Be SMART, Save I/O: Probabilistic Approach to Avoid Uncorrectable Errors in Storage Systems

Md Arifuzzaman*, Masudul Bhuiyan†, Mehmet Gumus*, and Engin Arslan*
* University of Nevada, Reno, USA
{marifuzzaman, mgumus, earslan}@unr.edu
†CISPA, Germany
masudul.bhuiyan@cispa.de

***Abstract*—Silent data corruption poses a significant risk to the integrity of data in storage systems. Although error correction codes (ECC) can recover the majority of such errors, a non-negligible portion of them escape ECC, referred as uncorrectable errors (UEs). Despite being rare in nature, increasing scale of storage systems and fast-growing I/O rates decreased the mean time between UEs from months to hours. Yet, unlike disk failures, UEs are hard to predict with high precision, making it difficult to adopt proactive measures. In this paper, we introduce a probabilistic approach to deploy UE mitigation strategies that can capture significant portion of UE while keeping the system overhead at a tolerable range. To achieve this, we first estimate the probability of I/O operations to be exposed to UEs and find a minimum subset of disks for which employing UE avoidance strategies can lead to significant decrease in UE exposure. We demonstrate through extensive simulations that when the proposed probabilistic model is used to implement *write verification* strategy to detect and recover from UEs, more than 50% of all write-triggered UEs can be avoided with 1% read overhead, and more than 70% of UEs can be mitigated with less than 3.5% read overhead. We further measure the impact of incurred read overhead on write performance in production Lustre and GPFS file systems and validate our findings that more than half of UEs can be avoided while degrading write I/O throughout by less than 0.9%.**

I. Introduction

Advancements in computing and sensing technologies paved the way for many science applications to generate a massive volumes of data. For example, Hardware/Hybrid Accelerated Cosmology Code is an extreme-scale cosmology simulation that produces 20 petabyte data in a single particle simulation [1]. To accommodate the increasing storage needs of scientific applications, the capacity of storage systems have seen a steady increase over the years. As an example, while the fastest supercomputer in 2010, Jaguar, had 10 petabyte storage capacity, the fastest supercomputer in 2022, Frontier, has around 700 petabyte storage capacity [2]. Increasing capacity in turn leads to frequent observation of otherwise rare silent errors (e.g., dropped and off-track writes) which could lead to permanent data loss and service interruptions [3].

Although most disk drives utilize error correcting codes (ECC) to detect and recover from silent errors, a non-negligible portion of them happen in a way that ECC cannot correct, referred as uncorrectable errors (UEs) or latent sector errors [3], [4], [5]. They are typically caused by undetected write errors or media imperfections and if not handled timely may result in complete data loss even with parity-based RAID arrays [3]. Previous studies showed that UEs take place once in every 10^{12} (125GB) to 10^{15} (125TB) bits of I/O operation [6], [7] and affect up to 25% of all hard disks in datacenters [8]. As today's high performance computing clusters and commercial datacenters host thousands of disks and handle hundreds of terabytes of I/O workload daily [9], UEs pose a significant risk to the reliability of file systems.

Since ECC falls short identify and recover from many types of undetected write errors, most modern file systems rely on checksumming, which computes and stores a unique hash value for each data block (or set of blocks) separate from data block [10]. However, Krioukov et al. showed that checksumming can fail to guarantee protections against complex failures such as lost writes and misdirected writes [5]. Although the use of file system level checksumming together with parity or mirror-enabled disk arrays (e.g., RAID-Z) can help to detect and recover from many types of UEs through RAID reconstruction, the cost of recovery could significantly deteriorate I/O response time [11], [12]. Even worse, recovery procedures may occasionally lead to partial or full system outages. As an example, a previous study found that 46% of all system-wide outages in BlueWaters [13] supercomputer is caused by file system issues as existing fail-over strategies fall short to handle all error types properly [14]. In another instance, data corruption in an `inode` block of a Lustre file system has led to 13 days of complete service interruption in an HPC cluster [15]. Therefore, we argue that an ability to predict UEs before they occur will improve the efficiency and reliability of file systems as proactive measures can be taken to mitigate them before they cause catastrophic failures.

To achieve this goal, we first train machine learning models to estimate the probability of UEs ahead of time using historical Self-Monitoring, Analysis and Reporting Technology (SMART) [16] which report various health statistics for disk (and SSD) drives such as current device temperature, and reallocated sector count [17]. We find that the machine learning models achieve up to 98% accuracy in predicting the occurrence of UEs a day in advance. Yet, they cause relatively high (up to 11% for 90% true positive rate) false positive rate (i.e., incorrect error estimations), which is a major impediment in the adoption of proactive measures (e.g., data scrubbing and write verification) due to requiring to check too many healthy

disks along with UE-prone ones. As an example, 3% false positive rate requires checking 300 healthy disks (i.e., no UE exposure) along with 5 unhealthy ones in a storage system with $10,000$ disks. To overcome this issue, we introduce a novel *probabilistic model* to quantify UE probability for I/O operations, then determine a minimum set of disks to deploy UE mitigation strategies to lower the risk to a tolerable range. As an example, write verification (aka read after write) can be used to detect and correct write-triggered UEs by reading data back from disk for verification purposes, however it induces significant overhead to storage systems in addition to slowing down write performance considerably. We demonstrate that the proposed probabilistic model can be used to identify a subset of disks for which enabling write verification can detect and mitigate the majority of UEs while keeping the cost on the file system at minimum. Our extensive simulation results show that the use write verification probabilistically leads to detection of more than half of all write-triggered UEs by verifying only less than 1% of all write operations. *Hence, we believe that the probabilistic model makes an important contribution to the field by paving the way for the deployment of error mitigation strategies despite low precision of prediction models, thereby enhancing the reliability of storage systems.* In summary, this paper makes following contributions:

- We process 143 million SMART logs from 106 thousand disks to analyze the characteristics of UEs and train prediction models. We find that eXtreme Gradient Boosting (XGBoost) classifiers attains the best performance by correctly identifying more than 90% of all UEs with less than 2% false positive rate.
- We introduce a probabilistic model to capture and recover from UEs by finding a minimum set of disks to take precautionary measures such as checking the correctness of write operations.
- We evaluate the performance of probabilistic model through extensive simulations and observe that when it is used to verify file write operations, more than half of all write-triggered UEs can be captured by verifying less than 1% write operations. It can also capture more than 70% of UEs with less than 3.5% read overhead.
- We evaluate the performance impact of the probabilistic write verification model on four production HPC clusters and show that more than half of UEs can be captured with negligible (less than 0.9%) impact on write throughput. Finally, we discuss possible deployment challenges and lay out potential solutions.

II. Related Work

Most previous work on error detection in file systems focus on predicting *complete* disk failures [18], [19], [20], [21], [22], [23]. For example, Lu et al. developed machine learning models to process historical SMART logs and performance statistics and predict the disk failures [21]. Xiao et al. employed online learning to disk failure predictions to adapt the prediction models to evolving nature of disk statistics and workload characteristics [24]. Han et al. further extends the online learning by also updating the model type as change in data may cause performance degradation for the original model (e.g., Online Random Forest) [20].

