have a high recall. This may cause the prediction process to identify more healthy storage devices as positive. Our mechanism also generates mirror devices for these mispredicted devices as it can't tell these false positive devices from the true positive devices. In this case, the mirror devices are wasted since the original device doesn't actually need a backup. This may cause inadequate spare devices on the server when other storage devices are identified as faulty later. The meaningless wear of the spare device also reduces the endurance of the spare device without any benefit. The previously mentioned observation period is set to reduce the effect of the problem mentioned above.

Two problems for the design of the observation period should be discussed. Firstly, the length of the observation period should be confirmed. We choose 10 days as the length of the observation period, which is an empirical value used by existing work [16] and can be adjusted. In fact, the observation period should be set to coincide with the prediction period. This is intuitive since the probability given by the prediction model is the probability that the device will fail in prediction period. The prediction period refers to the limit of the prediction algorithm on how much time in the future a failure will occur. Secondly, the subsequent processing of the devices that survive the observation period should be considered. Despite surviving the observation period, the device can be still identified as a positive device in the later prediction. This is due to two reasons. First, the prediction model depends on the device's current and historical statistical information, which may remain similar to that in the previous prediction and cause the device to be identified as positive again. Second, the device itself may actually be a potentially faulty device that survives longer than the observation period. However, the true positive devices that can survive the observation period are rare. Thus our mechanism treats all the devices that survive the observation period as healthy devices. These devices are tagged and mirror device generation is forbidden for these devices. Although this may leave the device unprotected when it's really going to fail, the influence is small enough to be ignored as this case is rare.

The whole process of the observation period and its succeeding process is summed in Figure 9. First, the mirror device is generated for the positive device and back it up. Then the two devices enter the observation period (10 days or other duration), waiting for the positive device to fail. If the positive device fails during the observation period, then the faulty device is removed and replaced by the mirror device. Otherwise, the positive device continues to function and survives the observation period. In this case, the current

mirror device is released and the positive device is tagged as healthy. Later prediction results may identify it as positive again, but the tag will stop our mechanism from generating a new mirror device for it.

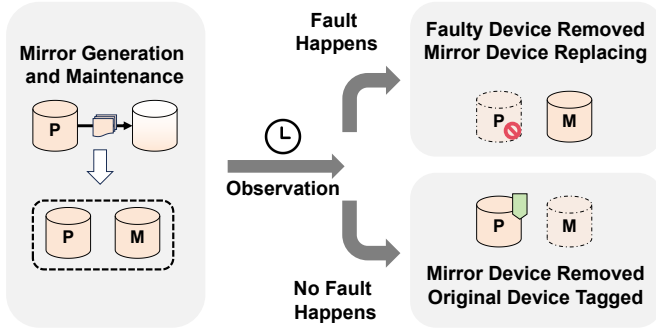


Fig. 9: Observation period and succeeding process.

4) *Unpredicted Failure*: Although we raise the recall of the prediction, unpredicted failures (i.e., false negatives) are still inevitable. These unpredicted fail devices are replaced using healthy devices and recovered by the post-failure recovery process. They have no influence on the mirror generation process and the observation period.

E. Discussion

As we mentioned above in Section III-D3, our mechanism may generate mirror devices for the false positive devices, which are actually healthy. Despite these misused mirror devices are finally released, the mirror generation and maintenance process may cause wear to them, which is a hidden cost in our method. However, as we will show in Section IV-B, with a proper threshold set for the prediction, the false positive devices can be rare. So these wear are acceptable. There is actually a trade-off between the required extra mirror device and RAID degradation time reduction. Our evaluation proves that a proper point can be found.

As we will show in Section IV-A, our mechanism has limited improvement in write throughput. This is mainly because the complexity of the write process in parity-based RAID, especially with a small I/O size, is similar to that of the degraded read process. Thus the write process degradation isn't obvious compared to that of the read process. This limited optimization space limits our mechanism's effect.

As mentioned earlier, the proposed pre-failure recovery mechanism cannot handle all device failures due to the inevitable existence of unpredicted device failures in prediction. Therefore, the post-failure recovery mechanism is still necessary to handle unpredicted device failures. But as there is a non-negligible number of predicted device failures, the pre-failure recovery mechanism can still have an inspiring effect on degradation time reduction, which will be further studied in Section IV-B2.

IV. EVALUATION

Restricted by experimental resources, we can't prove our mechanism in a data center. Thus we carry out the evalua-

tion in two steps. First, we evaluate the pre-failure recovery mechanism's benefit in a single RAID system in Section IV-A. In this section, we disable a storage device in the RAID to simulate a device failure. The degradation time and effect on the performance of RAID are evaluated. Second, we use the degradation evaluation results in Section IV-A and the performance of the prediction model tuned in Section III-B to form a numeric model to simulate the situation in a data center in Section IV-B. We use the numeric model to evaluate the total benefit of our mechanism in a data center.

A. Single RAID System Evaluation

1) *Experiment Setup*: As mentioned above, we carried out our experiments on a server with two 20-core Intel XEON CPUs and 384GB DRAM, running Ubuntu 22.04 with Linux kernel v6.2.0. We use four 1TB ES3500P SSDs as the high-density SSDs, three for array construction and one for replacement. For simplicity, we only use an 128GB partition on each SSD to construct the parity-based RAID. For the RAID managing system, we adopt the widely adopted Linux mdraid [19] and use RAID5 as the RAID format for the four SSDs. The chunk size of RAID is set to 512KB by default and the RAID worker thread number is set to 16. For the foreground workload, we use FIO [20] to generate read and write workloads. I/O size is set to 4KB and 1MB for random workload and sequential workload, respectively. We compare the proposed pre-failure recovery mechanism with the traditional post-failure recovery mechanism. For the pre-failure recovery mechanism, our evaluation prototype utilize the mdraid's RAID1 for mirror device generation and maintenance. For the post-failure recovery mechanism, the failed device is recovered by the mdraid using parity and remaining data.

2) *Degradation Comparison*: In this section, the degradation period length and performance degradation are compared between the post-failure recovery mechanism and the pre-recovery mechanism. As mentioned above, the degradation period of the post-failure recovery mechanism refers to the time after the device failure (i.e., step (iii) in Figure 1a). The recovery process is supposed to start immediately after device failure is detected. For our proposed pre-failure recovery mechanism, the benefit exists when the failure is previously predicted and a mirror device is generated and kept until the failure happens. In this case, replacing the faulty device with the mirror device won't take more than a second. However, the mirror generation process lasts long and may have negative effects on performance. Thus we define the degradation period of the pre-failure recovery mechanism as the mirror generation period (i.e., step (ii) in Figure 1b).

Degradation Throughput Evaluation: We first measured the foreground throughput (with fio thread number set to 32 and queue depth set to 16, which saturate the throughput of our implemented RAID system) during degradation to compare the performance of the two mechanisms. The results are shown in Figure 10. For the pre-failure recovery mechanism, the performance degradation is subtle. The degraded throughput

of foreground workload is no less than 93% of the normal throughput. The mirror generation process doesn't have much influence on foreground workload. In contrast, the throughput degradation of the post-failure recovery mechanism is obvious. The random read throughput degrades to 23.4% of the normal throughput. And the sequential read throughput degrades to 23.9% of the normal throughput. This is because data on the failed device is lost and must be recovered by the parity-based recovery process, which involves reading all remaining devices and calculation. Such a process significantly degrades the throughput. The write throughput degrades slightly in both mechanisms due to complexity of the write process, as mentioned in Section III-E.

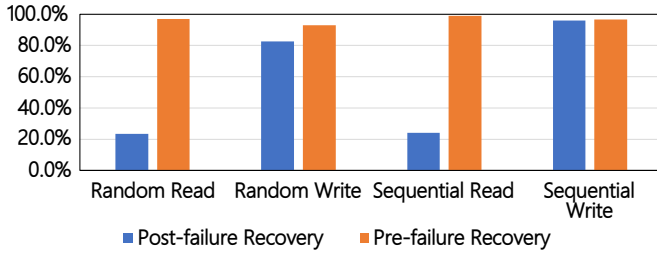


Fig. 10: Degraded throughput relative to normal throughput (the higher is better).

We further look into the case of 4KB-random-read accesses. We modify the queue depth and thread number job parameters to generate workloads of different pressures. Then we run the workload of the same parameter separately under the degraded state and normal state of the two recovery methods for evaluation. The throughput degradation of the same workload is shown in Figure 11. The degraded performance of pre-failure recovery varies between 96.9% and 99.0% of the normal performance, which shows subtle degradation. In contrast, the degradation of post-failure recovery failure can be degraded to 72.0% to 79.8% under a low foreground workload pressure. With the foreground workload pressure going higher, the performance of RAID under post-failure recovery failure can be degraded to only 23.4% of the normal performance.