In the area of *partial* disk failures, Bairavasundaram et al. analyzed Latent Sector Errors (LSEs), commonly caused by uncorrectable errors, for 1.53M hard disks over a period of 32 months [25] and found that 3.25% of all disks developed at least one LSE. Moreover, recent studies based on Facebook and Google datacenters statistics reported that a significant portion of solid state drives ($20-57\%$) developed at least one LSE during their lifetime [26], [27]. This, in turn, poses data loss risk as Hafner at al. showed that undetected errors when followed by disk failures may result in complete data loss even in parity based RAID arrays [3]. Although combining file system-level checksumming with parity or mirror-enabled disk arrays can find and fix most undetected write errors through disk scrubbing [28], it can have adverse impact on system performance due to requiring to recreate inaccessible data blocks.

The most relevant work to our study is conducted by Mahdisoltani et al. in which the authors proposed Random Forest models to detect reallocated sector errors and uncorrectable errors using historical SMART logs [8]. Our work differs from [8] in two ways: First, we find that Extreme Gradient Boosting (XGB) model attains slightly better performance (around 2% higher true positive rate 3% less false negative rate) compared to Random Forest. Second and more importantly, [8] proposed adaptive disk scrubbing solution to scrub file system at an accelerated rate to find and recover from errors quicker. On the other hand, we introduce a novel probabilistic model which first quantifies the likelihood of a given workload to be exposed to a UE, then finds a minimum set of disks to deploy mitigation strategies (e.g., write verification and write redirection) if the predicted error rate is not tolerable for the application. Unlike file system scrubbing which adversely affects all application alike, the probabilistic model offers an application/workload-specific solution such that it can be selectively used based on workload/application requirements to avoid incurring system-wide overhead. Moreover, adaptive disk scrubbing can only be applicable for small clusters for which the mean time between errors is in the order of several days or weeks to limit its impact. Finally, while adjusting the rate of disk scrubbing helps to reduce mean time to detection (MTTD) by $1.3\times - 1.7\times$, its UE detection frequency is still in the order of days. In contrast, when probabilistic model is used to implement write verification measure, it can detect UE as soon as file write is completed, reducing MTTD to seconds to minutes.

III. Modeling Uncorrectable Errors

To gain insights into UEs, we analyze publicly available Backblaze dataset that contains 270 million SMART logs 200 thousand disks collected over the period of 93 months, from January 2014 to September 2021 [29]. SMART logs are populated once a day by each disk to report several operational statistics, such as temperature and hours-on-power as shown

TABLE I
OVERVIEW OF DISK MODELS USED IN THIS WORK.

Disk model	Capacity	Total Disks	Ratio of Disks with Uncorrectable Errors
ST12000NM0007	12 TB	38,797	5.08%
ST4000DM000	4 TB	37,038	10.29%
ST8000NM0055	8 TB	15,161	6.86%
ST8000DM002	8 TB	10,283	6.75%
ST3000DM001	3 TB	4,708	36.81%

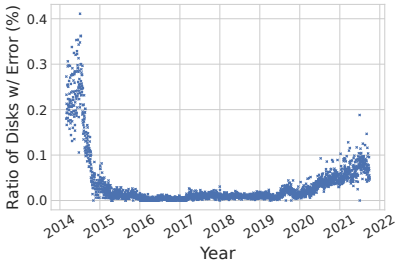


Fig. 1. The ratio of disks with uncorrectable errors is around 0.02% during most of observation duration but reached to 0.3% in 2014 and 0.1% in late 2021, affecting $5-36\%$ of all disks.

in Table II. Although the dataset contains logs from more than 80 disk models, we excluded the ones with less than thousand disks to avoid misleading results. This filtering left a total of 106 thousand disks from five disk models whose details are given in Table I. The disk models ST12000NM0007 and ST4000BM000 constitute 71.55% of all disks as there are more than $37K$ disks from each model. Despite having the similar number of disks, $2x$ more disks developed UEs in ST4000DM000 disk model compared to ST12000NM0007. Moreover, while the ratio of disks with UEs is less than 10.3% for four disk models, it reaches to 36.8% for ST3000DM001, indicating a significant variation between the disk models.

We find that $9.25K$ of $106K$ (8.7%) disks reported at least one UE within the observation period. Figure 1 presents the ratio of disks that reported new UEs in a daily basis. The ratio was as high as 0.43% in July 2014, but later stabilized at around 0.02% after disk model ST3000DM001 was removed from the system. We observe another increase in UE rate in last 24 months as the rate approached to 0.1%. Given that petabyte scale file systems are composed of thousands of disks, these seemingly small values correspond to several occurrence of UEs every day. For example, Frontier supercomputer consists of $47,700$ hard disk drives which would result in $9-10$ UEs on a daily basis for a conservative UE rate of 0.02%. It is therefore important to predict UEs before they happen and take precautionary steps to prevent catastrophic failures and protect sensitive data against irreparable corruptions.

To achieve this, we first investigate the possibility of using SMART reports to estimate the occurrences of UEs ahead of time. We apply Random Forest to assess the relationship between previous day's SMART metrics and the value of UE (1 if new UE is reported, 0 otherwise). We find that 13 features (listed in Table II) can explain 99% variance for UE, indicating a strong correlation between UE incidents and

TABLE II
SMART METRICS USED TO TRAIN THE PREDICTION MODELS

SMART ID	Attribute Name	Importance Score
7	Seek Error Rate	0.1711
9	Power-On Hours	0.1633
1	Read Error Rate	0.1346
193	Load Cycle Count	0.1208
187	Uncorrectable Error	0.1176
5	Reallocated Sectors Count	0.0953
194	Temperature	0.0557
4	Start/Stop Count	0.0367
12	Power Cycle Count	0.0357
198	Uncorrectable Sector Count	0.0286
197	Current Pending Sector Count	0.0262
188	Command Timeout	0.0098
199	UltraDMA CRC Error Count	0.0046

previous day's SMART metrics. Note that the attribute #187 in selected features refers to total number of UEs that a disk has reported until day i, which we use to predict the value of feature #187 for day $i+1$. High importance score for #187 suggests that the probability of new UEs is correlated to the number of UEs that a disk has reported previously. In fact, we find that 92% of all disks with at least one UE developed more than one UE in their lifetime.

Since the daily number of disks with UEs constitutes around 0.02% of all disks, there is nearly $1:5000$ imbalance between positive (i.e., error) and negative (i.e., no error) samples, an issue that adversely affects the performance of machine learning models. The typical approach to balance multiclass datasets involves resampling the minority or majority classes through undersampling or oversampling. Oversampling duplicates the observations of the minority class and undersampling drops the observations of the majority class to obtain a balanced dataset. We chose to undersample SMART logs with negative label as it significantly shortens training time and reduces false positive rate compared to oversampling. We experimented with several undersampling methods such as Cluster Centroids, Condensed Nearest Neighbour, Edited Nearest Neighbours, One Sided Selection, and Random Undersampler. We observe that they perform similarly, so we pick Random Undersampler due to its simplicity. We also applied various undersampling rates (e.g., 1:10, 1:5, 1:3, 1:1) but did not observe any difference, so settled with $1:1$ sampling ratio. Finally, we trained four machine learning models that are commonly used for classification problems, Support Vector Classifier (SVC), Decision Tree (DT), Random Forest (RF), eXtreme Gradient Boosting (XGB). We also tested Deep Neural Network (DNN) and CNN-LSTM models to evaluate the performance of neural network models as they yield competitive results for disk error prediction problems using SMART logs [21].

Since hyperparameters have significant impact on the performance of these models, we used *scikit-optimize* library – with Gaussian-based Bayesian optimization – to independently discover the optimal values for each machine learning and disk models combinations. We used Neural Architecture Search (NAS) [30] to find an optimal architecture for the DNN model. In the training phase, we mark SMART logs with a positive

TABLE III
THE PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS IN PREDICTION UNCORRECTABLE ERRORS. EXTREME GRADIENT BOOSTING YIELDS THE BEST PERFORMANCE IN OVERALL WITH ALMOST 95% AUC SCORE FOR ALL DISK MODELS EXCEPT ST3000DM001.

Models	DT	CNN-LSTM [21]	DNN	SVC	RF [8]	RF	XGB
ST12000NM0007	90.40	95.41	96.90	97.93	97.85	98.19	98.25
ST4000DM000	81.68	93.79	94.17	93.98	93.81	94.45	94.70
ST8000NM0055	81.94	90.08	91.82	94.81	94.32	94.87	95.57
ST8000DM002	83.90	92.20	93.85	95.87	95.17	95.47	96.05
ST3000DM001	76.80	88.13	91.30	88.71	91.83	92.63	91.78

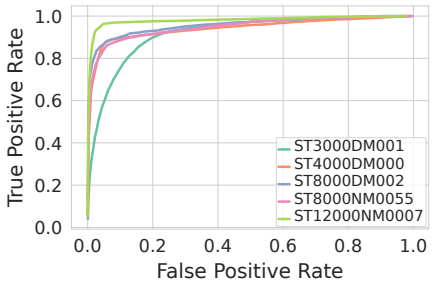


Fig. 2. ROC curve for different disk models using XGBoost classifier.

label if corresponding disk reported at least one UE in the following day (i.e., increase in value of #187 in SMART log), and with a negative label otherwise. The performance of models is then evaluated based on the accuracy in predicting UEs using SMART attributes of previous day in unlabeled data (i.e., test dataset). Rather than randomly splitting SMART logs for training and test datasets, we adopted time-based partitioning to avoid data leakage. As an example, SMART logs from Nov 2014 to Dec 2019 are used for training and SMART logs from Jan-March of 2020 are used for testing. We repeated this five times using different training and test splits and report the average results.

Model Evaluation: To compare the performance of different machine learning models, we calculate True Positive Rate (TPR) and False Positive Rate (FPR) values. TPR indicates the portion of positive samples in test data that is correctly identified by a model. In this paper, it represents the percentage of disks with UEs (i.e., positive class) that the prediction models are able to identify correctly. FPR, on the other hand, refers to the portion of actual negative cases that is misidentified as positive cases by a model. It represents the percentage of disks without UEs that is *incorrectly* classified as disks with uncorrectable error. Even though the default threshold of 0.5 is used to determine the class of a sample in binary classifier (e.g., a sample with 0.51 error probability is marked with error label), one change the threshold to find a balance between TPR and FPR values. Receiver operating characteristic (ROC) curve then presents TPR and FPR values for different classification threshold. As an example, Figure 2 presents ROC curve for different disk models using XGB classifier. Finally, Area Under Curve (AUC) is used to combine TPR and FPR values into a single metric by calculating the area under ROC curve.

Table III presents AUC scores of machine learning models for different disk models. RF [8] presents the performance of Random Forest model that is used to predict latent sector errors using SMART logs. It is clear that most models yield competitive results with over 90% AUC score. Despite being significantly costly to train, the neural network models (CNN-LSTM and DNN) always fall behind RF and XGB models. The RF model described in [8] also perform slightly lower than the RF model we trained, which demonstrates the benefit of automated hyperparameter tuning. In addition to performance variation of the machine learning models, disks also exhibit discrepancy in prediction performances. For example, all prediction models return the lowest AUC score for ST3000DM001. Similarly, the models attain the best AUC score for ST12000NM0007.

Despite yielding high AUC scores, the prediction models results in relatively high false positive rate (i.e., low precision). As an example, XGB model results in $2-11\%$ false positive rate to attain 90% TPR. This in turn impedes the adoption of common mitigation strategies such as disk removal due to high overhead. To put this into a perspective, 5% FPR would require retiring 300 healthy disks along with $2-3$ unhealthy (i.e., exposed to UE) ones in a cluster with $10,000$ disks with 0.02% average UE probability. Even less intrusive mitigation strategies such as disk scrubbing and write verification can have significant impact on system/application performance due to checking too many disks. We therefore introduce a probabilistic model to to capture as many UEs as possible while keeping system overhead at minimum. To do so, we utilize the prediction models to quantify the probability of a system/application to be exposed to a UE, then find a minimum set of disks to apply preventive measures if the calculated UE rate is higher than a tolerable value. Please note that while we demonstrate the effectiveness of the probabilistic model in the implementation of write verification-based UE mitigation strategy at the application level in § V, it can be used to implement other UE avoidance strategies such as I/O redirection. Hence, we believe that the probabilistic model makes a novel contribution to the field by paving the way for the adoption of ML models in production systems even with relatively low precision performance. Next, we present the probabilistic model and show its application to detect and recover from write I/O-triggered UEs.

IV. The Probabilistic Model

In the absence of predictive models, one can estimate the probability of UEs using Uncorrectable Bit Error Rate ($UBER$), which defines the probability of UE for unit (i.e., single bit) operation. Although this information is not released by disk manufacturers, previous studies estimate it to be around 10^{-15} ($125TB$) for enterprise disks and flash drives [6], [7]. Hence, the probability a workload W that issues a total of b bit data to N ($N \geq 1$) disks (due to RAID or file system striping) to be exposed to a UE can be estimated by

$$E(W) = 1 - \prod_{i=1}^{N}(1 - b_i \cdot UBER) \tag{1}$$

where b_i refers data size (in bits) written to disk i. Equation 1 can be approximated to

$$E(W) \approx \sum_{i=1}^{N} b_i \cdot UBER = b \cdot UBER \tag{2}$$

when $UBER$ is $\approx 10^{-15}$ and N is small (e.g., $< 1,000$). Note that the number of disks and the distribution of workload to disks has no impact in this estimation since the value of $UBER$ is the same for all disks when no other information is available. Next, we calculate the UE probability using the *XGBoost* model that is trained with historical SMART logs as described in § III. Specifically, the XGBoost model uses previous day's SMART report of a disk to predict the likelihood of new UEs to take place in the disk within next day[1]. Let p_i be the probability of i^{th} disk to develop an UE as estimated by the XGBoost model for $i = 1, \ldots, N$, b_i be the data size that is issued to disk i, and D_i is the total I/O size that the disk i handles on a given day, where $b_i \leq D_i$. Then, the probability of the workload W to be exposed to a UE on disk i becomes

$$E(W_i)_M = b_i \frac{p_i}{D_i} = P_i \tag{3}$$

Since p_i is the probability of exactly one UE to happen in a day during which a total of D_i bits are processed, $\frac{p_i}{D_i}$ is analogous to $UBER$ as it represents the probability of a UE to affect a unit write operation (i.e., one bit). We then multiply this by write I/O size the workload W issues (b_i) to find the probability of the workload W to be exposed to a UE on disk i, referred as P_i. Therefore, the probability of at least one UE for W can be estimated by