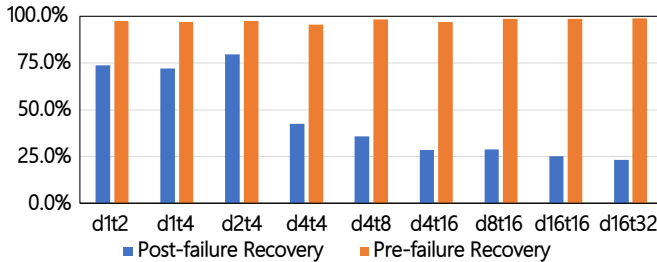


Fig. 11: Degradation of 4KB-random-read throughput. For labels on the x-axis, d refers to queue depth, t refers to thread number, and the following number is their value separately.

We also run a 4KB-random-read FIO workload and set a device to faulty or begin a mirror generation process to evaluate the transient influence on performance. Figure 12

shows the results. Under the traditional post-failure recovery mechanism, where mirror generation is not involved, a device failure directly causes degradation and reduces the throughput sharply to only about one-fifth of the normal throughput, as shown in Figure 12a. Under the proposed pre-failure recovery mechanism, we first evaluate the mirror generation process. The mirror generation process only causes a degradation of less than 100MB/s, shown in Figure 12b. After the mirror generation completes, we set the original device to faulty. As shown in Figure 12c, the device failure in the pre-failure recovery process only causes 150MB/s of throughput degradation, which is subtle compared to that of the post-failure recovery mechanism. Note that the throughput increases a bit after the mirror generation process completes due to the use of RAID1 as the mirror maintenance method. So in the left half of Figure 12c, where the mirror is already generated and maintained, the performance is about 150 MB/s higher than that in Figure 12b.

Degradation Time Evaluation: In this part, we compared the degradation time of the two mechanisms. Results are shown in Table II. We first test the degradation time without foreground workload (i.e., idle state). In this case, the degradation takes a similar time under both mechanisms. Both degradation periods last for about 11 minutes. A shorter degradation duration may be achieved by adopting a more aggressive recovery strategy theoretically since the bandwidth utilization under both circumstances is far less than the maximum. With a shortened degradation duration, our mechanism is more favorable compared to the post-failure recovery since the RAID1's replica-based recovery is theoretically simpler than a parity-based recovery. But here we adopt the default mdadm configuration, which is conservative and tries not to occupy too much of the system resources. Then we set a FIO-random-read workload as the foreground workload. The FIO workload is set to acquire the maximum available throughput of the RAID system (i.e., 32 threads with a queue depth of 16) to keep the RAID system busy. In this case, the pre-failure recovery mechanism's degradation time still lasts for 11 minutes. However, the post-failure recovery process takes about 41 minutes, which is significantly extended. The recovery process of RAID1 (i.e., mirror generation process) is simpler than the parity calculation of RAID5, which guarantees a faster recovery.

TABLE II: Degradation time. The foreground workload is a 4KB-random-read workload.

	Pre-failure Recovery	Post-failure Recovery
Without foreground workload	11min	11min
With foreground workload	11min	41min

In conclusion, degradation hardly exists in a pre-failure recovery process, and the duration of degradation is also significantly shortened. Note that though the pre-failure recovery still lasts for tens of minutes or hours when the device capacity comes to the terabyte level, the performance degradation is subtle with pre-failure recovery. As a result, the pre-failure

recovery mechanism will have a relatively shorter degradation duration with almost unnoticeable performance degradation.

B. Group Modeling Analysis

1) *Prediction Evaluation*: To validate the effectiveness of our design in clusters, we replicate a failure prediction model to explore how *Precision* and *Recall* really vary and what benefits can be made in our design. We build the prediction model using the Random Forest algorithm ($n_{trees}=100$, $max_depth=50$), which has been adopted by several previous works to be the most effective algorithm currently available in solving SSD failure prediction. Of the above parameters, n_{trees} represents the trees to be generated when building the model, and max_depth represents the maximum depth of each tree.