$$E(W)_M = 1 - \prod_{i=1}^{N}(1 - P_i) \approx \sum_{i=1}^{N} P_i \tag{4}$$

Note that even $E(W)_M$ will be different than $E(W)$ in most cases since individual workloads only write to a small set of disks and UE rate for majority of disks is much smaller than average UE rate $UBER$. We can thus exploit an ability to compute UE probability at an higher precision that model-agnostic approach (based on $UBER$) to decide when to activate UE mitigation strategies. As an example, one can set a threshold for error probability to activate write verification such that it will only be used if the estimated probability is higher than the threshold, thereby reducing the system overhead. Since it can be challenging to find the right value for the threshold, we use an improvement ratio over $UBER$-based error prediction as a metric to determine when to activate mitigation strategies, which can be calculated by

$$\underbrace{k}_{\textit{improvement ratio}} = \frac{E(W)}{E(W)_M} \tag{5}$$

For example, if an application/system demands UE probability to be decreased by $100\times$ over base UE calculation (i.e., $E(W)_M \leq \frac{E(W)}{100}$), then the probabilistic model will search for a set of disks to apply preventive actions such that the predicted error probability, $E(W)_M$, for the *remaining disks* will lead to *improvement ratio* of 99% or more ($k = 100 * \frac{X - 0.01 * X}{X}$). Note that the improvement ratio can be negative if the XGBoost model predicts that the selected disks have higher UE probability than $UBER$. Equation (4) can further be extended to calculate the probability of developing more than one UE, but we focus on at least one error probability calculation in the rest of the paper and leave the analysis and evaluation of more than one error scenario as a future work.

V. Probabilistic Write Verification

We next demonstrate the implementation of the probabilistic model to mitigate UEs that are caused by undetected write errors (UWE) as previous studies find that UWEs are among the major reasons of UEs both for enterprise disks and flash drives [6], [7]. UWEs can alter multiple bits at unintended disk sectors [25], [3], [31], [6], causing corrupted data to be stored in disks. Although Data Integrity Field has been proposed (aka T10 DIF) to improve the integrity of write operations, it is not widely available due to hardware support requirement. Moreover, some types of UWEs such as dropped writes can still take place even if T10 DIF is implemented [3]. Therefore, additional preemptive measure can be implemented to protect sensitive write operations against UWEs thereby avoiding UE to go undetected.

One potential way to detect and recover from UWEs is reading data back from disk after completing the write operation to check for UEs, similar to read-after-write (RAW) that some disk models support to detect and fix the errors [32], [33]. Since disk-level RAW does not cover end-to-end path and is not widely available in all storage systems, it can be implemented in the application level to detect and fix UWEs. As an example, some file transfer services (e.g., Globus [34], Shift [35], and XRootD [36]) support end-to-end integrity verification to avoid silent errors that might happen while transmitting data in the network or writing it to storage. However, using application-layer verification as a default solution

[1]Since SMART metrics are populated once a day, the predictions made by the XGBoost model are used for all estimations made until new SMART metrics published the following day.

for all write operations increases load on storage systems and degrades I/O performance. Hence, we propose *probabilistic write verification* to exercise write verification strategy only when the predicted UE rate is higher than a tolerable range. For example, if an I/O operation is split into two disks with 10^{-13} and 10^{-17} UE probabilities, then we may want to verify the write operations issued to the first disk as its error probability is $100\times$ higher than average UE rate of 10^{-15}. By doing so, we can capture and recover from potential UEs on the first disk which reduces the effective UE rate for the I/O operation by $10,000$ times compared to initial UE probability. Although conducting write-verification for both disks will eliminate UEs completely, it comes at the cost of increased overhead on storage systems. Thus, the probabilistic write verification aims to find a sweet spot between error avoidance and associated costs.

Let S be the set of disks used to handle a write workload W using N disks, $S = \{1, \ldots, N\}$ and $E_{S_c}(W)_M$ be UE probability when write verification is applied on a subset of disks, $S_c \subset S$, which can be calculated by

$$E_{S_c}(W)_M = \sum_{i \in S \setminus S_c}^{N} b_i \frac{p_i}{D_i} \tag{6}$$

We can then find a subset of S that satisfies the improvement ratio requirement for the UE rate by iterating over all subsets of S. Since multiple subsets of S can meet the requirement, we define a cost function to find the one that minimizes the overhead (i.e., amount of data that write verification is applied) while lowering overall UE rate as

write-verified I/O size *UE probability*

$$C(S_c) = \alpha \sum_{i \in S_c} b_i + (1-\alpha) \sum_{i \in S \setminus S_c} b_i \frac{p_i}{D_i} \tag{7}$$

where $0 \leq \alpha \leq 1$. The cost function consists of two parts as total data size to apply write verification, $\sum_{i \in S_c} b_i$, and the predicted UE rate, $\sum_{i \in S \setminus S_c}$ for the remaining portion of the data. α is then used to strike a balance between the overhead and the improvement ratio. When α is set to 1, the cost function will search for a subset S that requires the lowest amount of data to be write-verified while meeting the improvement ratio. On the other hand, when α is equal to 0, the cost function will find a subset of disks for which enabling write verification will lead to the lowest UE rate. Note that although we only consider subsets of S that satisfy the desired improvement factor requirement, $\alpha < 1$ gives us an opportunity to reduce the error probability further if an increase in the overhead is negligible. For instance, one can prefer 5% write-verification overhead (5% of all writes are verified) over 3% overhead even if both satisfies the improvement ratio requirement but the former results in significantly higher (e.g., $2x$ more) improvement ratio.

A straightforward approach to find a subset of disks with minimum cost would involve iterating over all subsets of S, i.e., ***brute-force*** search. However, it is not be a feasible solution especially when the number of disks is high due to exponential growth of the subset size. We therefore develop a ***greedy*** search to find a close-to-optimal solution in a polynomial time. It follows an incremental approach to find a solution. Specifically, it first calculates the error probability when write-verification is not enabled for any write operations ($S_0 = \{\}$) and checks to see if it satisfies the improvement factor. If it does, then the search will be terminated and empty set will be returned as a solution. Otherwise, it will select a disk for which enabling write verification would lower the cost function the most using

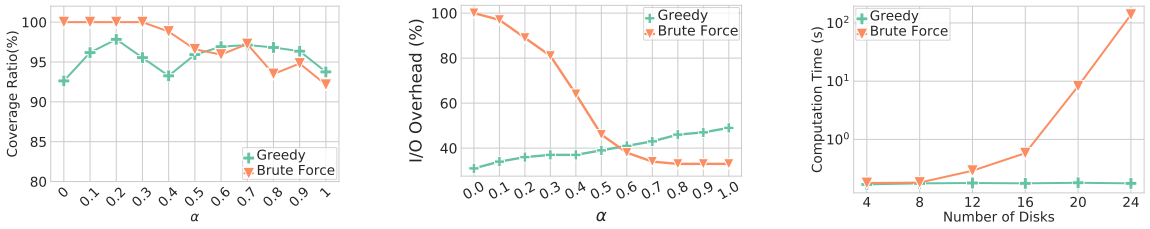
$$S_1 = \underset{S_c \in \{\{i\} \mid i \in S\}}{\arg\min} C(S_c). \tag{8}$$

and add it to the set, S_1. If $E_{S_1}(W)_M$ meets the desired improvement ratio, then it returns S_1 as a solution. Otherwise, it will continue the process by selecting another disk in addition to the disk indexed by the element of S_1. The search will continue until a subset that satisfies the improvement ratio is found or all disks are selected. Since the greedy approach only selects one disk at a time, its time complexity becomes $O(N^2) = N + (N-1) + (N-2) + \cdots + 2$. The results as presented in the next section show that the greedy search performs similar to brute-force approach in terms of mitigating UEs while taking significantly shorter time to execute.

A. Simulation Results

Due to prohibitive cost of producing UEs organically, we rely on simulations to thoroughly evaluate the performance of the probabilistic write verification algorithm. We choose file size, file count, the number of disks, and UE probability of the selected disks as follows: *File Size:* Previous studies show that file sizes in HPC facilities follow a long-tail distribution with median file size is around $1MB$ [37], [38], [9]. We thus used *lognormal* distribution with mean of 14 and standard deviation of 1.5 to simulate file sizes in workloads. It produces files with median size of $1.1MiB$ and average size of $3.3MiB$. Please note that file size does not affect results as we report the percentage of I/O operations that needs to be checked against UE, which is not dependent of file size. Yet, Section V-B evaluates the potential impact of the probabilistic write verification in production file systems using workloads that contain only only one type of files as small, medium, or large. *File Count:* Since HPC application can greatly vary on the number of files they produce, we use geometric distribution to estimate file count in workloads. Geometric distribution represents the probability of the number of successive failures before a success is obtained in a Bernoulli trial which can have only two outcomes as success and failure. We set the probability of success (p) to 0.3, which results in mean value of 3.3 and standard deviation value of 7.7. We also present a detailed analysis on the impact of file count of the performance in Figure 7.

Disk Count: The number of disks, N, depends on the number of files in the workload and file system settings including RAID (e.g., RAID level and RAID stripe size) and file system (e.g., stripe count) configurations. Without loss of generality,



(a) Coverage (b) I/O Overhead (c) Computation Time

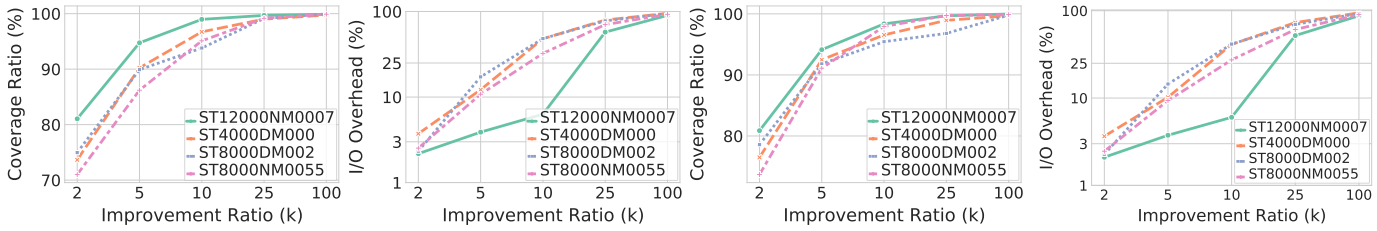
Fig. 3. Performance comparison of brute force and greedy approaches in terms of coverage ratio, I/O overhead, and execution time. The greedy approach when used with $\alpha = 0$ achieves similar coverage and I/O overhead as the brute-force method with $\alpha = 1$, thus it can be used to find a solution quickly.

we set the RAID level to 0 (i.e., no parity or mirroring), the number of disks in RAID arrays to 8, RAID stripe size to $128KB$, and file system stripe count– the number of storage servers used to distribute a file– to 1, as default settings in simulations. Thus, the number of disks used for each file is set to 8, by default. To assign a UE probability for selected disks, we randomly select N SMART logs from the Backblaze dataset which are then evaluated by the XGBoost model for error prediction. We use the SMART logs for December 2020 which contain $300K - 700K$ unique SMART logs depending on disk model. The UE prediction models are then trained using the SMART logs for January 2014 to November 2020. We repeat the simulation for 1 million times for each disk model using randomly selected workload characteristics (i.e., file count and size) and disks (i.e., SMART logs). Since file count generator returns 3 files on average and each file is assumed to be distributed to 8 disks, each SMART log in the test dataset is selected 34 ($\frac{1M \times 3 \times 8}{700K}$ for ST12000NM0007) to 80 ($\frac{1M \times 3 \times 8}{300K}$ for ST8000DM002) times, on average.

Metrics: We define two metrics as *coverage ratio* and *I/O overhead ratio* for performance evaluation. The coverage ratio represents the percentage of write workload that is protected against undetected write errors through write verification. Note that it only considers the portion of data that is directed to disks with UEs. Since the number of disks with UE only constitutes 0.02% of all disks, this results in approximately 0.02% of write data to be issued to disks with error, assuming equal distribution of workload over available disks. Overhead ratio, on the other hand, refers to the percentage of I/O that is verified by reading it back from disk. For example, assume a total of $100GiB$ I/O is written to a file system out of which $100MiB$ is directed to disks with UEs. Also assume that the probabilistic write verification checked $2GiB$ of $100GiB$ I/O against UEs out of which $80MiB$ were on disks with UE. Then, the coverage ratio becomes 80% meaning that $80MiB$ of total $100MiB$ I/O on disks with UEs are protected against UEs by write verification. I/O overhead is equal to 2% as $2GiB$ of total $100GiB$ write workload is read back for write verification. The goal of the probabilistic write verification model is therefore to maximize coverage ratio while keeping minimizing I/O overhead as described in Equation 7. Unless otherwise specified, we set the improvement ratio, k, to 2, which requires finding a set of disks, S_c that will lower the risk of uncorrectable errors by at least two times compared to default error probability, $E(W)$, as described in Equation 2.

Brute force vs greedy: Figure 3 compares the performance of brute force and greedy approaches for varying α values as defined in Equation 7. We observe that α value has a limited impact on the coverage ratio of the greedy solution as improvement ratio 90% is sufficient enough to check high-risk disks against UEs. On the other hand, the brute force approach yields 100% I/O overhead when $\alpha = 0$ as it chooses the subset with all disks to lower $E(T)_{model}$. In exchange, it yields 100% coverage ratio by capturing all UEs. Brute force, however, achieves 92% coverage ratio with 31% I/O overhead when α is set to 1. Higher α values lead $10 - 15\%$ increased in I/O overhead for the greedy method. This is because it selects disks with smallest write I/O size first to lower data size portion of the cost function, $\sum_{i \in S_c} b_i$, but those disks may fall short to satisfy the improvement ratio requirement for the UE rate. Consequently, it will then move to disks with larger I/O size, resulting in a suboptimal solution. As an example, if we are given a list of integers $\{1, 2, 10\}$ and asked to find a subset whose sum of elements is the smallest value that is greater than 5, then the greedy approach will end up with a subset that contains all three numbers due to selecting the smallest number in each iteration. As a result, the greedy solution might yield suboptimal results when α value is close to 1. We leave its optimization under such circumstances as a future work and use $\alpha = 0$ in the rest of the experiments as it yields competitive results compared to brute force method with $\alpha = 1$ (33% vs 31% in I/O overhead and 92% vs 93% in coverage ratio). Figure 3(c) compares the execution time to find a solution with greedy and brute force approaches for increased number of disks. Note that although the number of disks used for a single file is set to 8 by default, total number of disks used for a workload can be high if a workload contains multiple files or file system striping is enabled. It is obvious that brute force method is not feasible approach when the number of disks is larger than 20. On the contrary, the execution time for greedy is smaller than one second even when searching for a solution with 100 disks. Therefore, we used the greedy approach in the rest of the analysis as it yields competitive results while taking significantly less time to find a solution.