The data uses the publicly available dataset from Alibaba [21], which contains about 700,000 SSDs as well as more than 16,000 faulty SSDs. Through training, we are able to obtain the probability that a sample will be identified as a faulty device. As mentioned in Section III-B, by adjusting the probability threshold, we are able to easily implement adjustments to the model in terms of recall and precision. It is important to note that the predictive model is not the focus of our work, so we do not perform optimizations for the model beyond the necessary steps mentioned earlier. Based on the existing setting, we get results similar to most of the previous work [15], [22]. The results are shown in Figure 13. First of all, we can clearly see the “See-saw effect” of these two indicators. It is difficult to reach a higher value at the same time. In order to show the effect of our design, we choose three typical points in the result, A, B and C in the figure. The indicators corresponding to these three points are listed in Figure 13.

These three points represent three different scenarios. Point A represents the traditional prediction application scenario, i.e., it is not desirable to have a situation where the health device is misclassified. Point B, on the other hand, represents the more balanced scenario, where recall and precision are at a relatively high level. And point C is a more aggressive high recall scenario mentioned in our design. Subsequent experiments will be analyzed based on these three points to reveal the superiority of our design.

2) *Group Numeric Modeling*: To evaluate our mechanism’s advantage, we use previous results to calculate the reduction of degradation time. Note that for a successful pre-failure recovery, the performance degradation is subtle, as shown in Section IV-A. Thus in this section, we only consider the post-failure recovery’s degradation time as benefit.

Given prediction’s recall r , the reduced degradation time can be calculated as $t_{reduced} = r \times n_f \times t_d$, where n_f is the total number of failed devices in the whole data center; and t_d is the average degradation time of the post-recovery process. With the above experiment results, we set n_f to 16,000, t_d to 11 minutes or 41 minutes. 16000 is the true number of failed devices in the dataset. And 11 minutes and 41 minutes represent the degradation time with no foreground load running and with foreground load running respectively. For the three points mentioned in Section IV-B1, we compare the benefit they get as well as the overhead. We define the benefit as the reduction in total degradation time, i.e., how long we are able to keep RAID systems out of degradation. This is shown as a bar in the result. The overhead, on the other hand, is the number of invalid backup devices, i.e., we still backed up the device even though it didn’t finally fail. In particular, it should be emphasized that these devices used for backup are not discarded, even though the number of these devices mentioned here is considered as overhead. They still get into the cluster eventually for the next backup need.

As shown in Figure 14, Point B gains a large improvement compared to Point A, with a reduction in degradation time increase of more than 150%. However, the increase in the number of invalid backup devices is only 481, which is almost negligible compared to the overall 700,000 devices and 16,000 failed devices. The more aggressive Point C resulted in a 238% revenue improvement and an overhead of 2,026 invalid backup devices.

3) *Trade Off Evaluation for Performance and Overhead*: As shown in the previous results in Figure 14, although point C seems to bring higher benefits, it also brings much higher overheads. Therefore, this section specifically discusses how to select more appropriate model performance points. In order to show further trends in benefits and overheads, we selected 100 model performance points equally distributed in Figure 13. The overheads they incurred and the benefits

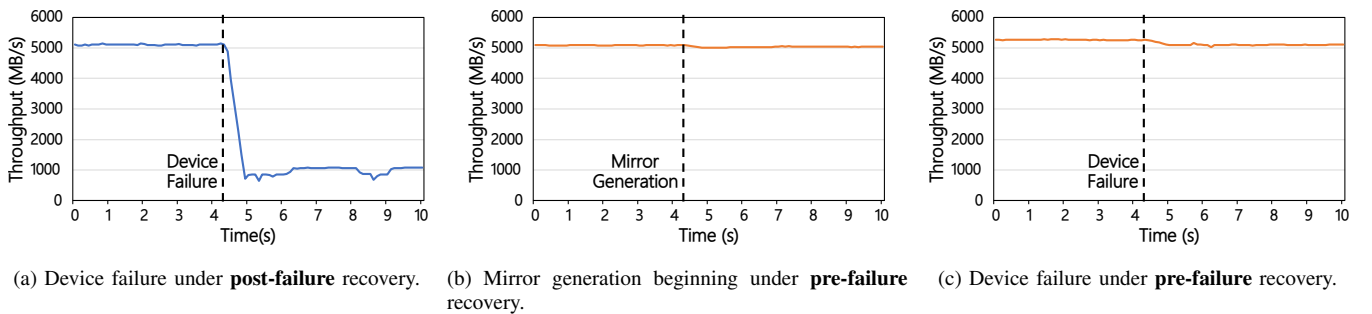


Fig. 12: Transient influence of device failure and mirror generation on 4KB-random-read throughput.