Batch vs Streaming: Write verification can be implemented in two main ways for workloads that create multiple files over time. The first approach (*batch*) either assumes that workload characteristics (i.e., file count and sizes) and selected disks



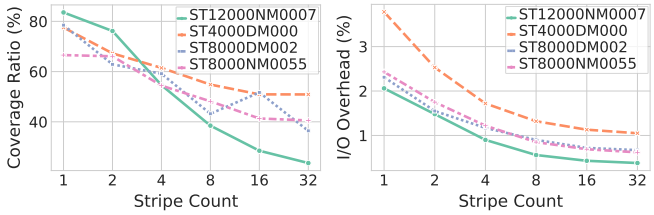
(a) Batch - UE Coverage (b) Batch - I/O Overhead (c) Streaming - UE Coverage (d) Streaming - I/O Overhead

Fig. 4. Uncorrectable error detection rate and I/O overhead of the probabilistic write verification model for batch (a and b) and streaming (c and d) workloads. Improvement ratio (k) of 2 can help to capture $53-73\%$ of UEs with less than 5% I/O overhead.

are known ahead of time or waits for all write operations to complete learn workload information and used disks. The second (*streaming*) approach, on the other hand, makes no assumption and gives write verification decisions as new files are created over time. The batch approach has a potential benefit of obtaining a better solution as it has a global view of the workload and selected disks. However, it requires an additional support from the application. In the case it assume prior knowledge of workload and disk information, it necessitate applications to make accurate estimation about workload characteristics and selected disks to create files, which can be nearly impossible to achieve. In case it waits for workloads to complete, it demands applications to cache (or reproduce) write operations to recover from UEs since deferring the write verification decision until all files are created will prevent file systems to keep the data in its cache. Streaming approach alleviates this risk as it makes write verification decisions as soon as new files are created, thus it is easier to retrieve the corrupted data from storage node caches and reissue the write operation with correct data. Please note that while the streaming method is unable to change the decisions it has given previously, it can keep track of previous decisions and adjust future decisions accordingly.

Altough Section V formulates the probabilistic write verification for batch workloads, the greedy model can be extended to operate in streaming manner as follows: When the first file f_1 is created at time $t = 1$, the greedy model estimates the UE probability for it, $E(W)_M$ (Equation 2), then selects a subset disks to enable write verification such that the improvement ratio requirement can be met. When a second file, f_2 is created at time $t = 2$, the model recalculates $E(W)_M$ considering unverified disks of f_1 and all disks used for f_2. If the new $E(W)_M$ is too high to meet the improvement ratio requirement, then the model will start selecting disks of f_2 one by one for write verification until $E(W)_M$ is lowered sufficiently or all disks are selected. Note that the streaming model can also operate at periodic intervals to process a set of files that are created at around same time. Yet, the results for streaming method are collected under the one file at-a-time scenario to present its worst case performance as increasing number of files leads to higher overall performance (please refer to Figure 7).

Figure 4(a) and 4(b) demonstrate the impact of improvement ratio on the performance of probabilistic write verification

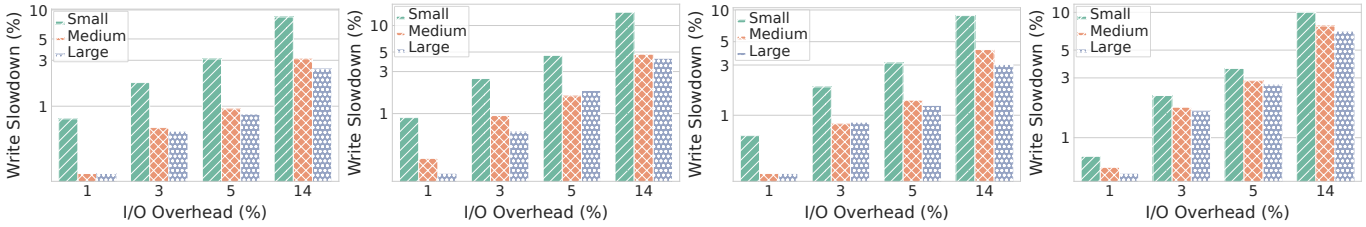


(a) UE Coverage (b) I/O Overhead

Fig. 5. File system striping can be used to reduce I/O overhead in exchange of reduced UE coverage rate.

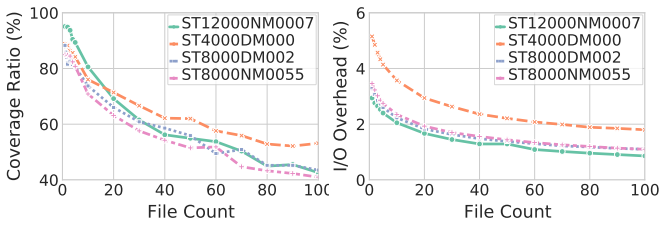
using the batch selection method. As expected, increasing improvement ratio leads to higher coverage ratio as it executes the write verification on more disks to lower the UE rate. Checking more disks, in turn, leads to higher I/O overhead. For example, when the improvement ratio is set to 2, more than 70% of UE coverage can achieved with less than 4% I/O overhead. On the other hand, the improvement ratio of 5 can increase the UE coverage to over 88% in exchange of increasing the I/O overhead up to 17%. Figure 4(c) and 4(d) show the performance of the probabilistic model when using the streaming approach to execute the probabilistic write verification model. Surprisingly, the model yields slightly higher coverage ratio (around 3%) despite attaining similar I/O overhead. This could be attributed to the fact that the streaming method tends to check more files for write verification than the batch method since it can only make decision for one file at a time. On the other hand, the batch model typically checks fewer large files for write verification because their probability of UE exposure is higher compared to small files due to I/O size. As a result, the batch method mostly protects large files against UE exposure even if they are issued to "relatively" low risk disks. Since the streaming method is more convenient to use and achieves higher coverage ratio, we present the rest of the results when the write verification model is executed using the streaming method.

File System Striping: We also assessed the impact of *file system striping* on the performance of probabilistic integrity verification for a fixed improvement ratio of 2. File system striping is used to split and store files on multiple storage servers to improve I/O throughput. While the default striping level is set to 1 in most production parallel file systems, the use of higher values is advised for large files for improved I/O throughput. Figure 5 shows that increasing striping can



(a) Expanse (b) Stampede2 (c) Darwin (d) Campus Cluster

Fig. 6. Impact of probabilistic write verification on write performance in production distributed file systems for small (4KiB-4MiB), medium (250MiB-750MiB), and large (1GiB-20GiB) workloads. 1% read overhead leads to less than 0.9% slowdown in write performance.