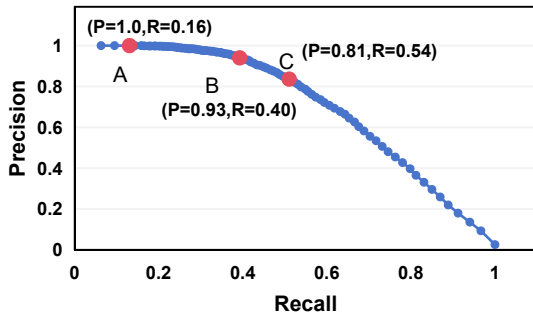


Fig. 13: Trends in precision and recall for different model states. Three points are selected with their precision and recall listed beside.

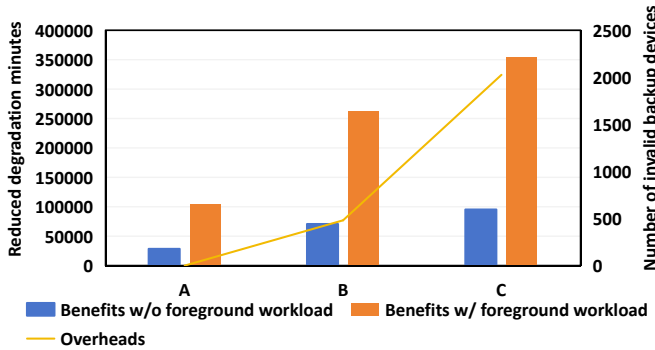


Fig. 14: Comparison of benefits and overheads under group numeric modeling.

improvement compared to point A were also calculated, as shown in Figure 15.

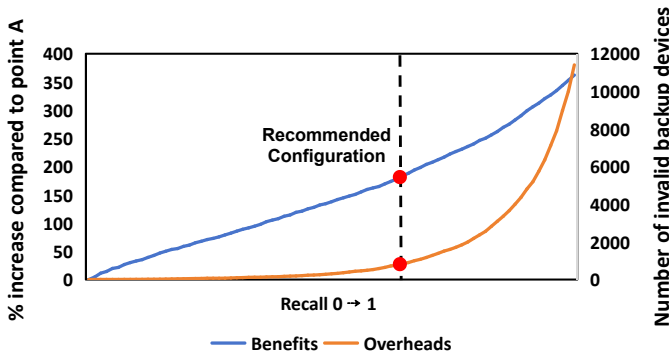


Fig. 15: Comparison of benefits and overheads under group numeric modelling. A recommended configuration point is also given in the figure.

It can be seen that both of them show an upward trend with the increase of recall. However, there is a significant difference in the rate of changes. The further back the curve goes, the faster the overhead grows, eventually even reaching 12,000 invalid backup devices. Figure 4 can explain that, when we adjust the threshold in the hope of identifying all faulty devices, we will inevitably judge some healthy devices as

faulty as well. And as the probability gets lower and lower, more and more healthy samples are gathered, which leads to more and more misclassified healthy samples. Therefore, we recommend keeping the overhead to 5% of the failed devices, in our case as 800 invalidly backed up device numbers, but being able to gain a 184% benefit improvement.

V. RELATED WORKS

A. Storage Device Failure Prediction

Besides the traditional HDDs, flash-based SSDs are also gaining wide adoption as storage devices [23], [24] and they are also error-prone devices [25]. The current definition of failure for storage devices can be classified into the following two directions:

- Fail-stop: The device is not responding and the data in it cannot be accessed [4], [13], [15], [21], [26]–[30].
- Fail-slow: The device can still be accessed, but the access speed becomes extremely low, affecting normal workload operation [31]–[34].

Both of these failures are relatively common in data center scenarios.