(a) UE Coverage (b) I/O Overhead

Fig. 7. For a fixed improvement ratio of 2, increasing number of files in workload lowers the I/O overhead from around 5% to less than 2% in exchange of significant reduction in coverage ratio.

considerably lower I/O overhead for all disk models. For example, while I/O overhead is between 2% and 3.5% when stripe count is set to 1, it reduces to less than 1% for all disk models for stripe count of 32. In return, the coverage ratio drops from around 70% to $25-50\%$ range. The reduction in I/O overhead can be attributed to the fact that as the number of disks increases, I/O amount per disk decreases for the same workload, letting the probabilistic write verification model to make more precise disk selection decisions to meet the improvement ratio.

File Count: Finally, Figure 7 presents the impact of the number of files in workload on incurred I/O overhead for a fixed improvement ratio of 2. As mentioned above, each file is distributed to 8 disks, hence the number of disks used for a workload is proportional to the number of files in the workload. While I/O overhead ranges between 2.9% and 5.1% when there is only one file in a workload, it reduces to nearly 2% as the number of files are increased to 100. In exchange, UE coverage ratio decreases from over 95% to nearly 40%. Similar to file striping results, this can be attributed to increase in the number of disks used for a workload.

B. Overhead Analysis on Production File Systems

In this section, we evaluate the impact of I/O overhead on write performance in production HPC clusters. In other words, since a probabilistic integrity verification reads some portion of a workload back from disk to check against UEs, this will increase the execution duration of write operations. Thus, we measure the slowdown rate of write operations under various I/O overhead rates that the probabilistic model can incur to quantify its impact on application performance.

We conducted tests on Expanse [39] (located at San Diego Supercomputing Center, CA), Stampede2 [40] (located at Texas Advanced Computing Center, TX), Darwin [41] (located at University of Delaware, DE), and campus HPC clusters. Expanse, Stampede2, and Darwin use Lustre parallel file system whereas the campus cluster relies on GPFS (aka IBM Spectrum Scale) file system to handle I/O workload of HPC applications. We created three workloads as small ($4KiB-4MiB$), medium ($250MiB-750MiB$), and large ($1GiB-20GiB$) to write files in different sizes. The exact file sizes are determined using a random number generator that picks a value within the file size range of each workload using uniform distribution. The number of files in each workload is adjusted to ensure that the experiment lasts long enough (i.e., more than five minutes) to make sure that the workload size is bigger than Lustre client cache size. The default file system striping is set to 1 in all Lustre clusters, thus each file in a workload is stored in a single Object Storage Target (OST) that are configured with 10-disk RAID-Z2 arrays.

As presented in Figure 4, I/O overhead of the probabilistic model ranges between 2.1% and 3.6% for $k=2$ and between 3.7% and 14.4% for $k=5$. Moreover, Figure 5 and 7 show that the use of file system striping and having a large number of files in a workload can lower the I/O overhead to around 1% while sustaining 50% or higher coverage ratio. Hence, we measured the impact of 1%, 3%, 5% and 14% I/O overhead on the execution time of write workloads. To avoid potentially misleading impact of caching, we created all files in each workload before reading some of them back for write verification. Although is it possible that files can still be cached at the OSTs, we verified that read performance in these experiments is not affected by OST caching as we obtained similar reach throughput when reading files after several hours. We randomly selected file blocks to read back whose total size corresponds to overhead ratio we are trying to evaluate. For example, we read back random file blocks whose total size is around 14.4 GiB when evaluating 3% overhead ratio for a workload with total size of 480 GiB. We repeated each experiment three times and report the average values.

Figure 6 shows that the rate of slowdown is always smaller than the I/O overhead ratio which can be attributed to having higher read throughput compared to write throughput. Specifically, 1% overhead ratio leads to $0.2-0.9\%$ slowdown and 5% I/O overhead leads to $1.3-4.5\%$ slowdown in write performance. It is important to note that both write and read operations in these experiments are issued from Lustre client nodes whereas in actual deployment read operations

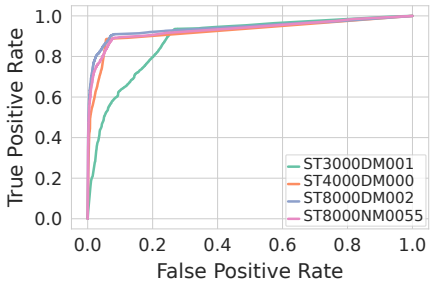


Fig. 8. ROC curve for the XGBoost model when trained with ST12000NM0007 disk model and tested with others.

(as part of write verification) can be executed directly at the storage server to further lower the cost of write slowdown. The results obtained at the campus cluster with GPFS file system (Fig 6(d)) matched with the ones captured in Lustre clusters. *As a result, it is fair to say that the probabilistic write verification model can be used to reduce UE risk by more than* 50% *with negligible (less than* 1%*) impact on write performance.*

TABLE IV
IMPACT OF PROBABILISTIC WRITE VERIFICATION ON TRANSFER TIME

	Write-Verified Files (%)			
	100%	90%	50%	10%
Transfer Time (sec)	360.8s	353.6s	263.0s	168.5s

Impact on application performance: Many file transfer applications used to transfer large scale datasets (e.g., Globus [34], XDRootD [36], and Shift [35]) employ full write verification to avoid undetected errors, which increases transfer times in addition to incurring read overhead on file systems [42], [43]. Thus, probabilistic write verification can be used to lower I/O overhead and improved transfer performance by eliminating write verification for files that are issued to low-risk disks. To validate this, we measure transfer duration of a dataset ($10 \times 1GB$ files) with various write verification levels in a network with $10Gbps$ bandwidth and $30ms$ round trip time. As an example, 10% write verification will conduct the integrity verification on disk data for 9 of the 10 files by reading them back from disk (i.e., write verification) and on cache data (as data is being streamed from the network) for the last file. The results (as given in Table IV) show that while the transfer task takes 360 seconds when the integrity verification process is conducted in a traditional way (i.e., 100% write verification), it takes only 168 seconds (53% reduction) and 263 seconds (27% reduction) when 10% and 50% of all I/O is verified, respectively. Consequently, integrating the probabilistic write verification to file transfer applications can eliminate high overhead on file systems and increase the transfer speeds significantly for file transfers while satisfying desired reduction rate in UE probability.

VI. Discussion on Deployment Challenges

One of the deployment challenges for the proposed method is the accessibility of SMART metrics. Luckily, most disk manufacturers support SMART monitoring system, thus it can be enabled to utilize pre-trained prediction models to decide whether or not disks are likely to develop a UE within next 24 hours. Since SMART reports contain less than 60 attributes, the storage footprint of collecting and storing daily SMART logs would be negligible compared to the scale of today's large-scale parallel file systems. Specifically, one day worth of SMART logs for $120K$ disks in Backblaze dataset took only around $30MiB$ disk space. In addition, we found that it is possible to lower the training data size for UE prediction models to only use last two months of SMART logs without sacrificing the performance considerable, thus it is possible further reduce the storage requirement of SMART logs.

Another potential deployment challenge is an ability to collect and train a UE prediction model for each disk type. While this is a common challenge for any supervised learning models, we show in Figure 8 that a model that is trained for one disk model performs well enough when used for other disk models. Specifically, we evaluate the performance of the XGBoost model when it is trained using SMART logs of ST12000NM0007 disk model and tested against other disk models. The model attained more than 80% TPR with less than 5% FPR for all disk models except ST3000DM001, which is sufficient to capture more than 80% of UEs with minimal overhead. Thus, while data collection is important to improve the performance of the prediction models, pre-trained models can be used to avoid majority of UEs with the help of transfer learning. Another potential solution is incremental learning which involves training prediction models with small amount of initial data and updating them over time as new data becomes available [20], [44].

VII. Conclusion

Uncorrectable errors pose a threat to data integrity in storage systems as they may result in complete data loss when not handled properly. Thus, an ability to predict uncorrectable errors (UE) before they occur allows users and systems administrators to take precautionary actions such as taking high-risk disks offline or verifying the integrity of I/O operations. In this paper, we analyzed $143M$ SMART logs from $106K$ disks over the period of 93 months to gain insights into the characteristics of uncorrectable errors and to derive prediction models. We find that XGBoost classifiers can be used to predict UEs with 90% accuracy with as low as 2% false negative rates. Building on these models, we then introduce a novel probabilistic model to execute UE mitigation strategies while keeping the system overhead at minimum. The model first quantifies the likelihood of UE for a given workload and then selects a minimum set of disks to launch UE mitigation strategies (e.g., write verification) if the calculated error rate is deemed to high for the workload. Without loss of generality, we demonstrated the impact of the probabilistic model on the implementation of *write verification* technique to detect and recover from UEs as soon as they happen. Our extensive simulations along with real-world evaluations show that verifying the correctness of write operations probabilistically leads to the elimination of most write-triggered UEs while incurring negligible impact on write performance. In particular, we show that more than 50% of write-triggered UEs can be avoided with less than 1% decrease in write throughput.

ACKNOWLEDGEMENT

The work in this study was supported in part by the NSF grant 2145742.

REFERENCES

[1] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka *et al.*, "Hacc: Simulating sky surveys on state-of-the-art supercomputing architectures," *New Astronomy*, vol. 42, pp. 49–65, 2016.

[2] "TOP 500," 2022, https://www.top500.org/.

[3] J. L. Hafner, V. Deenadhayalan, W. Belluomini, and K. Rao, "Undetected disk errors in raid arrays," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 413–425, 2008.

[4] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, pp. 1–28, 2008.

[5] A. Krioukov, L. N. Bairavasundaram, G. R. Goodson, K. Srinivasan, R. Thelen, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Parity lost and parity regained." in *FAST*, vol. 2008, 2008, p. 127.

[6] E. W. Rozier, W. Belluomini, V. Deenadhayalan, J. Hafner, K. Rao, and P. Zhou, "Evaluating the impact of undetected disk errors in raid systems," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 83–92.

[7] Y. Zhang, D. S. Myers, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Zettabyte reliability with flexible end-to-end data integrity," in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*. IEEE, 2013, pp. 1–14.

[8] F. Mahdisoltani, I. Stefanovici, and B. Schroeder, "Improving storage system reliability with proactive error prediction," in *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association, 2017, pp. 391–402.

[9] G. K. Lockwood, S. Snyder, S. Byna, P. Carns, and N. J. Wright, "Understanding data motion in the modern hpc data center," in *2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW)*. IEEE, 2019, pp. 74–83.

[10] Y. Zhang, A. Rajimwale, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "End-to-end data integrity for file systems: A zfs case study." in *FAST*, 2010, pp. 29–42.

[11] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "Workout: I/o workload outsourcing for boosting raid reconstruction performance." in *FAST*, vol. 9, 2009, pp. 239–252.

[12] J. Wan, J. Wang, Q. Yang, and C. Xie, "S2-raid: A new raid architecture for fast data recovery," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2010, pp. 1–9.

[13] "Bluewaters," http://www.ncsa.illinois.edu/enabling/bluewaters, 2021.

[14] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 610–621.

[15] "The largest unplanned outage in years and how we survived it," 2017, https://www.csc.fi/web/blog/post/-/blogs/the-largest-unplanned-outage-in-years-and-how-we-survived-it.

[16] B. Allen, "Monitoring hard disks with smart," *Linux Journal*, vol. 2004, no. 117, p. 9, 2004.

[17] M. S. Rothberg, "Disk drive for receiving setup data in a self monitoring analysis and reporting technology (smart) command," May 17 2005, uS Patent 6,895,500.

[18] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu, "Raidshield: characterizing, monitoring, and proactively protecting against disk failures," *ACM Transactions on Storage (TOS)*, vol. 11, no. 4, pp. 1–28, 2015.

[19] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *2013 IEEE 29th symposium on mass storage systems and technologies (MSST)*. IEEE, 2013, pp. 1–5.

[20] S. Han, P. P. Lee, Z. Shen, C. He, Y. Liu, and T. Huang, "Toward adaptive disk failure prediction via stream mining," in *Proceedings of IEEE ICDCS*, 2020.

[21] 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 ({FAST} 20)*, 2020, pp. 151–167.

[22] 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.

[23] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J.-G. Lou *et al.*, "Improving service availability of cloud systems by predicting disk error," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 481–494.

[24] 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.

[25] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2007, pp. 289–300.

[26] 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.

[27] 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 (FAST)*, 2016, pp. 67–80.

[28] A. Oprea and A. Juels, "A clean-slate look at disk scrubbing." in *FAST*, 2010, pp. 57–70.

[29] "Backblaze," 2019, https://www.backblaze.com.

[30] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, p. 1997–2017, jan 2019.

[31] B. Schroeder, S. Damouras, and P. Gill, "Understanding latent sector errors and how to protect against them," *ACM Transactions on storage (TOS)*, vol. 6, no. 3, pp. 1–23, 2010.

[32] J. Tillson, "Disk drive incorporating read-verify after write method," Aug. 24 1999, uS Patent 5,941,998.

[33] A. Riska and E. Riedel, "Idle read after write-iraw." in *USENIX Annual Technical Conference*, 2008, pp. 43–56.

[34] "Globus," 2021, https://www.globus.org/.

[35] "Shift," 2021, https://www.nas.nasa.gov/hecc/support/kb/shift-transfer-tool-overview_300.html.

[36] "XRootD," 2021, https://xrootd.slac.stanford.edu/doc/dev50/xrd_config.htm.

[37] F. Wang, H. Sim, C. Harr, and S. Oral, "Diving into petascale production file systems through large scale profiling and analysis," in *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems*, 2017, pp. 37–42.

[38] Z. Liu, R. Lewis, R. Kettimuthu, K. Harms, P. Carns, N. Rao, I. Foster, and M. E. Papka, "Characterization and identification of hpc applications at leadership computing facility," in *Proceedings of the 34th ACM International Conference on Supercomputing*, 2020, pp. 1–12.

[39] "Expanse," https://www.sdsc.edu/services/hpc/expanse/, 2021.

[40] "Stampede2," https://www.tacc.utexas.edu/systems/stampede2, 2021.

[41] "Darwin," https://dsi.udel.edu/core/computational-resources/darwin/, 2021.

[42] B. Charyyev, A. Alhussen, H. Sapkota, E. Pouyoul, M. H. Gunes, and E. Arslan, "Towards securing data transfers against silent data corruption," in *IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing, IEEE/ACM*, 2019.

[43] B. Charyyev and E. Arslan, "Riva: Robust integrity verification algorithm for high-speed file transfers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1387–1399, 2020.

[44] M. Arifuzzaman and E. Arslan, "Swift and accurate end-to-end throughput measurements for high speed networks," in *The Network Traffic Measurement and Analysis Conference*, 2022.