1) *Fail-stop*: MVTRF [15] use multi-task learning to simultaneously predict the type of failure and when it will occur through the same model. Chakrabortii et al. [13] proposed the use of a one-classification model to improve the accuracy, applicability, and interpretability of the model, and evaluated the performance of isolated forests. STREAMDFP [26] propose a online failure prediction method with a real-time disk log stream as input. The failure prediction implementation adopted in this paper is a basic method and all works mentioned above can be adopted in our methods with certain modifications.

2) *Fail-slow*: Gunawi et al. [32] explained in detail the types of fail-slow manifestations in real production environments. PERSEUS [33] utilizes a light regression-based model to quickly identify and analyze fails-slow failures at the granularity of the drive. IASO [34] works primarily on timeout signals and converts them into stable and accurate fault-slowness metrics. Since the fail-slow devices also require to be replaced to avoid influencing QoS, our mechanism can also apply to the fail-slow related scenarios.

B. RAID Optimization

Work on RAID in recent years has generally focused on improving RAID performance and reducing recovery time.

1) *Improving RAID Performance*: Many existing works focus on optimizing the normal performance of parity-based RAID. NVM buffers are adopted in [35]–[37] to reduce parity-caused read and write operations on back-end storage device. ElasticRAID [38] and FusionRAID [39] propose to use a replica-parity hybrid mechanism to provide high performance with low space overhead. StRAID [40] improves the scalability and parallelism of the Linux md RAID. Rep-RAID [41] optimizes rubbish collection for RAID-enabled SSDs. Several other works involve optimizing the degraded performance of parity-based RAID. Short Code [42] proposes a RAID6 coding

mechanism that enables better degraded read performance. dRAID [43] optimizes the degraded read under dis-aggregated RAID. Our mechanism focuses on recovery and these works can be used along with our mechanism.

2) *Reduce Recovery Time*: Many works [12], [44], [45] propose to spread contents across the array, involving more devices in the recovery process to accelerate the recovery. Our mechanism can also be used with these works together since they may accelerate both pre-failure recovery and post-failure recovery. SSD-internal RAID recovery mechanism [11] is also proposed to reduce recovery data amount, which is different from our system-level RAID recovery mechanism.

VI. CONCLUSION

In this paper, we propose to use pre-failure recovery combined with failure prediction technology instead of post-failure recovery to reduce the degradation time of RAID systems. A challenge is that current prediction methods' emphasis on precision and neglect of recall limits the pre-failure recovery's reduction in total degradation time. To solve this problem, we propose a two-stage backup mechanism to enable the improvement of recall in prediction while avoiding replacing misclassified healthy storage devices. Exceptional cases are also discussed to prove the robustness of the proposed method. The evaluation shows that the two-stage backup mechanism can have a significant reduction in degradation time in the data center.

REFERENCES

- [1] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988, pp. 109–116.
- [2] R. Wang, Y. Li, H. Xie, Y. Xu, and J. C. Lui, "Graphwalker: An i/o-efficient and resource-friendly graph analytic system for fast and scalable random walks," in *2020 USENIX Annual Technical Conference*, 2020, pp. 559–571.
- [3] J. Canny, H. Zhao, B. Jaros, Y. Chen, and J. Mao, "Machine learning at the limit," in *2015 IEEE International Conference on Big Data*, 2015, pp. 233–242.
- [4] S. Maneas, K. Mahdavian, T. Emami, and B. Schroeder, "A study of ssd reliability in large scale enterprise storage deployments," in *18th USENIX Conference on File and Storage Technologies*, 2020, pp. 137–149.
- [5] R. Kesavan, J. Hennessey, R. Jernigan, P. Macko, K. A. Smith, D. Tennant, and V. Bharadwaj, "Flexgroup volumes: A distributed waf file system," in *2019 USENIX Annual Technical Conference*, 2019, pp. 135–148.
- [6] J. Li, P. Li, R. J. Stones, G. Wang, Z. Li, and X. Liu, "Reliability equations for cloud storage systems with proactive fault tolerance," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, pp. 782–794, 2018.
- [7] M. H. Tong, R. L. Grossman, and H. S. Gunawi, "Experiences in managing the performance and reliability of a large-scale genomics cloud platform," in *2021 USENIX Annual Technical Conference*, 2021, pp. 973–988.
- [8] S. Wu, H. Li, B. Mao, X. Chen, and K.-C. Li, "Overcome the gc-induced performance variability in ssd-based raids with request redirection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 822–833, 2018.
- [9] Y. Kim, J. Lee, S. Oral, D. A. Dillow, F. Wang, and G. M. Shipman, "Coordinating garbage collection for arrays of solid-state drives," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 888–901, 2012.

- [10] M. Hao, G. Soundararajan, D. Kenchammana-Hosekote, A. A. Chien, and H. S. Gunawi, "The tail at store: A revelation from millions of hours of disk and ssd deployments," in *14th USENIX Conference on File and Storage Technologies*, 2016, pp. 263–276.
- [11] D. Hong, K. Ha, M. Ko, M. Chun, Y. Kim, S. Lee, and J. Kim, "Reparo: A fast raid recovery scheme for ultra-large ssds," *ACM Transactions on Storage*, vol. 17, no. 3, pp. 1–24, 2021.
- [12] G. Zhang, Z. Huang, X. Ma, S. Yang, Z. Wang, and W. Zheng, "Raid+: Deterministic and balanced data distribution for large disk enclosures," in *16th USENIX Conference on File and Storage Technologies*, 2018, pp. 279–294.
- [13] C. Chakrabortii and H. Litz, "Improving the accuracy, adaptability, and interpretability of ssd failure prediction models," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 120–133.
- [14] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, "Disk failure prediction in data centers via online learning," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.
- [15] Y. Zhang, W. Hao, B. Niu, K. Liu, S. Wang, N. Liu, X. He, Y. Gwon, and C. Koh, "Multi-view feature-based ssd failure prediction: What, when, and why," in *21st USENIX Conference on File and Storage Technologies*, 2023, pp. 409–424.
- [16] S. Lu, B. Luo, T. Patel, Y. Yao, D. Tiwari, and W. Shi, "Making disk failure predictions smarter!" in *18th USENIX Conference on File and Storage Technologies*, 2020, pp. 151–167.
- [17] "Wikipedia. S.M.A.R.T." [https://en.wikipedia.org/wiki/S.M.A.R.T.](https://en.wikipedia.org/wiki/S.M.A.R.T)
- [18] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
- [19] "Linux. Linux raid." [https://raid.wiki.kernel.org/index.php/Linux_Raid.](https://raid.wiki.kernel.org/index.php/Linux_Raid)
- [20] "Jens Axboe. FIO: Flexible I/O Tester." [https://github.com/axboe/fio.](https://github.com/axboe/fio)
- [21] S. Han, P. P. Lee, F. Xu, Y. Liu, C. He, and J. Liu, "An in-depth study of correlated failures in production ssd-base data centers," in *19th USENIX Conference on File and Storage Technologies*, 2021, pp. 417–429.
- [22] F. Xu, S. Han, P. P. Lee, Y. Liu, C. He, and J. Liu, "General feature selection for failure prediction in large-scale ssd deployment," in *51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2021, pp. 263–270.
- [23] L. Shi, J. Li, C. J. Xue, C. Yang, and X. Zhou, "Exlru: a unified write buffer cache management for flash memory," in *Proceedings of the Ninth ACM International Conference on Embedded Software*, 2011, p. 339–348.
- [24] C. Gao, L. Shi, C. Ji, Y. Di, K. Wu, C. J. Xue, and E. H.-M. Sha, "Exploiting parallelism for access conflict minimization in flash-based solid state drives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 168–181, 2018.
- [25] L. Shi, Y. Di, M. Zhao, C. J. Xue, K. Wu, and E. H.-M. Sha, "Exploiting process variation for write performance improvement on nand flash memory storage systems," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, no. 1, pp. 334–337, 2016.
- [26] S. Han, P. P. Lee, Z. Shen, C. He, Y. Liu, and T. Huang, "Toward adaptive disk failure prediction via stream mining," in *2020 IEEE 40th International Conference on Distributed Computing Systems*, 2020, pp. 628–638.
- [27] E. Xu, M. Zheng, F. Qin, Y. Xu, and J. Wu, "Lessons and actions: What we learned from 10k ssd-related storage system failures," in *USENIX Annual Technical Conference*, 2019, pp. 961–976.
- [28] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A large-scale study of flash memory failures in the field," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 177–190, 2015.
- [29] B. Schroeder, R. Lagisetty, and A. Merchant, "Flash reliability in production: The expected and the unexpected," in *14th USENIX Conference on File and Storage Technologies*, 2016, pp. 67–80.
- [30] I. Narayanan, D. Wang, M. Jeon, B. Sharma, L. Caulfield, A. Sivasubramaniam, B. Cutler, J. Liu, B. Khessib, and K. Vaid, "Ssd failures in datacenters: What? when? and why?" in *Proceedings of the 9th ACM International on Systems and Storage Conference*, 2016, pp. 1–11.
- [31] R. Lu, E. Xu, Y. Zhang, F. Zhu, Z. Zhu, M. Wang, Z. Zhu, G. Xue, J. Shu, M. Li *et al.*, "From missteps to milestones: A journey to practical fail-slow detection," *ACM Transactions on Storage*, vol. 19, no. 4, pp. 1–28, 2023.
- [32] H. S. Gunawi, R. O. Suminto, R. Sears, C. Gollhofer, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey *et al.*, "Fail-slow at scale: Evidence of hardware performance faults in large production systems," *ACM Transactions on Storage*, vol. 14, no. 3, pp. 1–26, 2018.

- [33] R. Lu, E. Xu, Y. Zhang, F. Zhu, Z. Zhu, M. Wang, Z. Zhu, G. Xue, J. Shu, M. Li *et al.*, “Perseus: A fail-slow detection framework for cloud storage systems,” in *21st USENIX Conference on File and Storage Technologies*, 2023, pp. 49–64.
- [34] B. Panda, D. Srinivasan, H. Ke, K. Gupta, V. Khot, and H. S. Gunawi, “Iaso: A fail-slow detection and mitigation framework for distributed storage services,” in *2019 USENIX Annual Technical Conference*, 2019, pp. 47–62.
- [35] J. Colgrove, J. D. Davis, J. Hayes, E. L. Miller, C. Sandvig, R. Sears, A. Tamches, N. Vachharajani, and F. Wang, “Purity: Building fast, highly-available enterprise flash storage from commodity components,” 2015, p. 1683–1694.
- [36] S. Im and D. Shin, “Flash-aware raid techniques for dependable and high-performance flash memory ssd,” *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 80–92, 2010.
- [37] C.-C. Chung and H.-H. Hsu, “Partial parity cache and data cache management method to improve the performance of an ssd-based raid,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 22, no. 7, pp. 1470–1480, 2013.
- [38] Z. Gu, J. Li, Y. Peng, Y. Liu, and T. Zhang, “Elastic raid: Implementing raid over ssds with built-in transparent compression,” in *ACM International Conference on Systems and Storage*, ser. SYSTOR ’23, 2023, p. 83–93.
- [39] T. Jiang, G. Zhang, Z. Huang, X. Ma, J. Wei, Z. Li, and W. Zheng, “Fusionraid: Achieving consistent low latency for commodity ssd arrays,” in *19th USENIX Conference on File and Storage Technologies*, 2021, pp. 355–370.
- [40] S. Wang, Q. Cao, Z. Lu, H. Jiang, J. Yao, and Y. Dong, “Straid: Stripe-threaded architecture for parity-based raids with ultra-fast ssds,” in *2022 USENIX Annual Technical Conference*, 2022, pp. 915–932.
- [41] J. Li, B. Gerofi, F. Trahay, Z. Cai, and J. Liao, “Rep-raid: An integrated approach to optimizing data replication and garbage collection in raid-enabled ssds,” in *Proceedings of the 24th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, 2023, pp. 99–110.
- [42] Y. Fu, J. Shu, X. Luo, Z. Shen, and Q. Hu, “Short code: An efficient raid-6 mds code for optimizing degraded reads and partial stripe writes,” *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 127–137, 2017.
- [43] J. Shu, R. Zhu, Y. Ma, G. Huang, H. Mei, X. Liu, and X. Jin, “Disaggregated raid storage in modern datacenters,” in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Volume 3, ser. ASPLOS 2023, 2023, p. 147–163.
- [44] G. Alvarez, W. Burkhard, L. Stockmeyer, and F. Cristian, “Declustered disk array architectures with optimal and near-optimal parallelism,” in *Proceedings. 25th Annual International Symposium on Computer Architecture*, 1998, pp. 109–120.
- [45] X. Luo, J. Shu, and Y. Zhao, “Shifted element arrangement in mirror disk arrays for high data availability during reconstruction,” in *2012 41st International Conference on Parallel Processing*, 2012, pp. 178–188.